Contents

Machine Learning Theory

Unsupervised Learning

Appliactions

Schedae Informaticae Vol. 25 (2016): 9–23
doi: 10.4467/20838476SI.16.001.6182

tfml 2017
theoretical foundations
of machine learning, Kraków

Extreme classification under limited space and time budget

Kalina Jasinska[1], Krzysztof Dembczyński[1],
Nikos Karampatziakis[2]
[1]Institute of Computing Science, Poznan University of Technology, Poznań, Poland
[2]Microsoft Research, Redmond, USA
E-mail: *{kjasinska,kdembczynski}@cs.put.poznan.pl*, *nikosk@microsoft.com*

Abstract. We discuss a new framework for solving extreme classification (i.e., learning problems with an extremely large label space), in which we reduce the original problem to a structured prediction problem. Thanks to this we can obtain learning algorithms that work under a strict time and space budget. We mainly focus on a recently introduced algorithm, referred to as LTLS, which is to our best knowledge the first truly logarithmic time and space (in the number of labels) method for extreme classification. We compare this algorithm with two other approaches that also rely on transformation to structured prediction problems. The first algorithm encodes original labels as binary sequences. The second algorithm follows the label tree approach. The comparison shows the trade-off between computational complexity (in time and space) and predictive performance.

Keywords: supervised learning, space and time complexity of learning algorithms, extreme classification, multi-class classification, learning reductions

1. Introduction

Extreme classification refers to multi-class and multi-label problems where the size m of the output space is extremely large. This type of problems appears in many

Received: 11 December 2016 / Accepted: 30 December 2016

application areas of machine learning, such as recommendation, ranking, and language modeling. The extreme setting brings a lot of challenges, such as, time and space complexity of training and prediction, long tail of labels, missing labels and very few training examples per label.

A naive solution for the problem is to train an independent model for each label individually. This approach, often referred to as one-vs-all (OVA), has linear time and space complexity in the number of labels. Unfortunately, in many real world problems this complexity is too costly. One of the main challenges of extreme classification is to reduce the complexity, bearing in mind the need to control the trade-off between lowering the complexity and retaining good predictive performance. To achieve this goal, we consider a new approach for solving extreme classification, which casts the original problem to a structured prediction problem.

An example of a simple structured prediction problem is sequence labeling in which a categorical label is assigned to each member of a sequence of observed values. The aim is to find the most probable labeling for a given sequence. Finding the best output for a given example usually relies on performing a complex inference task. For simple problems, different variants of Viterbi algorithm [1, 2] can be applied. For more complex problems, advanced search techniques are used to explore the large output space efficiently [3]. A structured prediction problem can be treated as a complex multi-class classification problem over all possible label assignments to the sequence. In this paper, we consider the opposite approach that casts classification to structured output prediction. The simplest transformation of this type is to encode labels by sequences of bits. The length of the sequence does not have to be the same for each label. Then, by choosing a proper dependence structure between bits and using appropriate training and inference methods one can get a very compact representation of an extreme classification problem. In this new reduction framework, we can formulate the problem as optimizing the predictive performance under limited time and space budged.

Recently, Jasinska and Karampatziakis [4] have introduced a truly log-time and log-space training and prediction algorithm that can produce its top k predictions in time $O(k \log(k) \log(m))$ for an output space of size m. To do so, the algorithm, referred to as LTLS, encodes labels as paths in a trellis of width 2. The inference consists in finding the longest weighted path in the trellis from the source node to the sink node. Each weight associated with an edge in the trellis is obtained from a binary classifier trained by stochastic gradient descent. To perform efficient inference a variant of Viterbi algorithm is used. In the following we compare this approach to two other methods. The first method is also logarithmic in time and space, but treats all the code bits to be independent. We refer to this method as sequences of independent bits (SIB). The second method are probabilistic classifier trees (PCTs) [5, 6]. In this method the labels are coded by paths in a tree. Such a tree can be treated as a generalization of the trellis. Unfortunately, for this method we cannot give strict logarithmic bounds on time and space, but as shown in [5] this method is statistically consistent. In this paper we focus on multi-class problems, however, PCTs and LTLS can also be easily modified to multi-label problems [6, 4].

The paper is organized as follows. The next section shortly describes state-of-the-art methods for extreme classification. Section 3 states the problem formally. In Section 4 and 5 we present the SIB and PCT algorithm, respectively. Section 6

describes a variant of LTLS that is made to be similar to the previous methods in order to perform a fair comparison between these different approaches. Section 7 presents experimental results. Last section concludes the paper.

2. Related work

There are several groups of extreme classification algorithms that follow different paradigms such as sparsity, low-rank approximation, tree-based search, or label filtering.

The sparsity-based methods can reduce model size and sometimes training and prediction times due to fewer operations. An example of such an approach is PD-Sparse [7], where the authors show that it is possible to get accurate sparse models in high dimensional datasets. However sparsity is not guaranteed to reduce the model size without severely hurting model accuracy. Examples of the low-rank methods, also called embedding methods, are SLEEC [8], LEML [9], WSABIE [10] or Rembrandt [11]. These techniques can be thought of as (supervised) dimensionality reduction followed by an OVA classifier. All these approaches still remain linear in the size of the output space during training and prediction unless additional approximations are employed, such as subsampling of the negative labels.

The tree-based approaches can be divided into decision tree- and label tree-based methods. Those methods reduce prediction time, but not necessary lead to models with space complexity that is logarithmic in the number of labels. For example, FastXML [12] builds a tree of depth logarithmic in the number of training examples. The multi-class logarithmic time prediction is also addressed by LOMtree [13]. Label tree-based methods such as PCT, discussed later in this paper, have usually $O(\log(m))$ training time, since an update with one training instance is applied to $O(\log(m))$ models. Even though these algorithms reduce prediction time significantly, by not querying all the models, their complexity in general is greaten than $O(\log(m))$.

The last group of algorithms assumes that learning can be performed off-line (so the complexity of training is allowed be higher) and focuses on the use of appropriate data structures to accelerate classification of test examples in the prediction phase. Therefore, this approach is sometimes called label filtering [14] as it avoids a linear scan over all labels. The label partitioning for sublinear ranking method [15] uses clustering to group training examples, and then assigns a set of possible labels to each group in a way that optimizes the overall performance. The clustering step is used only for filtering the labels and is performed independently from training a final model which can be any multi-class or multi-label classifier, even very expensive. During classification a test example is first assigned to one of the groups, and then the final model is called only for labels assigned to this group. Other approaches use Bloom filters [16], filtering lines [14] or tree structures [17]. In case of linear models (e.g., logistic regression, perceptron, the last layer in a deep network), the problem of speeding up classification of test examples is often referred to as maximum inner product search (MIPS) [18, 19]. An exact solution can be optimally obtained by the so-called threshold algorithm [20] which can be used in a variety of machine

learning tasks [21]. However, this algorithm does not scale well to extreme classification. Therefore approximate algorithms need to be considered. The MIPS problem is similar, but not equivalent, to the nearest neighbor search. It is therefore possible to adapt approximate nearest neighbor algorithms to this problem. For example, a modified variant of the locality-sensitive hashing [22] has been introduced in [19]. Let us also emphasize that the MIPS problem can be applied during training as a specific instance of negative sampling [18].

3. Problem Setting

In the following we consider multi-class classification problems. We denote with $(\boldsymbol{x}, y)$ a multi-class instance, where $\boldsymbol{x}$ is a feature vector, $\boldsymbol{x} \in \mathbb{R}^d$, and y a label, $y \in \{1, \ldots, m\}$. We focus on classification methods that are optimized for precision@k. In case of $k = 1$, this performance measure corresponds to accuracy, or stated differently, to:

$$\mathit{precision}@1 = 1 - \ell_{0/1}(y, f(\boldsymbol{x})) = 1 - [\![y \neq f(\boldsymbol{x})]\!] ,$$

where $\ell_{0/1}(y, f(\boldsymbol{x}))$ is the 0/1 loss and $f(\boldsymbol{x})$ a multi-class classifier.

The methods we consider rely on representing labels as binary sequences. The difference between methods lays in the choice of encoding and the assumed dependence structure between elements of the sequence. In general, this approach reduces the original problem to a bunch of binary subproblems. The task is then to appropriately transform original training examples to binary examples for each subproblem. During prediction, the binary outcomes are decoded to original labels. This inference task can vary depending on the chosen encoding and dependence structure.

We analyze each method under a strict time and space budget. For example, we would like to have budget that is logarithmic in the number of labels. We follow here the learning reduction framework [23] which studies a decomposition of complex learning tasks into simpler problems for which numerous and powerful algorithms are available. This decomposition should guarantee that a solution to the simple subproblems gives a solution to the original problem [24]. Ideally, a no-regret (i.e., optimal) solution to each base problem should translate into an optimal solution to the original problem. In such case, a reduction (i.e., decomposition) is called *consistent*. It is, however, an open question whether we can obtain consistent reductions under a strict time and space budget.

We assume that the time complexity of training of a binary classifier with respect to a single training example is $O(1)$. Similarly, calling a binary classifier for a single test example is also $O(1)$. The space complexity is harder to formalize in this way. We assume, however, that storing a single binary classifier is also $O(1)$. In this way we can easily express the computational complexity in terms of the number of labels. In the analysis, we do not take into account the space complexity needed for storing the mapping between original labels and bit sequences, which in general is $O(m)$, and the time complexity needed to encode labels to bit sequences.

4. Simple coding by sequences of independent bits

We start our discussion with a very simple algorithm. It relies on encoding labels as sequences of independent bits in such a way that each label is assigned to one and only one binary code. We refer to this approach as sequences of independent bits (SIB). In general, we encode an original label y by a binary code $c(y)$ of length l. Let $c_i(y)$ indicate the i-th bit in the code. If we assume that bits of the code are independent, then the probability of label y can be obtained by:

$$P(y \,|\, \boldsymbol{x}) = P(c(y) \,|\, \boldsymbol{x}) = \prod_{i=1}^{l} P(c_i(y)|\boldsymbol{x}) \,.$$

To get the model, we need to train base classifiers that estimate $P(c_i(y)|\boldsymbol{x})$. We can use any method that estimates probabilities, for example, logistic regression.

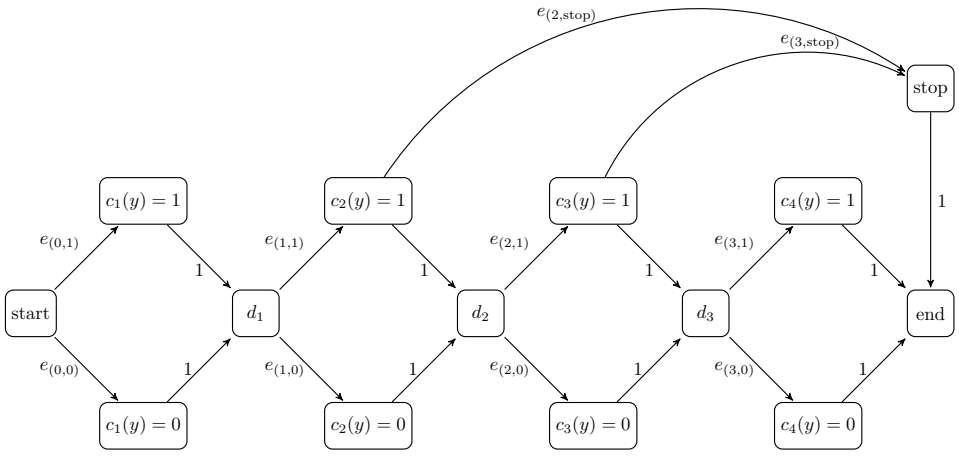


Figure 1. A trellis used in SIB for $m = 22$. Node *start* is connected to both states of the first bit; both states of the last bit are connected to node *end*. To handle an arbitrary number of classes m we connect to the *stop* node the *up* states at bits corresponding to 1's in binary representation of m.

When the number of labels is a power of 2, then we can use binary coding of fixed length. In such a case, learning and prediction with SIB is straight-forward. Otherwise, we need to use a specific coding to make the method logarithmic in time and space with the number of labels. We use here encoding similar to the one used in LTLS [4]. The codes of labels can be visualized as paths in a trellis with additional by-pass edges (see Figure 1). The binary states of bits are represented by *up* and *down* nodes (for the i-th bit, these are nodes $c_i(y) = 1$ and $c_i(y) = 0$, respectively). Moreover, we use auxiliary nodes, *begin*, *end*, and *stop*. The first two indicate the start and the end of the code. The *stop* node determines an early stop of the code. The intermediate nodes

d_i are used to show independence of bits. In this representation, we have exactly m paths, one for each label. More formally, let $2^a \leq m \leq 2^{a+1}$. Then, the first 2^a labels can be coded using a vanilla binary code on a bits. The rest of labels $b = m - 2^a$ are coded using shorter codes. We assume that the last bit in such code is always set to 1. In that way we can encode arbitrary number b of labels, $b \in \{1, 2^a - 1\}$. If the code ends after the i-th bit, then we get additional 2^{i-1} codes. The use of this code, however, requires to train a base classifier for additional *stop* class, for each bit i we use to extend the number of codes.

All incoming edges to *up*, *down*, and *stop* nodes are associated with a probability estimation function $Q_e(\boldsymbol{x})$. To indicate an edge we use a pair $(i, c_i(y))$, where i is the bit and $c_i(y)$ its value. For all other edges, we use weight equal 1. Thanks to this convention, prediction of the most probable label corresponds to finding the most probable path. To this end, we can use dynamic programming, which in context of probabilistic models is known as the Viterbi algorithm [1]. The top-k scoring paths can be found by a modification of the Viterbi algorithm called List Viterbi [2]. Let us remark that the upper bound of the number E of edges with probability estimates in the trellis is $3\lfloor \log_2 m \rfloor$. So, the space complexity of the model is logarithmic in the number of labels. Since the Viterbi algorithm is also linear with the number of edges, the time complexity is also logarithmic.

To compute estimates $P(c_i(y)|\boldsymbol{x})$, we train edge classifiers $Q_{(i,c_i(y))}(\boldsymbol{x})$ in such a way that:

$$Q_{(i,0)} + Q_{(i,1)} = 1\,.$$

Moreover, we need to train additional classifiers that check the early stop of the code, i.e., $Q_{(i,\text{stop})}$. In such a case, we can assume:

$$Q_{(i,0)} + Q_{(i,1)} + Q_{(i,\text{stop})} = 1\,.$$

Let us underline that the code described above is not entirely binary, as it includes additional symbol *stop*.

The independence assumption made in SIB is rather unrealistic. Therefore, this method will often lead to a very crude approximation. However, this simple approach can be treated as a good baseline for other methods with a strict time and space budget, since its time and space complexity is logarithmic with the number of labels. Let us also remark that SIB resembles the ECOC (Error-Correcting Output Codes) approach for classification [25]. ECOC uses, however, codes with additional redundancy, what is not a case of SIB.

5. Probabilistic classifier trees

In contrast to SIB, probabilistic classifier trees (PCTs) [5] take all dependences between bits into account. Moreover, they use prefix codes, so there is no need to use any additional symbol, like *stop* in case of SIB. For label y coded by $c(y)$ of length l its

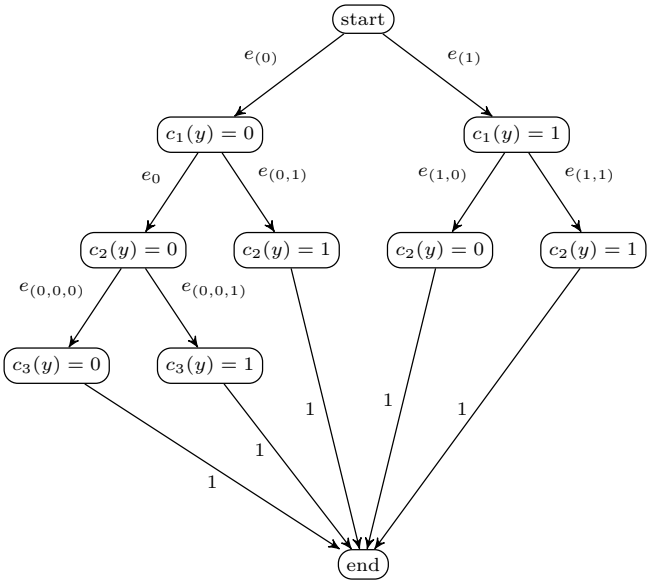


Figure 2. A tree used in PCT for $m = 5$. All leaf nodes are additionally connected with node *end* to resemble the trellis used in SIB.

conditional probability in PCT is given by:

$$P(y \,|\, \boldsymbol{x}) = P(c(y) \,|\, \boldsymbol{x}) = \prod_{i=1}^{l} P(c_i(y)|c_{i-1}(y), \ldots, c_1(y), \boldsymbol{x}) \,.$$

In other words, each bit of the code depends on all antecedent bits. This formulation is known as the chain rule of probability and holds for any joint distribution, i.e., any distribution can be factorized in this way. Therefore PCTs are statistically consistent [5].

Let us recall that any prefix code can be represented by a tree with 0/1 splits. If the number of labels is a power of 2, then binary coding of fixed length can be used (in result a fully balanced tree is obtained). Otherwise, one can use complete trees or Huffman coding [26]. Each path from the root to a leaf node corresponds to a code word. Figure 2 shows an example of a coding tree for multi-class classification with 5 labels. To make the representation of the tree to be similar to the trellis used in SIB, we denote the root node as the *start* node and added the *end* node. All nodes except those incoming to the *end* node are associated with probability estimators $Q_e(\boldsymbol{x})$. To indicate an edge, we use bits on a path from the *start* node to the node to which the edge directs, i.e., $(c_1(y), \ldots, c_i(y))$. To compute estimate $P(c_i(y)|c_{i-1}(y), \ldots, c_1(y), \boldsymbol{x})$, we train edge classifiers $Q_{(c_1(y),\ldots,c_i(y))}(\boldsymbol{x})$ in such a way that:

$$Q_{(c_1(y),\ldots,c_{i-1}(y),0)}(\boldsymbol{x}) + Q_{(c_1(y),\ldots,c_{i-1}(y),1)}(\boldsymbol{x}) = 1 \,.$$

It is easy to see that there are $m - 1$ classifiers in the tree (i.e., one classifier per internal node of the tree). Therefore, the space complexity of this method is linear

in the number of labels. However, by using the hashing trick [27] jointly over all models, the space complexity can be made constant. One can also use model sharing to reduce the space complexity, for example, by using one model for each tree level. The learning time for Huffman and balanced trees is logarithmic, since a given training example is used only in classifiers on a path corresponding to the code of a given label. Prediction can be logarithmic if it is made in a greedy way. PCTs can use, however, more involved search procedures such as uniform-cost search or A* [28, 29]. Thanks to them, the regret of PCTs can be bounded. Moreover, it can be shown that search time is inversely proportional to the probability of the top label. If this probability is lower bounded, then for the balanced trees the top label can be found in the logarithmic time. In the worse case scenario, however, the time complexity of the prediction procedure is linear in the number of labels [28, 5].

Let us notice that similar algorithms to PCTs appear under different names in the literature. In multi-class classification, this method is also known under the name of conditional probability trees [30] and nested dichotomies [31]. The same concept is also known in neural networks and natural language processing under the name of hierarchical softmax [32].

6. LTLS

LTLS, proposed by Jasinska and Karampatziakis [4], is a model that can be situated in-between sequences of independent bits and probabilistic classifier trees. In the following, we consider a specific instance of this approach that resembles maximum entropy markov models (MEMMs). In general, MEMMs take dependencies up to the k-th degree into account:

$$P(y \mid \boldsymbol{x}) = P(c(y) \mid \boldsymbol{x}) = \prod_{i=1}^{l} P(c_i(y) | c_{i-1}(y), \ldots, c_{i-k}(y), \boldsymbol{x}) \,.$$

We use $k = 1$, i.e., the current bit depends only on one previous bit. There are different possibilities of estimating $P(c_i(y)|c_{i-1}(y), \boldsymbol{x})$ and coding of original labels. LTLS undertakes the following approach.

As already mentioned above, SIB use the same encoding as LTLS. The trellis used in LTLS, presented in Figure 3, resembles the one used in SIB. As before, we have *up* and *down* nodes, and three auxiliary nodes, *start*, *end*, and *stop*. The number of bits is also $\lfloor \log(m) \rfloor$. The main difference lays in additional edges that model dependencies between consecutive bits. Therefore, the upper bound of the number E of edges with probability estimates in the trellis is $5\lfloor \log_2 m \rfloor$. To indicate an edge we use here a triple $(i, c_i(y), c_{i-1}(y))$, i.e., $e = (2, 1, 0)$ means that the edge is between the *up* state of the second bit and the *down* step of the first bit. As before, prediction is made by using the Viterbi algorithm.

Training of LTLS is logarithmic in the number of labels. To compute estimates of

$P(c_i(y)|c_{i-1}(y), \boldsymbol{x})$, LTLS trains edge classifiers $Q_{(i,c_i(y),c_{i-1}(y))}$ in such a way that

$$Q_{(i,0,1)} + Q_{(i,1,1)} = 1 \quad \text{and} \quad Q_{(i,1,0)} + Q_{(i,0,0)} = 1\,.$$

As in case of SIB, we also need to train additional classifiers checking the early stop of the code, i.e., $Q_{(i,\text{stop},1)}(\boldsymbol{x})$. In such a case, we have:

$$Q_{(i,0,1)} + Q_{(i,1,1)} + Q_{(i,\text{stop},1)} = 1\,.$$

Probability estimators Q can be trained in various ways. In this paper, for the sake of consistency, we use logistic regression. We refer to this method using the name LTLS-LR.

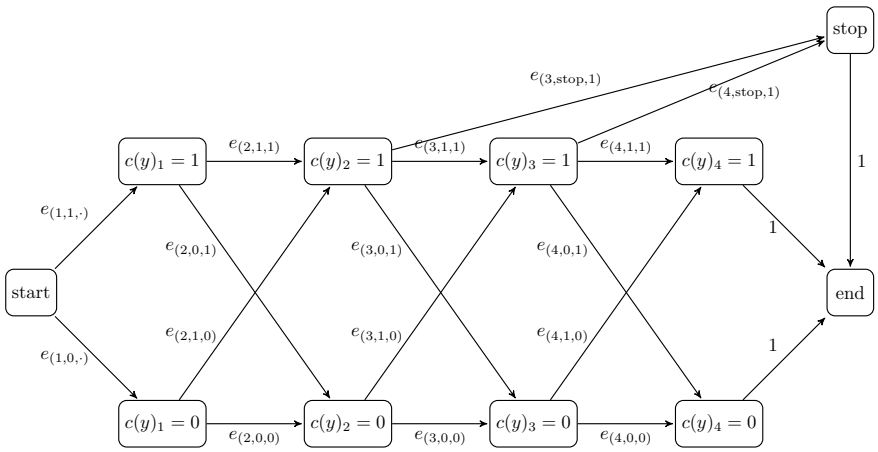


Figure 3. A trellis used in LTLS-LR for $m = 22$. Node *start* is connected to both states of the first bit; both states of the last bit are connected to node *end*. To handle an arbitrary number of classes m we connect to the *stop* node the *up* states at bits corresponding to 1's in binary representation of m. The code corresponding to path $[e_{(1,1,\cdot)}, e_{(2,0,1)}, e_{(3,1,0)}, e_{(4,stop,1)}]$ is $(1, 0, 1)$.

.

One can also consider a learning procedure, which will make LTLS to be very similar to conditional random fields [33]. We do not discuss this possibility in this paper. Let us, however, remark that in the original paper LTLS has been used with a learning procedure that minimizes a variant of structured hinge loss. From this point of view, LTLS can be seen as an instance of structured support vector machines [34]. In this paper, however, we have decided to investigate a variant of LTLS that is as much as possible similar to SIB and PCT to make a fair comparison between these methods.

7. Experiments

This section presents experimental evaluation of analyzed approaches. We report the results of SIB, LTLS-LR and PCT. For comparison, we also include results of two well-known tree-based algorithms, LOMtree [13], and FastXML [12]. We have run SIB, LTLS-LR and PCT on datasets used in [7], where one can find a comparison of a set of multi-class and multi-label classification algorithms in terms of precision@1, prediction time, and model size. The basic information about the datasets is given in Table 1. Table 2 contains performance results of all methods. The results of LOMtree and FastXML are taken from [7]. We do not report here other methods whose results are reported in the cited paper (including PD-sparse introduced therein), since we focus on graph- and tree-based methods, and other methods apply different approach.

We report precision@1 and the model size. We report precision@1 only since this value was given in [7]. To compare the model size in a comprehensive way, we need to remind several important issues. FastXML uses a sparse representation of the weight vectors. LOMtree (implemented in vw[35]) and PCT use feature hashing, so their model size can be treated as constant (i.e., can be set to all available memory; if it is too small then many conflicts between feature weights occur, which deteriorate their performance). The weight vectors of SIB and LTLS-LR are stored in dense representation. Therefore, in case of SIB and LTLS-LR the model size is proportional to $d \times E$, where E is the number of edges (with probability estimators).

Comparing the results of SIB, LTLS-LR and PCT one can see the trade-off between model size and predictive performance. With a growing size of the model, precision@1 also grows. The results of LTLS-LR are comparable to LOMtree and FastXML. On Dmoz dataset LTLS-LR gets results close to LOMtree, using a much smaller model. Results of PCT are competitive to FastXML.

The training and prediction times of all methods are not comparable due to significant differences in implementation, therefore they are not reported. Moreover, even training and prediction times of methods implemented in a similar manner cannot be clearly compared on benchmark datasets, since they depend not only on the number of labels, but also on the number of features and non-zero features in the dataset. Therefore, to show the dependence on the number of labels of SIB, PCT and LTLS-LR, we report results of an experiment on artificial data. Figure 4 shows the prediction times as a function of the number of labels, from 2^4 to 2^{13}, scaled logarithmically. One can notice, that in case of both LTLS-LR and SIB the dependence of the prediction time on the logarithm of m is definitely linear, but with a different constant. The prediction time of PCT does not depend linearly on the logarithm of m, but the average prediction time is definitely sublinear in m.

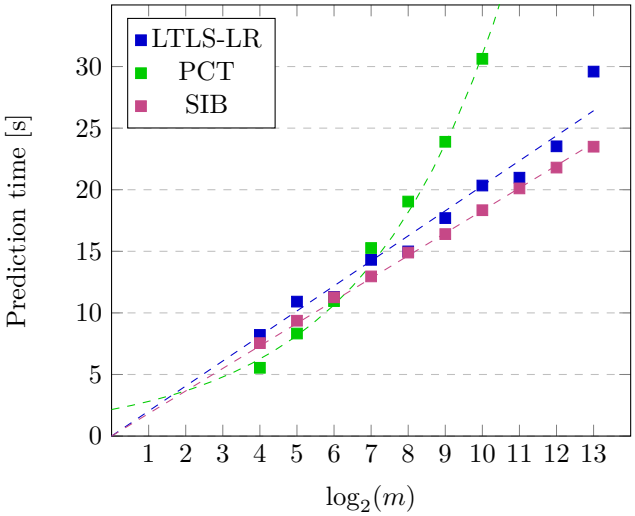


Figure 4. The prediction time, in seconds, of LTLS-LR (blue), PCT (green) and SIB (magenta) on artificial data. The artificial datasets were of the same number of instances and features. The numbers of labels m in the datasets were subsequent powers of 2.

Table 1. Basic statistics of benchmark datasets.

	sector	**aloi.bin**	**Dmoz**	**LSHTC1**
#examples	8658	100000	345068	83805
#features (d)	55197	636911	833484	347255
#labels (m)	105	1000	11947	12294

8. Conclusions

We have introduced a general learning reduction technique that relies on a transformation of a multi-class classification problem to a structured prediction problem. In this way we can control the time and space complexity. By using different codes and dependence structures between elements of the code, we show the trade-off between the predictive performance and the complexity. The preliminary experiments show that algorithms based on the proposed transformation technique can achieve results comparable with the state-of-the-art algorithms that are more costly in terms of time and space. The future work will aim at finding a well-sounded theoretical framework for the reduction of extreme classification to structured prediction problem.

Table 2. Precision@1 and model size [MB] of compared algorithms.

		SIB	LTLS-LR	PCT	LOMtree	FastXML
sector	precision@1	0.8543	0.8616	0.8730	0.8210	0.8490
	model size	6.43	12.06	16.00	17.00	7.00
aloi.bin	precision@1	0.7697	0.8128	0.9088	0.8947	0.9550
	model size	114	209	128	106	992
Dmoz	precision@1	0.1819	0.2082	0.3263	0.2127	0.3840
	model size	215	397	2048	1800	1500
LSHTC1	precision@1	0.0914	0.0950	0.1524	0.1056	0.2166
	model size	272	525	1024	744	308

Acknowledgments

This work has been supported by the Polish National Science Centre under grant no. 2013/09/D/ST6/03917. Large-scale computations have been performed in Poznan Supercomputing and Networking Center.

We would like to thank Wojciech Kotłowski for helpful discussions and valuable insights.

9. References

[1] Viterbi, A.J., *Error bounds for convolutional codes and an asymptotically optimum decoding algorithm.* IEEE Transactions on Information Theory, 1967.

[2] Seshadri, N., Sundberg, C.E.W., *List viterbi decoding algorithms with applications.* IEEE Transactions on Communications, 1994.

[3] Doppa, J., Fern, A., Tadepalli, P., Hc-search: Learning heuristics and cost functions for structured prediction. In: *Journal of Artificial Intelligence Research (JAIR)*, 2013.

[4] Jasinska, K., Karampatziakis, N., Log-time and log-space extreme classication. In: *Workshop on Extreme Classification at Neural Information Processing Systems (NIPS)*, 2016.

[5] Dembczyński, K., Kotłowski, W., Waegeman, W., Busa-Fekete, R., Hüllermeier, E., Consistency of probabilistic classifier trees. In: *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD).* Springer-Verlag 2016.

[6] Jasinska, K., Dembczynski, K., Busa-Fekete, R., Pfannschmidt, K., Klerx, T., Hüllermeier, E., Extreme F-measure maximization using sparse probability estimates. In: *International Confernece on Machine Learning (ICML)*, 2016.

[7] Yen, I.E.H., Huang, X., Ravikumar, P., Zhong, K., Dhillon, I., Pd-sparse : A primal and dual sparse approach to extreme multiclass and multilabel classification. In: *International Conference on Machine Learning (ICML)*, 2016.

[8] Bhatia, K., Jain, H., Kar, P., Varma, M., Jain, P., Sparse local embeddings for extreme multi-label classification. In: *Neural Information Processing Systems (NIPS)*, 2015.

[9] Yu, H., Jain, P., Kar, P., Dhillon, I.S., Large-scale multi-label learning with missing labels. In: *International Conference on Machine Learning (ICML)*, 2014.

[10] Weston, J., Bengio, S., Usunier, N., Wsabie: Scaling up to large vocabulary image annotation. In: *International Joint Conference on Artificial Intelligence (IJCAI)*, 2011.

[11] Mineiro, P., Karampatziakis, N., Fast label embeddings via randomized linear algebra. In: *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD)*, 2015.

[12] Prabhu, Y., Varma, M., FastXML: A fast, accurate and stable tree-classifier for extreme multi-label learning. In: *Knowledge Discovery and Data Mining (KDD)*, 2014.

[13] Choromanska, A., Langford, J., Logarithmic time online multiclass prediction. In: *Neural Information Processing Systems (NIPS)*, 2015.

[14] Niculescu-Mizil, A., Abbasnejad, E., Label filters for large scale multilabel classification. In: *Workshop on Extreme Classification at the International Confernece on Machine Learning (ICML)*, 2015.

[15] Weston, J., Makadia, A., Yee, H., Label partitioning for sublinear ranking. In: *International Conference on Machine Learning (ICML)*, 2013.

[16] Cissé, M., Usunier, N., Artières, T., Gallinari, P., Robust bloom filters for large multilabel classification tasks. In: *Neural Information Processing Systems (NIPS)*, 2013.

[17] Jasinska, K., Dembczynski, K., Consistent label tree classifiers for extreme multilabel classification. In: *Workshop on Extreme Classification at the International Confernece on Machine Learning (ICML)*, 2015.

[18] Vijayanarasimhan, S., Shlens, J., Monga, R., Yagnik, J., Deep networks with large output spaces. In: *Workshop contribution at International Conference on Learning Representation (ICLR)*, 2014.

[19] Shrivastava, A., Li, P., Improved asymmetric locality sensitive hashing (ALSH) for maximum inner product search (mips). In: *Uncertainty in Artificial Intelligence (UAI)*, 2015.

[20] Fagin, R., Lotem, A., Naor, M., Optimal aggregation algorithms for middleware. In: *Principles of Database Systems (PODS)*. ACM 2001.

[21] Stock, M., Pahikkala, T., Airola, A., De Baets, B., Waegeman, W., *Efficient pairwise learning using kernel ridge regression: an exact two-step method.* Computing Research Repository (CoRR), 2016.

[22] Indyk, P., Motwani, R., Approximate nearest neighbors: Towards removing the curse of dimensionality. In: *ACM Symposium on Theory of Computing*, 1998.

[23] Beygelzimer, A., Langford, J., Zadrozny, B., Machine learning techniques-reductions between prediction quality metrics. In: *Performance Modeling and Engineering*. Springer-Verlag 2008.

[24] Beygelzimer, A., Daumé, H., Langford, J., Mineiro, P., Learning reductions that really work. In: *Proceedings of the IEEE*, 2016.

[25] Dietterich, T., Bakiri, G., *Solving multiclass learning problems via error-correcting output codes.* Journal of Machine Learning Research (JMLR), 1996.

[26] Huffman, D., *A method for the construction of minimum-redundancy codes.* Proceedings of the Institute of Radio Engineers (IRE), 1952.

[27] Weinberger, K., Dasgupta, A., Langford, J., Smola, A., Attenberg, J., Feature hashing for large scale multitask learning. In: *International Conference on Machine Learning (ICML)*, ACM, 2009.

[28] Dembczyński, K., Waegeman, W., Cheng, W., Hüllermeier, E., An analysis of chaining in multi-label classification. In: *European Conference on Artificial Intelligence (ECAI)*, 2012.

[29] Mena, D., Montanes, E., Quevedo, J.R., del Coz, J.J., Using a* for inference in probabilistic classifier chains. In: *International Joint Conference on Artificial Intelligence (IJCAI)*, 2015.

[30] Beygelzimer, A., Langford, J., Lifshits, Y., Sorkin, G.B., Strehl, A.L., Conditional probability tree estimation analysis and algorithms. In: *Uncertainty in Artificial Intelligence (UAI)*, 2009.

[31] Fox, J., *Applied regression analysis, linear models, and related methods.* Sage, 1997.

[32] Morin, F., Bengio, Y., Hierarchical probabilistic neural network language model. In: *Artificial Intelligence and Statistics Conference (AISTATS)*, 2005, pp. 246–252.

[33] Lafferty, J.D., McCallum, A., Pereira, F.C.N., Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: *International Conference on Machine Learning (ICML)*, 2001.

[34] Tsochantaridis, Y., Joachims, T., Hofmann, T., , Altun, Y., *Large margin methods for structured and interdependent output variables.* Journal of Machine Learning Research (JMLR), 2005.

[35] Langford, J., Strehl, A., Li, L., *Vowpal wabbit*, 2007 `http://mloss.org/software/view/53/`.

Schedae Informaticae Vol. 25 (2016): 25–35
doi: 10.4467/20838476SI.16.002.6183

tfml 2017
theoretical foundations
of machine learning, Kraków

Misclassification-Driven Sample Relabeling for Supervised Kernel Principal Component Analysis

Krzysztof Adamiak, Krzysztof Ślot
Institute of Applied Computer Science, Łódź University Technology
Stefanowskiego 18/22, 90-924 Łódź, Poland
e-mail: *krzysztof.adam.adamiak@gmail.com, kslot@p.lodz.pl*

Abstract. Supervised kernel-Principal Component Analysis (S-kPCA) is a method for producing discriminative feature spaces that provide nonlinear decision regions, well-suited for handling real-world problems. The presented paper proposes a modification to the original S-kPCA concept, which is aimed at improving class-separation in resulting feature spaces. This is accomplished by identifying outliers (understood here as misclassified samples) and by an appropriate reformulation of the original S-kPCA problem. The proposed idea is to replace binary class labels that are used in the original method, by real-valued ones, derived using sample-relabeling scheme aimed at preventing potential data classification problems. The postulated concept has been tested on three standard pattern recognition datasets. It has been shown that classification performance in feature spaces derived using the introduced methodology improves by 4%-16% with respect to the original S-kPCA method, depending on a dataset.

Keywords: pattern recognition, feature extraction, kernel methods, supervised kernel PCA.

1. Introduction

Common attributes of datasets corresponding to hard, real-world data classification problems are presence of outliers, complex nonlinear and multi-modal character of

Received: 11 December 2016 / Accepted: 30 December 2016

class decision boundaries, uneven representation of classes and class' modes as well as noise and erroneous sample labeling. These problems have to be addressed by pattern recognition procedures that aspire to be of practical use. In fact, all well-established pattern recognition methods that have been developed so far, such as Support Vector Machines (SVM [1]), neural networks (especially trained using deep-learning techniques [2]) or probabilistic classifiers [3], attempt to handle the aforementioned problems. Some of these methods operate on raw data, but typically they assume object representations in carefully selected feature spaces. Therefore, feature space derivation becomes an important element of pattern recognition procedure and an enormous amount of research has been done in this field. Appropriate feature spaces facilitate classification process, eliminate curse of dimensionality problem, thus reducing a risk of classifier overfitting, but also, enable new insights into intrinsic object properties, relations and dependencies that can be revealed by an adopted representation.

A variety of feature-space derivation strategies have been proposed so far. They emphasize various aspects of data representation that are of importance for a given application. Criteria used for derivation of new feature spaces range from maximization of data scatter (PCA and its nonlinear extension - kernel PCA, abbreviated henceforth using the term kPCA), through maximization of sample independence (Independent Component Analysis [4], and its kernelized extension [5]) to maximization of class discrimination (Linear Discriminant Analysis along with its kernelized version and supervised versions of PCA). Other concepts behind a search for reduced representations of samples involve for example preservation of original data structure (as e.g. in Multidimensional Scaling or Iso-mapping).

The presented paper is concerned with a modification of Supervised Kernel Principal Component Analysis (S-kPCA) [6], which is a supervised extension to the kPCA (labeled samples are considered in feature space derivation), proposed in [7]. Kernel PCA in turn generalizes the classical PCA in such a way that the discovered maximum scatter directions become nonlinear. Properties of kPCA address several basic requirements crucial for classification of real-world data, such as low sensitivity to outliers or nonlinear data mapping, which is crucial for solving linearly non-separable problems. Atop on that, S-kPCA provides features that maximize correlations between samples and their class labels, thus eliminating one of the main drawbacks of scatter-maximization based strategies.

Despite numerous advantages, S-kPCA is clearly not an ultimate solution to the problem of feature space derivation. For difficult datasets it fails to provide perfect data separation and one of the reasons behind its deteriorating performance is a lack of diversification of individual samples' role in building new data representation. This issue is explored in research reported in the presented paper. We postulate to diversify significance of different samples by their appropriate relabeling, so that samples that may potentially pose classification problems become more important. We verify this concept on three publicly available pattern recognition datasets and we show that significant improvement (4%-16%, depending on dataset) over the original S-kPCA approach can be obtained.

A structure of the paper is the following. We begin with a short review of related concepts: kPCA and S-kPCA. Then we present in detail the proposed sample relabeling principles and the adopted feature space derivation procedure. Finally, we provide results of experimental evaluation of the concept, where the modified S-kPCA method

is confronted with the original one and feature spaces derived for both approaches are indirectly compared using classification performance results.

2. Related work

A basis for the presented research is laid out by an impressive development of kernel methods for data classification and processing, which followed a success of the Support Vector Classification (SVM) [8, 9, 10, 11, 12]. Theory of kernel methods has been expanded also onto data preprocessing domain and several 'kernelized' versions of well-established concepts were formulated. These include concepts that are directly related to the presented research: kernel Principal Component Analysis and its supervised version S-kPCA.

Kernel Principal Component Analysis, proposed in [7], extends classical Principal Component Analysis concept in order to identify nonlinear data scatter directions. A concept of implicit problem-solving in high-dimensional, intermediate spaces, which can be accomplished using kernels, provides a means for making the relevant computations feasible. An objective of kPCA is to find directions of the maximum variability among samples $\mathbf{x}_i$ that are projected to some high dimensional space, using a transformation $\Phi(.)$ (i.e. $\mathbf{X}_i = \Phi(\mathbf{x}_i)$). In other words, an objective is to find eigenvectors $\mathbf{V} = [\mathbf{v}_0, \mathbf{v}_1, ...]$ of the projected data covariance matrix:

$$(\mathbf{X} - \mathbf{M})(\mathbf{X} - \mathbf{M})^T \mathbf{V} = \Lambda \mathbf{V} \tag{1}$$

where $\mathbf{M}$ is a matrix of mean-valued vectors $\mathbf{m}$, computed for the projections in high-dimensional space, and Λ is a diagonal matrix of eigenvalues. As eigenvectors lie in a subspace defined by projected samples:

$$\mathbf{v}^i = \sum_{j=0}^{n-1} \alpha_j^i (\mathbf{X}_j - \mathbf{m}) = (\mathbf{X} - \mathbf{m})\mathbf{a}^i,$$

premultiplying the equation (1) by the term $(\mathbf{X} - \mathbf{M})^T$ yields alternative formulation of the eigenproblem:

$$(\mathbf{X} - \mathbf{M})^T (\mathbf{X} - \mathbf{M})\mathbf{A} = \Lambda \mathbf{A} \tag{2}$$

where $\mathbf{A} = [\mathbf{a}^0, \mathbf{a}^1, ...]$ comprises vectors of coefficients that become a solution to the modified eigenproblem. Observe, that only dot products are involved in computations of the eigenproblem (2), so they can be replaced by kernels. Introducing a Gramm matrix, with elements $G_{i,j} = \hat{K}(\mathbf{x}_i, \mathbf{x}_j)$, where $\hat{K}$ is some kernel function, centered in high-dimensional space, one can rewrite (2) in a compact form:

$$\mathbf{G}\mathbf{A} = \Lambda \mathbf{A} \tag{3}$$

A solution to (3), which can be found for reasonable amounts of samples, defines directions of the maximum variability in a high-dimensional space and can be used

for projecting unknown samples:

$$(\Phi(\mathbf{z}) - \mathbf{m})^T \mathbf{v}^i = (\Phi(\mathbf{z}) - \mathbf{m})^T (\mathbf{X} - \mathbf{m})\mathbf{a}^i = \left[\hat{K}(\mathbf{z}, \mathbf{x}_0), ...\hat{K}(\mathbf{z}, \mathbf{x}_{n-1})\right] \mathbf{a}^i \quad (4)$$

As it can be seen, projections onto each eigenvector $\mathbf{v}_i$ can be determined in the original, low-dimensional space, using kernel operations and the computed coefficient vectors $\mathbf{a}^i$.

The second concept relevant to the presented paper is a supervised version of kPCA. The proposed idea is to use Hilbert-Schmidt Independence Criterion (HSIC) [13] as an objective function that is to be maximized. HSIC measures a level of cross-covariance between samples and their labels:

$$\mathbf{C}_{x,y} = E(\mathbf{X} - \mathbf{m}_x)(\mathbf{Y} - \mathbf{m}_y)^T = E(\mathbf{XH})(\mathbf{YH})^T \quad (5)$$

where $\mathbf{X}$ is a matrix of input samples with a mean vector $\mathbf{m}_x$, $\mathbf{Y}$ is a matrix of labels, with their mean $\mathbf{m}_y$, and $\mathbf{H}$ is a centering matrix. HSIC uses a Hilbert-Schmidt norm, which, in essence, aggregates squared entries of the cross-covariance (5). It can be easily shown that this can be expressed as:

$$HSIC = k \cdot \mathbf{tr}(\mathbf{C}_{x,y}\mathbf{C}_{x,y}^T) \quad (6)$$

where $\mathbf{tr}$ denotes the trace of a matrix and k is a scaling factor. As the criterion (6) involves dot products, one can introduce kernels: on input samples - $\mathbf{K} = [k(\mathbf{x}_i, \mathbf{x}_j)]$ and on labels - $\mathbf{L} = [l(\mathbf{y}_i, \mathbf{y}_j)]$, and rewrite the criterion in the form:

$$HSIC = k \cdot \mathbf{tr}(\mathbf{KHLH}) \quad (7)$$

An objective of S-kPCA procedure is to find such a transformation matrix $\mathbf{U}$ of original samples $\mathbf{x}$, i.e.:

$$\mathbf{x'} = \mathbf{Ux}$$

which, after plugging $\mathbf{x}$' into (5) provides maximization of the criterion (7). This can be seen as searching for such a combination of original samples that ensure data transformations (through kernel functions) that maximize correlation between samples and their labels.

3. Misclassification-driven sample relabeling

The original S-kPCA procedure is not addressing an issue of diversifying sample labels and is not exploring its impact on discriminative properties of a resulting feature space. By default, all class samples are treated evenly: for example, for a two-class problem (samples belong either to class $\mathcal{A}$ or class $\mathcal{B}$), each sample $\mathbf{x}$ is labeled with a two-element vector $\mathbf{y}_i$ with binary entries:

$$\forall_{s_i \in \mathcal{A}} \quad \mathbf{y}_i = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \forall_{s_j \in \mathcal{B}} \quad \mathbf{y}_j = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad (8)$$

where $\mathcal{A}$ -class sample label is linked to its class by the first component of the label vector and $\mathcal{B}$-class samples, by the second component.

As the criterion (6) that underlies feature space derivation focuses on sample-label correlation, it is evident that varying the label affects the relevant computations and leads to a different solution. One can observe that by varying numerical label representations one can modify sample-class correlations in several possible ways. For the considered two-class problem, where labels are represented by two-element vectors, one can vary any of the two entries. Moreover, one can easily interpret such changes. An increase in a value of sample's 'own' label vector component (the first for $\mathcal{A}$ -class samples and the second for $\mathcal{B}$-class samples) makes a class-sample correlation stronger, whereas decreasing this value - weakens the correlation. This way, one can diversify a significance of samples in a process of constructing a novel feature space. In addition, one can observe that a sample can be forced to negatively correlate with the opposite class. This can be accomplished by substituting the 'zero-correlation' component of the label vector (the second one for $\mathcal{A}$ -class samples and the first one for $\mathcal{B}$-class samples) with a negative value. As a result, every sample would get individual labels that can be represented as:

$$\forall_{s_i \in \mathcal{A}} \quad \mathbf{y}_i = \begin{bmatrix} a_i^p \\ -a_i^n \end{bmatrix}, \quad \forall_{s_j \in \mathcal{B}} \quad \mathbf{y}_j = \begin{bmatrix} b_i^p \\ -b_i^n \end{bmatrix}, \tag{9}$$

where $a..$, $b..$ are positive real numbers and the superscripts p and n identify 'positive' and 'negative' correlation coefficients.

The presented concept of sample relabeling seems a viable way to modify a role of different samples in derivation of new feature spaces. In particular, one can apply this mechanism to increase significance of samples that are harder to be correctly classified over significance of samples that pose no serious classification problems.

To improve discriminative properties of feature spaces derived using S-kPCA concept, one needs to elaborate rules that enable reasonable sample relabeling, i.e. that enable determining for which samples label alterations should be made and what should be a magnitude of such an alteration. We propose to use training sample classification results statistics as the basis for both determination of samples to be affected and determination of amounts of label changes.

The proposed procedure has been schematically depicted in Fig. 1. Original dataset is split into two parts: training/validation and test sets. Feature space derivation is performed only on samples from the former one and begins with its random split into temporary training and temporary test subsets. Next, the original S-kPCA procedure, which assumes binary class labels (8) is executed on the temporary training subset, followed by data classification performed in the derived space using Gaussian Mixture Model (GMM) classifier. All misclassified samples are then recorded and saved for a future use. The GMM classification method has been chosen, as it has a little in common with the adopted kernel-based feature space derivation methodology and can therefore be considered as an unrelated tool for feature space evaluation. For the current split of the training set, classification is performed using a k-fold cross validation scheme and once it is completed, the whole procedure is repeated n-times for another n random splits of the training set into new temporary training/test parts. After completion of this iterative procedure, misclassification percentage is determined for each sample. The computed coefficients are used as a basis for sample

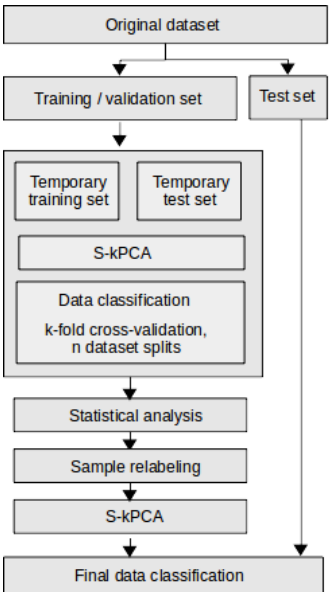


Figure 1. Block diagram of the proposed feature space derivation procedure.

relabeling. Assuming that some $i-$th sample from a class $\mathcal{A}$ has been misclassified $m-$times, the corresponding correction coefficient is determined:

$$c_i = \alpha \frac{m}{n} \tag{10}$$

where α is a positive constant that controls a magnitude of label updates.

The coefficient (10) is then used to produce the 'positive' component of the sample label vector:

$$a_i^p = 1 + c_i \tag{11}$$

or the 'negative' component (updates for samples from a class $\mathcal{B}$ are analogous):

$$a_i^n = -c_i \tag{12}$$

Having the samples relabeled, another S-kPCA procedure is executed, producing the resulting feature space for classification of samples from the test set. Next, classification result is recorded and the whole procedure is repeated p-times for other random splits of the original dataset.

4. Experimental evaluation of the proposed concept

To verify the proposed concept, a series of experiments (using Python's sckit package [14]) on seven different pattern recognition datasets: Digits, Ionosphere, Pima Indian

Table 1. Datasets used in experiments.

Database	Digits	Ionosphere	Pima	Glass	E-Coli	Parkinson	Heart
Classes	10	2	2	6	5	2	5
Attributes	64	34	8	10	8	23	14
Samples	1797	351	768	213	327	195	297

Diabetes, Glass, E-coli, Parkinson's Disease [15] and Cleveland Heart Disease [16] (all datasets available at UCI repository [17]) were performed. Dataset highlights are shown in Table 1. Three of these datasets correspond to binary classification problems, so that sample labels are represented by two-element vectors, as shown in (9). For the remaining datasets, the presented sample relabeling approach requires only a straightforward modification: label vectors are composed of multiple entries that get appropriately updated during the validation procedure.

Three different label alteration scenarios were considered during the experiments:

- Only positive components of sample's label vector (i.e. a_i^p or b_i^p depending on sample's class) were being updated according to (11)
- Only negative components of sample's label vector (i.e. a_i^n or b_i^n) were being updated according to (12)
- Both negative and positive components were modified

For each dataset, a procedure explained in the previous Section was executed. In each case we assumed a 5-fold cross validation (k=5), the inner loop (i.e. estimations of misclassification rates for a given split of the original dataset) was executed 100 times (i.e. n=100) and 100 classifications were made (p=100) to asses an overall classification performance for each method. Throughout all experiments, a Gaussian kernel was used in S-kPCA procedure:

$$k(\mathbf{x}_i, \mathbf{x}_j) = exp\left(-\gamma \cdot \ ||\mathbf{x}_i - \mathbf{x}_j||^2\right) \tag{13}$$

where $||.||$ denotes a distance between samples and γ was chosen using the grid-search method [18].

An objective of the first phase of experiments was to compare the three adopted sample relabeling scenarios. Classification experiments (GMM method was used for data classification both in the validation and in the test step) were performed on the first three databases and the results, plotted as a function of the parameter $\alpha \in [0..2]$ (10) have been shown in Fig. 2. The experiments have been summarized in Table 2. For comparison purposes, performance evaluation of GMM classification of raw data, as well as GMM classification in a space derived using the original S-kPCA procedure (for identical splits into training and test parts) were provided. One can observe that classification in feature spaces derived using the proposed strategy outperforms the reference methods: classification made on raw data and classification made in a space derived using the original S-kPCA. Also, it can be seen that a combination of both

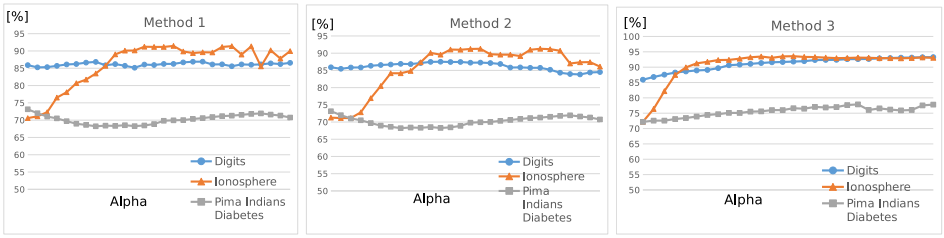


Figure 2. Average classification performance as a function of varying label update magnitudes for all considered scenarios and datasets.

Table 2. Classification performance using Gaussian Mixture Models (in percent) with 95% confidence intervals

Database	Digits	Ionosphere	Pima
Raw data	76,2 ± 3.6	72.4 ± 3.2	63.9 ± 4.7
S-kPCA	85.5 ± 1.8	72.1 ± 5.3	73.3 ± 2
Scenario 1	86.7 ± 1.9	91.4 ± 2.6	72.0 ± 2
Scenario 2	87.5 ± 2.1	91.2 ± 3	72.0 ± 2.4
Scenario 3	**93.2** ± 0.9	**93.0** ± 2	**76.0** ± 1.8

types of updates, i.e. the last scenario used for label alterations, provides the most noticeable gain, so we consider this strategy to be the best one.

An improvement in classification performance is statistically significant for the two datasets: Digits and Ionosphere. In case of the last dataset (Pima Indian Diabetes), although average classification results are better than for the original S-kPCA, large variations of individual results do not allow making any definite statement on the proposed method's superiority. On the other hand one can observe that Pima dataset samples have relatively low initial dimensionality, so that dimensionality reduction in that case may not be necessary at all.

As sample relabeling involving alterations both to positive and negative label vector components has been found to provide the best results, this approach has been adopted in the following experiments. This time, classification performance of three different procedures, involving no dimensionality reduction (raw data classification), dimensionality reduction with original S-kPCA and dimensionality reduction with sample relabeling using the adopted third scenario, were done for all considered datasets. GMM was used as the classification strategy in validation phase, whereas test set samples were classified with either GMM, linear SVM and k-NN ($k = 5$ was used) classifiers.

Performance comparison results are depicted in Fig.3, where the best performing α value for each dataset ($\alpha \in [0.3...1.1]$) was used for sample relabeling.

A few observations can be made from the presented results. The most important from the point of view of the proposed concept is that the proposed sample relabeling always results in feature spaces that have better class discrimination properties than

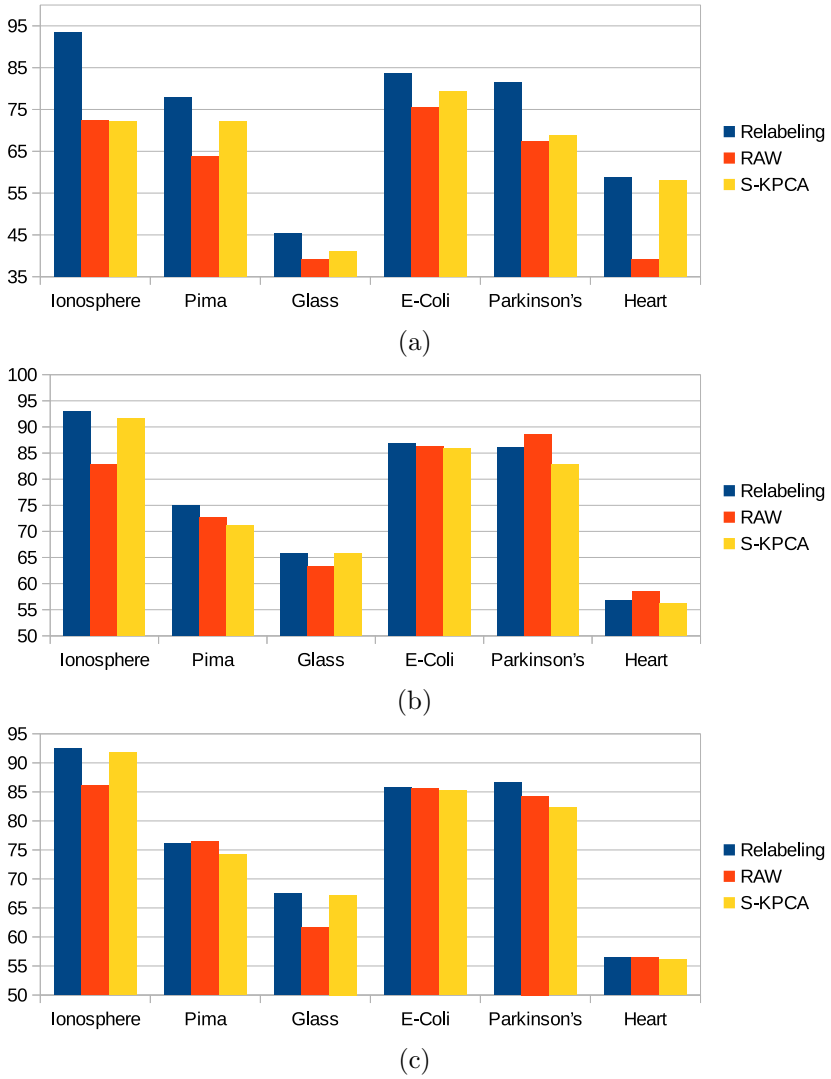


Figure 3. Maximum classification rates for different datasets and the three considered test-set classification methods: GMM (a), 5-NN (b) and linear SVM (c). 'Relabeling' denotes the proposed approach, 'RAW' denotes classification of original data and 'S-KPCA' denotes classification with the original method.

the ones produced by the original S-kPCA. In each case, classification performance improves, however, in several cases this improvement is not statistically significant. Secondly, it seems that classification strategy adopted in validation phase (GMM) implies the best improvements if the same classification strategy is used in the test phase (differences between results obtained for the sample-relabeled and the original S-kPCA are the most salient). Finally, performance for different datasets depends on the adopted classification strategy.

5. Conclusion

A strategy for improving class separation properties for feature spaces derived using Supervised kernel Principal Component Analysis has been presented in the paper. It has been shown that appropriate modifications to sample labels, which diversify their significance in derivation of target space features, result in increased data classification performance and that this gain can be substantial for some datasets. Although the concept needs to be thoroughly verified using many other existing data sources, we believe that the observed tendency will hold, improving significance of the S-kPCA concept.

One needs to bear in mind, that PCA-based data recognition methods are inherently computationally complex, which limits their use in time-critical applications. This also applies to S-kPCA and to the proposed modification. On the other hand, the considered data preprocessing offers several crucial advantages - it reveals structures that exist among data and that can be relevant for class-discrimination, reduces a risk of classifier overfitting and reduces sensitivity to bad class examples.

6. References

[1] Burges, C.J., *A tutorial on support vector machines for pattern recognition.* Data Mining and Knowledge Discovery, 1998, **2**(2), pp. 121–168.

[2] Bengio, Y., *Learning deep architectures for ai.* Foundations and Trends in Machine Learning, 2009, **2(1)**, pp. 1–127.

[3] Reynolds, D., *Gaussian mixture models.* Encyclopedia of biometrics, 2015, pp. 827–832.

[4] Comon, P., *Independent component analysis: a new concept?* Signal Processing, 1994, **36(3)**, pp. 287–314.

[5] Bach, F.R., Jordan, M.I., *Kernel independent component analysis.* Journal of Machine Learning Research, 2002, **3**, pp. 1–48.

[6] Barshan, E., Ghodsi, A., Azimifar, Z., Jahromi, M.Z., *Supervised principal component analysis:visualization,classification and regression on subspaces and submanifolds.* Pattern Recognition, 2011, **44**, pp. 1357–1371.

[7] Schölkopf, B., Smola, A., Müller, K.R., *Nonlinear component analysis as a kernel eigenvalue problem.* Neural Computation, 1998, **10**, pp. 1299–1319.

[8] Hofmann, T., Schölkopf, B., Smola, A.J., *Kernel methods in machine learning.* The Annals of Statistics, 2008, **36**(3), pp. 1171–1220.

[9] Smola, A.J., Schölkopf, B., *Learning with Kernels.* MIT Press, 2002.

[10] Wang, M., Sha, F., Jordan, M.I., *Unsupervised kernel dimension reduction.* Proc. of Conf. Advances in Neural Information Processing Systems, 2010, **23**, pp. 2379–2387.

[11] Mika, S., Rätsch, G., Scholkoph, W.J., Müller, K.R., *Fisher discriminant analysis with kernels.* Proc. of IEEE Conf. Neural Networks for Signal Processing, 1999, pp. 41–48.

[12] Baudat, G., Anouar, F., *Feature vector selection and projection using kernels.* Neurocomputing, 2003, **vol. 55**, pp. 21–38.

[13] Song, L., Smola, A., Gretton, A., Bedo, J., Borgwardt, K., *Feature selection via dependence maximization.* Journal of Machine Learning Research, 2012, **13**, pp. 1393–1434.

[14] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E., *Scikit-learn: Machine learning in Python.* Journal of Machine Learning Research, 2011, **12**, pp. 2825–2830.

[15] Little, M.A., McSharry, P.E., Roberts, S.J., Costello, D.A., Moroz, I.M., *Exploiting Nonlinear Recurrence and Fractal Scaling Properties for Voice Disorder Detection-6.* 12 2011, ,.

[16] Hungarian Institute of Cardiology. Budapest: Andras Janosi M.D., University Hospital Zurich, Switzerland: William Steinbrunn M.D., University Hospital Basel, Switzerland: Matthias Pfisterer M.D., V.A. Medical Center Long Beach and Cleveland Clinic Foundation:Robert Detrano M.D. Ph.D., *Heart Disease Data Set.* [online].

[17] Moshe, L., *UCI machine learning repository*, 2013.

[18] Chapelle, O., Vapnik, V., Bousquet, O., Mukherjee, S., *Choosing multiple parameters for support vector machines.* Machine Learning, 2002, **46**(1), pp. 131–159.

Schedae Informaticae Vol. 25 (2016): 37–47
doi: 10.4467/20838476SI.16.003.6184

tfml 2017
theoretical foundations
of machine learning, Kraków

On Certain Limitations of Recursive Representation Model

Stanisław Jastrzębski, Igor Sieradzki
Faculty of Mathematics and Computer Science
Jagiellonian University, Łojasiewicza 6, 30-348 Kraków, Poland
e-mail: *{stanislaw.jastrzebski, igor.sieradzki}@uj.edu.pl*

Abstract. There is a strong research effort towards developing models that can achieve state-of-the-art results without sacrificing interpretability and simplicity. One of such is recently proposed Recursive Random Support Vector Machine (R^2SVM) model, which is composed of stacked linear models. R^2SVM was reported to learn deep representations outperforming many strong classifiers like Deep Convolutional Neural Network. In this paper we try to analyze it both from theoretical and empirical perspective and show its important limitations. Analysis of similar model Deep Representation Extreme Learning Machine (DrELM) is also included. It is concluded that models in its current form achieves lower accuracy scores than Support Vector Machine with Radial Basis Function kernel.

Keywords: support vector machines, random recursive support vector machine, extreme learning machine, representation learning, stacked generalization

1. Introduction

Successes of deep architectures often comes at the cost of interpretability and fitting complexity [1, 2]. Popular techniques used to battle this problem include random projections, which are a basic building block of Extreme Learning Machines [3, 4],

Received: 11 December 2016 / Accepted: 30 December 2016

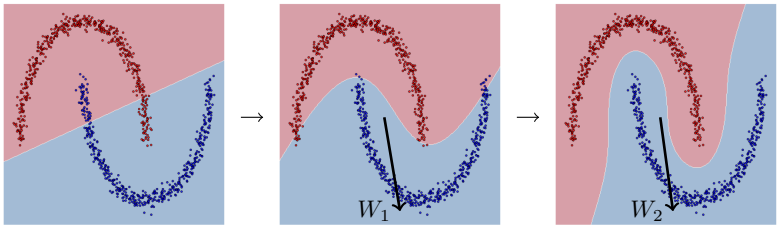


Figure 1. Visualization of R^{2}SVM model on two moon dataset. Random projections were manually adjusted.

where before classification data is projected into a higher dimensional space using random non-linear transformation. Such classifiers are stacked to form a deep architecture that can achieve state of the art results [5]. In this paper we analyze Random Recursive SVM (R^{2}SVM) model proposed by Vinyals et al. [6], which recursively transforms data using predictions from linear layers. We will also cover similar model called Deep Representation Extreme Learning Machine (DrELM) [7]. Both models are following "stacked generalization" introduced by Wolpert et al. [8].

R^{2}SVM uses as linear classifier Support Vector Machines (SVM), model proposed by Vapnik [9], one of the most successful classifiers of the last decade mostly thanks to its well motivated regularization method in the linear case and its efficient delinearization. While shallow learners, like SVM, are usually outperformed by Deep Learning models, combining strengths of principled shallow models and Deep Learning ones is an active field [10].

The most important advantages of R^{2}SVM and DrELM include their interpretability and scalability, while at the same time learning deep representation, as reported by authors of the models. Indeed both linear SVM and non–linear ELM can be fitted in $\mathcal{O}(N)$ time and only single fit is performed for each layer. Both models optimize a convex objective, that has a global minimum. The original papers did not include theoretical discussion of the obtained results and its limitations, which are the main topic of the paper.

R^{2}SVM and DrELM

First we introduce the model in an informal discussion. R^{2}SVM consists of multiple layers transforming recursively input dataset. Let's focus on the binary case, where each layer fits a hyperplane separating the dataset into two subspaces. The two groups are then moved in random opposite directions proportionally to distance of the hyperplane. The main idea behind the model is to separate those groups, which should improve classification performance at later stages. Transformed dataset with applied non-linearity (which prevents some degeneration of the method) is passed to the next layer.

Formally let's consider training dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$, where $\mathbf{x}_i \in \mathbb{R}^d$ is a feature

vector and $y_i \in \{1, 2, \ldots, K\}$ is class label. R^{2}SVM is a recursive model, where each layer transforms dataset using all previous outputs. Let's denote $\mathbf{X}_i \in \mathbb{R}^{N \times d}$ as representation output of the i$^{\text{th}}$ layer (see Fig. 2), $\mathbf{O}_i \in \mathbb{R}^{N \times K}$ is a vector of distances from hyperplanes and $\mathbf{W}_i^j \in \mathbb{R}^{K \times d}$ is a matrix containing random vectors as rows that maps output of i$^{\text{th}}$ layer to input of j$^{\text{th}}$ layer. Each random vector is drawn from standard normal distribution. Then we can write

$$\mathbf{X}_{i+1} = \sigma\left(\mathbf{X} + \alpha \sum_{j=1}^{i} \mathbf{O}_j \mathbf{W}_i^j\right), \tag{1}$$

where $\mathbf{X}$ is the original dataset, α controls size of applied transformation and σ is element–wise sigmoid, see Fig. 2 for diagram.

In the case of DrELM we have a slight modification. Most importantly classifier used is Extreme Learning Machine (with linear activation function). Authors also propose to use only the last output ($\mathbf{O}_i$):

$$\mathbf{X}_{i+1} = \sigma\left(\mathbf{X} + \alpha \mathbf{O}_i \mathbf{W}_i\right), \tag{2}$$

where σ is a sigmoid function.

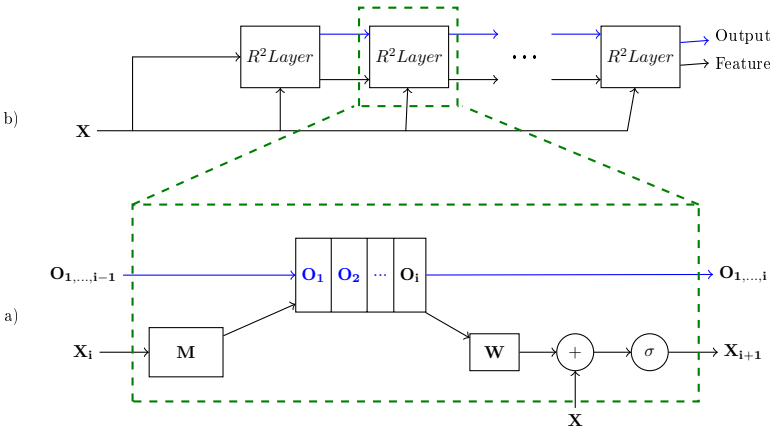


Figure 2. a) Each R^2 layer calculates new transformation based on the output of the previous layer. b) Model consists of repeated R^2 layers.

2. Theoretical analysis

To simplify notation we will use R^{2}M to denote both R^{2}SVM and DrELM models, where M denotes any classifier, so DrELM is denoted as R^{2}ELM. We start the discussion with analysis of R^{2}M behavior on the well–known spiral dataset. Despite its

simplicity model struggles to find correct random projections. We have exhaustively searched possible hypothesis space of a three layered R^{2}SVM and concluded that this dataset cannot be separated by the given model[1]. For simpler two–moon dataset approximately 1% of the runs separate the classes. Interestingly replicating dimensions (by stacking copies) of the datasets drastically improves performance, see Fig. 3.

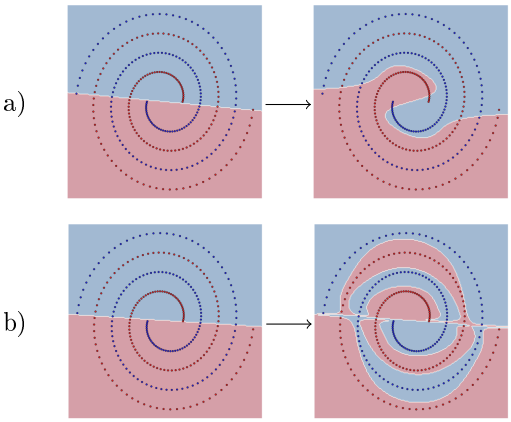


Figure 3. Visualization of 3 layered R^{2}SVM. a) Best run without replicating dimensions. b) Best run with replicated dimensions.

In this analysis we focus on model using only previous layer predictions, as in R^{2}ELM model. It simplifies analysis and does not decrease performance as is both conjectured by R^{2}ELM authors and proven empirically in this paper.

Recall that transformation used in R^{2}M for any linear M can be written as

$$\mathbf{X}_{i+1} = \sigma\left(\mathbf{X} + \alpha \mathbf{O}_i \mathbf{W}_i\right) = \sigma\left(\mathbf{X} + \alpha \sum \langle \mathbf{x}, \mathbf{v}_i \rangle\right) + \mathbf{b} = \sigma\left(\mathbf{X} + T(\mathbf{x})\right),$$

where by $T(\mathbf{x})$ we denote displacement function applied to vector given by $T(\mathbf{x}) = \sum_{i=1}^{K} (\langle \mathbf{x}, \mathbf{v}_i \rangle + \mathbf{b}_i)\mathbf{w}_i$. Note that we can write it as $T(\mathbf{x}) = \mathbf{x}\left(\sum_{i=1}^{K} \mathbf{v}_i \mathbf{w}_i^T\right) + \mathbf{b}$. Consequently

$$T(\mathbf{x}) = \mathbf{x}\left(\sum_{i=1}^{K} \mathbf{v}_i \mathbf{w}_i^T\right) + \mathbf{b} = \mathbf{x}\mathbf{A} + \mathbf{b}.$$

For linearly independent hyperplanes $rank(\mathbf{A}) = K$. Thus this displacement operator is a degenerated affine transformation. Whole layer could be interpreted as a two layer neural network with skip connections or a single layer neural network, see Fig. 4.

[1] We tested all hyperplanes with a rotation step. Original paper reported the experiment using a deeper R^{2}SVM.

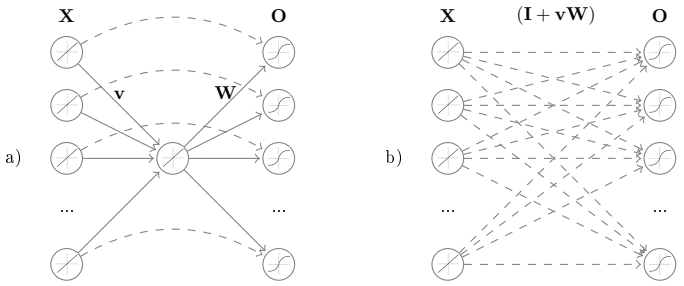


Figure 4. Interpretation of R^2SVM for binary case. a) As 2 layered neural network with skip connections. b) As fully connected single layer neural network ($\mathbf{x} + T(\mathbf{x}) = \mathbf{x}(\mathbf{I} + \mathbf{A})$).

2.1. Classifier importance

One of our claims is that R^2M performance is characterized by the intersection of found hyperplanes. This should be clarified by the following observation. Let $T_{V,W}$ be a transformation defined by hyperplanes, $V_1, \ldots, V_K$ and random vectors $\mathbf{w}_1, \ldots, \mathbf{w}_K$. From now simplified notation is used, where V_i is i^{th} hyperplane and $\mathbf{v}_i$ is its normal.

Observation 1 *Let $V_1, \ldots, V_K$ and $V_1', \ldots, V_K'$ denote hyperplanes then $\bigcap_i V_i = \bigcap_i V_i^{'} \Rightarrow \exists W'$, such that $T_{V',W} = T_{V,W'}$.*

Proof. Let's simplify by skipping bias in the equation for T (which is equivalent to adding constant dimension to x), then

$$T_{V,W} = \sum \mathbf{v}_i \mathbf{w}_i^T.$$

First we show that that $\ker T = \bigcap_i V_i$ if random vectors W are linearly independent. Left inclusion is obvious. Assume that $\mathbf{x} \in \ker T$ and $\mathbf{x} \notin \bigcap_i V_i$, but that implies $\exists \alpha_i : \sum_{i=1}^{K} \alpha_i \mathbf{w}_i = 0$. That would mean that random vectors W are not linearly independent. If the vectors $\mathbf{v}_1, \ldots, \mathbf{v}_K$ are linearly independent then $\dim(\ker T_{V,W}) = N - K$, because $T(\mathbf{x}) = 0 \Leftrightarrow \forall_{i \in \{1,\ldots,K\}}\ \mathbf{x} \perp \mathbf{v}_i$. In general we have $\dim(\ker T_{V,W}) \leq N - K$. We choose basis for V $\{\mathbf{e}_1, \ldots, \mathbf{e}_L\}, L \leq K$. $T_{V',W} = T_{V,W'}$ is equivalent to the set of $N \times L$ linear equations with $N \times K$ parameters:

$$T_{V',W}(\mathbf{e}_i) = T_{V,W'}(\mathbf{e}_i), \quad i = 1, \ldots, L$$

which has an unique solution. □

The observation suggests that multiclass classification is not crucial, as one can for instance rotate all hyperplanes or perform orthogonalization and still be able to find an equivalent model. The observation does not include any results on distribution of W and because of that only hints at this possibility that doing multiclass classification in the middle layers might not be optimal.

2.2. Model interpretation

In this section we derive an interpretation of the model. Let $\mathbf{x} \in \mathbf{X}$ is a data point. Let $T_{V,W}(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{b}$ be affine displacement operator. Clearly $\ker T$ is also an affine subspace, as it is intersection of K hyperplanes (affine subspaces). Denote by $P_{\ker T}(\mathbf{x})$ a projection onto this space, then

$$T'_{U,W}(\mathbf{x}) = T_{V,W}(\mathbf{x} - P_{\ker T}(\mathbf{x})).$$

We can now easily simplify further discussion. Let us first fix origin inside $\ker T$ and then subtract projection as follows: $\mathbf{y} = \mathbf{x} - P_{\ker T_{V,W}}$. We can now assume that space is a vector space, in which of course $\ker T'$ is a linear subspace.

Denote by $\mathbf{u}_1, \ldots, \mathbf{u}_K$ transformed vectors $\mathbf{v}_1, \ldots, \mathbf{v}_K$. Let $B = span(U)$ be subspace spanned by $\mathbf{u}_1, \ldots, \mathbf{u}_K$. We can proof the following observation

Observation 2 *Let $U^\star$ be set of optimally separating hyperplanes in B. Then exists W' such that $T_{U,W} = T_{U^\star,W'}$*

Proof. All vectors from $U^\star$ are linear combinations of vectors from U, $\mathbf{u}_i^\star \in U^\star \Leftrightarrow \mathbf{u}_i = \sum_{j=1}^{L} \alpha_j \mathbf{u}_j$. From this follows that $\bigcap_i U_i = \bigcap_i U_i^\star$. We can find W' using Observation 1. □

Observation 2 suggests that if B subspace is well separable, then model should perform considerably well, given that W' is not degenerated. Finding easily separable B space is unfortunately not equivalent to maximizing multiclass accuracy. One can for instance simply add new hyperplane that is not a good classifier, but divides space in a useful way. If the dataset is almost linearly separable then if V consist of hyperplanes separating classes then B space is likely to be linearly separable; however *the opposite* might not hold. This suggests that the method might not be well suited for datasets for which linear classifiers performs poorly.

It seems also natural that both models should work well when data manifold has smaller dimensionality than space, as it makes more likely that random projections are separating classes. We pose the following hypotheses, that are validated in the empirical section:

Hypothesis 1 *R^2M with L layers performs weakly if at any of the $1, \ldots, L-1$ layers data representation leads to weak linear classifier.*

Hypothesis 2 *R^2M work better if dataset resides on lower dimensional manifold.*

3. Empirical analysis

Tested models were Random Recursive Support Vector Machine (R^2SVM), Deep Representation Extreme Learning Machine (R^2ELM), Support Vector Machine with

Radial Basis Kernel (SVM+RBF), Extreme Learning Machine with Sigmoid Kernel (ELM+SIG) and Linear Support Vector Machine (SVM).

Additionally we added randomized version of R^2SVM called Fixed Prediction R^2SVM ($R^2$1+SVM), where middle layers are using constant prediction equal to one, i.e. $O_i = 1$[2]. By adding this model we evaluate if R^2SVM is not just using additional space besides dataset manifold to increase separability. Interesting results reported in theoretical section inspired us to add modifications of $R^2$1+SVM and R^2SVM, in which models use tripled version of the dataset (R^2_*1+SVM, R^2_*SVM).

3.1. Evaluation

We conducted our experiments on datasets taken from UCI repository [11] and LIBSVM dataset repository [12]. Summary of the datasets is presented in Table 1. To make sure we make a fair comparison we tested extensive grid of parameters for R^2SVM and R^2ELM. The grid tested all combinations (640) of the following parameters:

1. *recurrent*: If set to true model is reusing all the previous predictions as in R^2SVM model.
2. *scale*: If set to true model is performing scaling in every layer.
3. *fit*: If set to true model performs approximate regularization parameter C fitting in each layer.
4. α: Controls size of applied transformation. 8 values ranging from 0.1 to 2.0.
5. *depth*: Depth of the model. 10 values from 1 to 10.

Code used for experimentation is accessible online[3]. Experiments were conducted in Python using scikit-learn and LIBSVM package [12][13]. All of the experiments were done using 5-fold stratified cross validation. Every experiment for R^2SVM models (R^2SVM, R^2_*SVM, ...) is repeated three times (with different seed) and average accuracy of the best performing set of hyperparameters is reported.

3.2. Results

Results are reported in Table 2. We would like to focus on several outcomes. Experiments clearly confirm that R^2M is weaker or comparable to SVM+RBF model in classification accuracy. This does not contradict results reported by authors of the model, as their work did not include extensive testing. Additionally, datasets

[2] Similar results were obtained by several other random versions of R^2SVM, for instance using random hyperplanes.

[3] https://github.com/gmum/r2-learner

name	N	d	K	M	name	N	d	K	M
australian	690	14	2	1	liver	345	6	2	3
bank	1372	4	2	3	pendigits	10992	16	10	9
breast cancer	683	10	2	1	satimage	10870	36	6	6
crashes	540	20	2	1	segment	2310	19	7	7
diabetes	768	8	2	2	sonar	208	60	2	28
fourclass	862	2	2	2	splice	1000	60	2	55
german	1000	24	2	3	svmguide2	391	20	3	15
glass	214	9	6	6	svmguide4	612	10	6	1
heart	270	13	2	3	vehicle	846	18	4	6
indian	583	10	2	3	vowel	990	10	11	8
ionosphere	351	34	2	24	wine	178	4	3	2
iris	150	4	3	2					

Table 1. Experiments datasets summary. N – number of examples, d – number of dimensions, K – number of classes, M – manifold dimension estimation using PCA.

tested in this work are characterized by rather low dimensionality and high number of classes, which differs from set tested in the original papers. It is also consistent with our theoretical understanding of the model.

Both hypotheses are confirmed by the empirical results. Hypothesis 1 specifically stated that $\mathrm{R}^2\mathrm{M}$ performance is correlated with some measure of weakness of linear classifier. It is clearly visible for highly linearly non–separable datasets, e.g. *glass* or *vowel*. We did a heuristic hypothesis test by measuring correlation of $\frac{\mathrm{acc}_{SVM+RBF}}{\mathrm{acc}_{R^2SVM}}$ and $\frac{\mathrm{acc}_{SVM}-\mathrm{acc}_{SVM+RBF}}{\mathrm{acc}_{SVM+RBF}}$, where acc is the best accuracy on given dataset. We obtained approximately 0.9 Spearman's correlation coefficient, which supports the hypothesis.

Second hypothesis can be validated similarly, we calculate correlation between acc_{R^2SVM} and difference between dataset dimensionality and manifold estimation. We report Spearman's correlation coefficient equal to 0.7, which also confirms correlation, however weaker.

Additional result is the fact that tripling dataset dimensionality by replicating data improves accuracy. It is equivalent to increasing hidden layer sizes in neural network interpretation of $\mathrm{R}^2\mathrm{M}$ model. In that case R^2_*1+SVM model, which is not fitting linear models in the middle layers, is performing comparatively to R^2SVM.

dataset	R^2SVM		R^2ELM		ELM+SIG		SVM+RBF		SVM		$R^2$1+SVM		R^2_*SVM		*SVM+RBF		R^2_*1+SVM	
australian	0.87	±0.01	0.87	±0.01	**0.88**	± 0.02	0.87	±0.01	0.86	±0.01	0.86	±0.01	0.87	±0.01	0.86	±0.01	0.86	±0.01
bank	1.00	±0.01	0.97	±0.01	**1.00**	± 0.00	**1.00**	± 0.00	0.99	±0.01	1.00	±0.01	1.00	±0.01	**1.00**	± 0.00	**1.00**	± 0.00
breast_cancer	0.97	±0.01	0.97	±0.01	0.97	±0.01	0.97	±0.01	0.97	±0.01	0.97	±0.01	0.97	±0.01	0.97	±0.01	**0.97**	± 0.01
crashes	0.95	±0.01	0.95	±0.01	0.93	±0.01	**0.96**	± 0.02	0.96	±0.01	0.95	±0.01	0.95	±0.01	0.96	±0.01	0.95	±0.01
diabetes	0.78	±0.01	0.77	±0.01	**0.78**	± 0.03	0.78	±0.01	0.78	±0.01	0.76	±0.01	0.76	±0.01	0.78	±0.01	0.77	±0.01
fourclass	0.79	±0.01	0.78	±0.01	0.99	±0.01	**1.00**	± 0.00	0.77	±0.01	0.74	±0.01	0.81	±0.01	**1.00**	± 0.00	0.77	±0.01
german	0.77	±0.01	**0.77**	± 0.00	0.75	±0.01	0.76	±0.01	0.76	±0.01	0.73	±0.01	0.73	±0.01	0.76	±0.01	0.72	±0.01
glass	0.64	±0.01	0.64	±0.01	0.71	±0.01	**0.73**	± 0.05	0.62	±0.01	0.59	±0.01	0.69	±0.01	0.72	±0.01	0.65	±0.01
heart	0.84	±0.01	**0.85**	± 0.00	0.83	±0.01	0.85	±0.01	0.84	±0.01	0.85	±0.01	0.85	±0.01	0.85	±0.01	0.85	±0.01
indian	0.73	±0.01	0.72	±0.01	**0.73**	± 0.01	0.72	±0.01	0.72	±0.01	0.71	±0.01	0.71	±0.01	0.72	±0.01	0.71	±0.01
ionosphere	0.89	±0.01	0.92	±0.01	0.91	±0.01	**0.96**	± 0.02	0.89	±0.01	0.91	±0.01	0.89	±0.01	0.93	±0.01	0.92	±0.01
iris	0.97	±0.01	0.96	±0.01	**0.98**	± 0.02	**0.98**	± 0.03	0.95	±0.01	0.97	±0.01	**0.98**	± 0.01	0.97	±0.01	0.98	±0.01
liver	0.70	±0.01	0.69	±0.01	0.74	±0.01	**0.75**	± 0.04	0.71	±0.01	0.66	±0.01	0.70	±0.01	0.73	±0.01	0.71	±0.01
pendigits	0.94	±0.01	0.87	±0.01	0.97	±0.01	**1.00**	± 0.00	0.93	±0.01	0.94	±0.01	0.99	±0.01	1.00	±0.01	0.97	±0.01
satimage	0.89	±0.01	0.88	±0.01	0.89	±0.01	0.93	±0.01	0.82	±0.01	0.87	±0.01	0.92	±0.01	**0.97**	± 0.00	0.90	±0.01
segment	0.96	±0.01	0.94	±0.01	0.93	±0.01	0.97	±0.01	0.93	±0.01	0.93	±0.01	**0.97**	± 0.01	0.97	±0.01	0.94	±0.01
sonar	0.75	±0.01	0.77	±0.01	0.83	±0.01	**0.89**	± 0.05	0.76	±0.01	0.81	±0.01	0.76	±0.01	0.82	±0.01	0.84	±0.01
splice	0.81	±0.01	0.82	±0.01	0.76	±0.01	0.88	±0.01	0.80	±0.01	0.81	±0.01	0.81	±0.01	**0.88**	± 0.02	0.88	±0.01
svmguide2	0.83	±0.01	0.84	±0.01	0.82	±0.01	0.85	±0.01	0.84	±0.01	0.82	±0.01	0.83	±0.01	**0.85**	± 0.02	0.84	±0.01
svmguide4	0.85	±0.01	0.80	±0.01	0.76	±0.01	0.87	±0.01	0.81	±0.01	0.73	±0.01	**0.90**	± 0.01	0.87	±0.01	0.82	±0.01
vehicle	0.81	±0.01	0.80	±0.01	0.82	±0.01	**0.86**	± 0.01	0.78	±0.01	0.78	±0.01	0.82	±0.01	0.85	±0.01	0.82	±0.01
vowel	0.62	±0.01	0.54	±0.01	0.83	±0.01	0.99	±0.01	0.47	±0.01	0.49	±0.01	0.87	±0.01	**1.00**	± 0.00	0.72	±0.01
wine	0.83	±0.01	0.83	±0.01	**0.87**	± 0.04	0.86	±0.01	0.83	±0.01	0.84	±0.01	0.84	±0.01	0.84	±0.01	0.86	±0.01

Table 2. Main results of the experiments. Both accuracy and standard deviation is reported.

4. Summary

In this paper we analyzed rigorously R^2SVM and R^2ELM models both from theoretical and empirical point of view. Basing on our theoretical observations we have proven empirically two hypotheses, stating that R^2SVM and R^2ELM perform weakly if any of the layer data representation leads to weak linear classifier and that R^2SVM and R^2ELM rely on dataset residing on a much lower dimensional manifold. In summary it suggests that R^2SVM and R^2ELM might not be always learning useful representations, which is further confirmed by weak results in comparison with Support Vector Machine with RBF kernel. Future research should focus on making sure R^2SVM is more broadly applicable.

Acknowledgements

We would like to thank Wojciech Czarnecki, member of our research group, for support and guidance that he continuously expressed during our work on this paper.

5. References

[1] Bengio, Y., *Learning deep architectures for AI.* Foundations and Trends in Machine Learning, 2009, **2**.

[2] Podolak, I.T., Roman, A., *Theoretical foundations and experimental results for a hierarchical classifier with overlapping clusters.* Computational Intelligence, 2013, **29**(2), pp. 357–388.

[3] Czarnecki, W.M., Tabor, J., *Extreme entropy machines: Robust information theoretic classification.* arXiv preprint arXiv:1501.05279, 2015.

[4] Huang, G.B., Zhu, Q.Y., Siew, C.K., *Extreme learning machine: Theory and applications.* Neurocomputing, 2006, **70**(1âĂŞ3), pp. 489 – 501.

[5] Tissera, M., McDonnell, M., Deep extreme learning machines for classification. In: *Proceedings of ELM-2014 Volume 1.* vol. 3 of *Proceedings in Adaptation, Learning and Optimization.* Springer International Publishing 2015 pp. 345–354.

[6] Vinyals, O., Jia, Y., Deng, L., Darrell, T., Learning with recursive perceptual representations. In: *Advances in Neural Information Processing Systems 25.* Curran Associates, Inc. 2012 pp. 2825–2833.

[7] Yu, W., Zhuang, F., He, Q., Shi, Z., *Learning deep representations via extreme learning machines.* Neurocomputing, 2015, **149**, pp. 308 – 315.

[8] Wolpert, D.H., *Stacked generalization.* Neural Networks, 1992, **5**, pp. 241–259.

[9] Cortes, C., Vapnik, V., *Support-vector networks.* Machine learning, 1995, **20**(3), pp. 273–297.

[10] Tang, Y., Deep learning using linear support vector machines. In: *In ICML*, 2013.

[11] Lichman, M., *UCI machine learning repository*, 2013.

[12] Chang, C.C., Lin, C.J., *LIBSVM: A library for support vector machines.* ACM Transactions on Intelligent Systems and Technology, 2011, **2**, pp. 27:1–27:27.

[13] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E., *Scikit-learn: Machine learning in Python.* Journal of Machine Learning Research, 2011, **12**, pp. 2825–2830.

Schedae Informaticae Vol. 25 (2016): 49–59
doi: 10.4467/20838476SI.16.004.6185

tfml 2017
theoretical foundations
of machine learning, Kraków

On Loss Functions for Deep Neural Networks in Classification

Katarzyna Janocha[1], Wojciech Marian Czarnecki[2,1]
[1]Faculty of Mathematics and Computer Science,
Jagiellonian University, Krakow, Poland
[2]DeepMind, London, UK
e-mail: *kasiajanocha@gmail.com, lejlot@google.com*

Abstract. Deep neural networks are currently among the most commonly used classifiers. Despite easily achieving very good performance, one of the best selling points of these models is their modular design – one can conveniently adapt their architecture to specific needs, change connectivity patterns, attach specialised layers, experiment with a large amount of activation functions, normalisation schemes and many others. While one can find impressively wide spread of various configurations of almost every aspect of the deep nets, one element is, in authors' opinion, underrepresented – while solving classification problems, vast majority of papers and applications simply use log loss. In this paper we try to investigate how particular choices of loss functions affect deep models and their learning dynamics, as well as resulting classifiers robustness to various effects. We perform experiments on classical datasets, as well as provide some additional, theoretical insights into the problem. In particular we show that $\mathcal{L}_1$ and $\mathcal{L}_2$ losses are, quite surprisingly, justified classification objectives for deep nets, by providing probabilistic interpretation in terms of expected misclassification. We also introduce two losses which are not typically used as deep nets objectives and show that they are viable alternatives to the existing ones.

Keywords: loss function, deep learning, classification theory.

Received: 11 December 2016 / Accepted: 30 December 2016

1. Introduction

For the last few years the Deep Learning (DL) research has been rapidly developing. It evolved from tricky pretraining routines [1] to a highly modular, customisable framework for building machine learning systems for various problems, spanning from image recognition [2], voice recognition and synthesis [3] to complex AI systems [4]. One of the biggest advantages of DL is enormous flexibility in designing each part of the architecture, resulting in numerous ways of putting priors over data inside the model itself [1], finding the most efficient activation functions [5] or learning algorithms [6]. However, to authors' best knowledge, most of the community still keeps one element nearly completely fixed – when it comes to classification, we use log loss (applied to softmax activation of the output of the network). In this paper we try to address this issue by performing both theoretical and empirical analysis of effects various loss functions have on the training of deep nets.

It is worth noting that Tang et al. [7] showed that well fitted hinge loss can outperform log loss based networks in typical classification tasks. Lee et al. [8] used squared hinge loss for classification tasks, achieving very good results. From slightly more theoretical perspective Choromanska et al. [9] also considered $\mathcal{L}_1$ loss as a deep net objective. However, these works seem to be exceptions, appear in complete separation from one another, and usually do not focus on any effect of the loss function but the final performance. Our goal is to show these losses in a wider context, comparing one another under various criteria and provide insights into when – and why – one should use them.

Table 1. List of losses analysed in this paper. $\mathbf{y}$ is true label as one-hot encoding, $\hat{\mathbf{y}}$ is true label as +1/-1 encoding, $\mathbf{o}$ is the output of the last layer of the network, $\cdot^{(j)}$ denotes jth dimension of a given vector, and $\sigma(\cdot)$ denotes probability estimate.

symbol	name	equation
$\mathcal{L}_1$	L_1 loss	$\|\mathbf{y} - \mathbf{o}\|_1$
$\mathcal{L}_2$	L_2 loss	$\|\mathbf{y} - \mathbf{o}\|_2^2$
$\mathcal{L}_1 \circ \sigma$	expectation loss	$\|\mathbf{y} - \sigma(\mathbf{o})\|_1$
$\mathcal{L}_2 \circ \sigma$	regularised expectation loss[1]	$\|\mathbf{y} - \sigma(\mathbf{o})\|_2^2$
$\mathcal{L}_\infty \circ \sigma$	Chebyshev loss	$\max_j \lvert\sigma(\mathbf{o})^{(j)} - \mathbf{y}^{(j)}\rvert$
hinge	hinge [7] (margin) loss	$\sum_j \max(0, \frac{1}{2} - \hat{\mathbf{y}}^{(j)}\mathbf{o}^{(j)})$
hinge^2	squared hinge (margin) loss	$\sum_j \max(0, \frac{1}{2} - \hat{\mathbf{y}}^{(j)}\mathbf{o}^{(j)})^2$
hinge^3	cubed hinge (margin) loss	$\sum_j \max(0, \frac{1}{2} - \hat{\mathbf{y}}^{(j)}\mathbf{o}^{(j)})^3$
log	log (cross entropy) loss	$-\sum_j \mathbf{y}^{(j)} \log \sigma(\mathbf{o})^{(j)}$
$\log^2$	squared log loss	$-\sum_j [\mathbf{y}^{(j)} \log \sigma(\mathbf{o})^{(j)}]^2$
tan	Tanimoto loss	$\frac{-\sum_j \sigma(\mathbf{o})^{(j)}\mathbf{y}^{(j)}}{\|\sigma(\mathbf{o})\|_2^2 + \|\mathbf{y}\|_2^2 - \sum_j \sigma(\mathbf{o})^{(j)}\mathbf{y}^{(j)}}$
D_{CS}	Cauchy-Schwarz Divergence [10]	$-\log \frac{\sum_j \sigma(\mathbf{o})^{(j)}\mathbf{y}^{(j)}}{\|\sigma(\mathbf{o})\|_2 \|\mathbf{y}\|_2}$

[1] See Proposition 1

This work focuses on 12 loss functions, described in Table 1. Most of them appear in deep learning (or more generally – machine learning) literature, however some in slightly different context than a classification loss. In the following section we present new insights into theoretical properties of a couple of these losses and then provide experimental evaluation of resulting models' properties, including the effect on speed of learning, final performance, input data and label noise robustness as well as convergence for simple dataset under limited resources regime.

2. Theory

Let us begin with showing interesting properties of $\mathcal{L}_p$ functions, typically considered as purely regressive losses, which should not be used in classification. $\mathcal{L}_1$ is often used as an auxiliary loss in deep nets to ensure sparseness of representations. Similarly, $\mathcal{L}_2$ is sometimes (however nowadays quite rarely) applied to weights in order to prevent them from growing to infinity. In this section we show that – despite their regression roots – they still have reasonable probabilistic interpretation for classification and can be used as a main classification objective.

We use the following notation: $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N \subset \mathbb{R}^d \times \{0,1\}^K$ is a training set, an iid sample from unknown $P(\mathbf{x}, \mathbf{y})$ and σ denotes a function producing probability estimates (usually sigmoid or softmax).

Proposition 1. *$\mathcal{L}_1$ loss applied to the probability estimates $\hat{p}(\mathbf{y}|\mathbf{x})$ leads to minimisation of expected misclassification probability (as opposed to maximisation of fully correct labelling given by the log loss). Similarly $\mathcal{L}_2$ minimises the same factor, but regularised with a half of expected squared L_2 norm of the predictions probability estimates.*

Proof. In K-class classification dependent variables are vectors $\mathbf{y}_i \in \{0,1\}^K$ with $\mathrm{L}_1(\mathbf{y}_i) = 1$, thus using notation $\mathbf{p}_i = \hat{p}(\mathbf{y}|\mathbf{x}_i)$

$$\begin{aligned}\mathcal{L}_1 &= \tfrac{1}{N}\sum_i \sum_j |\mathbf{p}_i^{(j)} - \mathbf{y}_i^{(j)}| = \tfrac{1}{N}\sum_i \left[\sum_j \mathbf{y}_i^{(j)}(1-\mathbf{p}_i^{(j)}) + (1-\mathbf{y}_i^{(j)})\mathbf{p}_i^{(j)}\right] \\ &= \tfrac{1}{N}\sum_i \left[\sum_j \mathbf{y}_i^{(j)} - 2\sum_j \mathbf{y}_i^{(j)}\mathbf{p}_i^{(j)} + \sum_j \mathbf{p}_i^{(j)}\right] = 2 - 2\tfrac{1}{N}\sum_i \left[\sum_j \mathbf{y}_i^{(j)}\mathbf{p}_i^{(j)}\right].\end{aligned}$$

Consequently if we sample label according to $\mathbf{p}_i$ then probability that it actually matches one hot encoded label in $\mathbf{y}_i$ equals $P(\hat{l} = l|\hat{l} \sim \mathbf{p}_i, l \sim \mathbf{y}_i) = \sum_j \mathbf{y}_i^{(j)}\mathbf{p}_i^{(j)}$, and consequently

$$\mathcal{L}_1 = 2 - 2\tfrac{1}{N}\sum_i \left[\sum_j \mathbf{y}_i^{(j)}\mathbf{p}_i^{(j)}\right] \approx -2\mathbb{E}_{P(\mathbf{x},\mathbf{y})}\left[P(\hat{l} = l|\hat{l} \sim \mathbf{p}_i, l \sim \mathbf{y}_i)\right] + \text{const.}$$

Analogously for $\mathcal{L}_2$,

$$\begin{aligned}\mathcal{L}_2 &= -2\tfrac{1}{N}\sum_i \left[\sum_j \mathbf{y}_i^{(j)}\mathbf{p}_i^{(j)}\right] + \tfrac{1}{N}\sum_i \mathrm{L}_2(\mathbf{y}_i)^2 + \tfrac{1}{N}\sum_i \mathrm{L}_2(\mathbf{p}_i)^2 \\ &\approx -2\mathbb{E}_{P(\mathbf{x},\mathbf{y})}\left[P(\hat{l} = l|\hat{l} \sim \mathbf{p}_i, l \sim \mathbf{y}_i)\right] + \mathbb{E}_{P(\mathbf{x},\mathbf{y})}[\mathrm{L}_2(\mathbf{p}_i)^2] + \text{const.}\end{aligned}$$

□

For this reason we refer to these losses as *expectation loss* and *regularised expectation loss* respectively. One could expect that this should lead to higher robustness to the outliers/noise, as we try to maximise the expected probability of good classification as opposed to the probability of completely correct labelling (which log loss does). Indeed, as we show in the experimental section – this property is true for all losses sharing connection with *expectation losses*.

So why is using these two loss functions unpopular? Is there anything fundamentally wrong with this formulation from the mathematical perspective? While the following observation is not definitive, it shows an insight into what might be the issue causing slow convergence of such methods.

Proposition 2. *$\mathcal{L}_1$, $\mathcal{L}_2$ losses applied to probabilities estimates coming from sigmoid (or softmax) have non-monotonic partial derivatives wrt. to the output of the final layer (and the loss is not convex nor concave wrt. to last layer weights). Furthermore, they vanish in both infinities, which slows down learning of heavily misclassified examples.*

Proof. Let us denote sigmoid activation as $\sigma(x) = (1 + e^{-x})^{-1}$ and, without loss of generality, compute partial derivative of $\mathcal{L}_1$ when network is presented with x_p with positive label. Let o_p denote the output activation for this sample.

$$\frac{\partial(\mathcal{L}_1 \circ \sigma)}{\partial o}(o_p) = \frac{\partial}{\partial o}\left(|1 - (1 + e^{-o})^{-1}|\right)(o_p) = -\frac{e^{-o_p}}{(e^{-o_p} + 1)^2}$$
$$\lim_{o \to -\infty} -\frac{e^{-o}}{(e^{-o} + 1)^2} = 0 = \lim_{o \to \infty} -\frac{e^{-o}}{(e^{-o} + 1)^2},$$

while at the same time $-\frac{e^0}{(e^0+1)^2} = -\frac{1}{4} < 0$, completing the proof of both non-monotonicity as well as the fact it vanishes when point is heavily misclassified. Lack of convexity comes from the same argument since second derivative wrt. to any weight in the final layer of the model changes sign (as it is equivalent to first derivative being non-monotonic). This comes directly from the above computations and the fact that $o_p = \langle \mathbf{w}, \mathbf{h}_p \rangle + b$ for some internal activation $\mathbf{h}_p$, layer weights $\mathbf{w}$ and layer bias b. In a natural way this is true even if we do not have any hidden layers (model is linear). Proofs for $\mathcal{L}_2$ and softmax are completely analogous. □

Given this negative result, it seems natural to ask whether a similar property can be proven to show which loss functions should lead to *fast* convergence. It seems like the answer is again positive, however based on the well known deep learning hypothesis that deep models learn well when dealing with piece-wise linear functions. An interesting phenomenon in classification based on neural networks is that even in a deep linear model or rectifier network the top layer is often non-linear, as it uses softmax or sigmoid activation to produce probability estimates. Once this is introduced, also the partial derivatives stop being piece-wise linear. We believe that one can achieve faster, better convergence when we ensure that architecture together with loss function, produces a piecewise linear partial derivatives (but not constant) wrt. to final layer activations, especially while using first order optimisation methods.

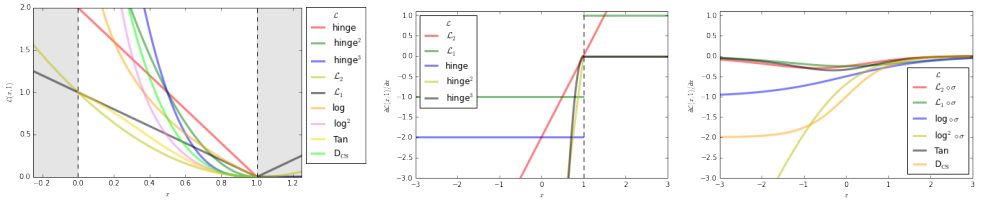


Figure 1. Left: Visualisation of analysed losses as functions of activation on positive sample. Middle: Visualisation of partial derivatives wrt. to output neuron for losses based on linear output. Right: Visualisation of partial derivatives wrt. to output neuron for losses based on probability estimates.

This property is true only for $\mathcal{L}_2$ loss and squared hinge loss (see Figure 1) among all considered ones in this paper.

Finally we show relation between Cauchy-Schwarz Divergence loss and the log loss, justifying its introduction as an objective for neural nets.

Proposition 3. *Cauchy-Schwarz Divergence loss is equivalent to cross entropy loss regularised with half of expected Renyi's quadratic entropy of the predictions.*

Proof. Using the fact that $\forall_i \exists!_j : \mathbf{y}_i^{(j)} = 1$ we get that $\log \sum_j \mathbf{p}_i^{(j)} \mathbf{y}_i^{(j)} = \sum_j \mathbf{y}_i^{(j)} \log \mathbf{p}_i^{(j)}$ as well as $\|\mathbf{y}_i\|_2 = 1$, consequently

$$
\begin{aligned}
D_{\mathrm{CS}} &= -\tfrac{1}{N} \sum_i \log \frac{\sum_j \mathbf{p}_i^{(j)} \mathbf{y}_i^{(j)}}{\|\mathbf{p}_i\|_2 \|\mathbf{y}_i\|_2} = -\tfrac{1}{N} \sum_i \log \sum_j \mathbf{p}_i^{(j)} \mathbf{y}_i^{(j)} + \tfrac{1}{N} \sum_i \log \|\mathbf{p}_i\|_2 \|\mathbf{y}_i\|_2 \\
&= -\tfrac{1}{N} \sum_i \sum_j \mathbf{y}_i^{(j)} \log \mathbf{p}_i^{(j)} + \tfrac{1}{2N} \sum_i \log \|\mathbf{p}_i\|_2^2 \approx \mathcal{L}_{\log} + \tfrac{1}{2} \mathbb{E}_{P(\mathbf{x},\mathbf{y})}[H_2(\mathbf{p}_i)]
\end{aligned}
$$

□

3. Experiments

We begin the experimental section with two simple 2D toy datasets. The first one is checkerboard – 4 class classification problem where [-1,1] square is divided into 64 small squares with cyclic class assignment. The second one, spiral, is a 4 class generalisation of the well known 2 spirals dataset. Both datasets have 800 training and 800 testing samples. We train rectifier neural network having from 0 to 5 hidden layers with 200 units in each of them. Training is performed using Adam [6] with learning rate of 0.00003 for 60,000 iterations with batch size of 50 samples. In these simple problems one can distinguish (Figure 2) two groups of losses – one able to fit to our very dense, low-dimensional data and one struggling to reduce error to 0. The second group consists of $\mathcal{L}_1$, Chebyshev, Tanimoto and expectation loss. This division becomes clear once we build a relatively deep model (5 hidden layers), while for shallow ones this distinction is not very clear (3 hidden layers) or is even completely lost (1

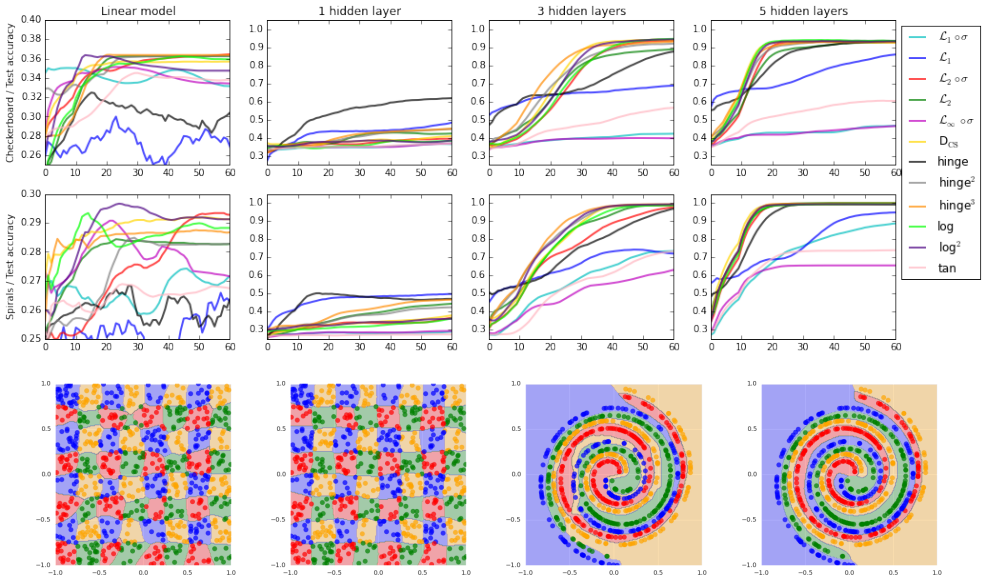


Figure 2. Top row: Learning curves for toy datasets. Bottom row: examples of decision boundaries, from left: $\mathcal{L}_1$ loss, log loss, $\mathcal{L}_1 \circ \sigma$ loss, hinge^2 loss.

hidden layer or linear model). To further confirm the lack of ability to easily overfit we also ran an experiment in which we tried to fit 800 samples from uniform distribution over $[-1, 1]$ with randomly assigned 4 labels and achieved analogous partitioning.

During following, real data-based experiments, we focus on further investigation of loss functions properties emerging after application to deep models, as well as characteristics of the created models. In particular, we show that lack of ability to reduce training error to 0 is often correlated with robustness to various types of noise (despite not underfitting the data).

Let us now proceed with one of the most common datasets used in deep learning community – MNIST [11]. We train network consisting from 0 to 5 hidden layers, each followed by ReLU activation function and dropout [12] with 50% probability. Each hidden layer consists of 512 neurons, and whole model is trained using Adam [6] with learning rate of 0.00003 for 100,000 iterations using batch size of 100 samples. There are few interesting findings, visible on Figure 3. First, results obtained for a linear model (lack of hidden layers) are qualitatively different from all the remaining ones. For example, using regularised expectation loss leads to the strongest model in terms of both training accuracy and generalisation capabilities, while the same loss function is far from being the best one once we introduce non-linearities. This shows two important things: first – observations and conclusions drawn from linear models do not seem to transfer to deep nets, and second – there seems to be an interesting co-dependence between learning dynamics coming from training rectifier nets and loss functions used. As a side note, 93% testing accuracy, obtained by $\mathcal{L}_2 \circ \sigma$ and D_{CS}, is a very strong result on MNIST using linear model without any data augmentation or model regularisation.

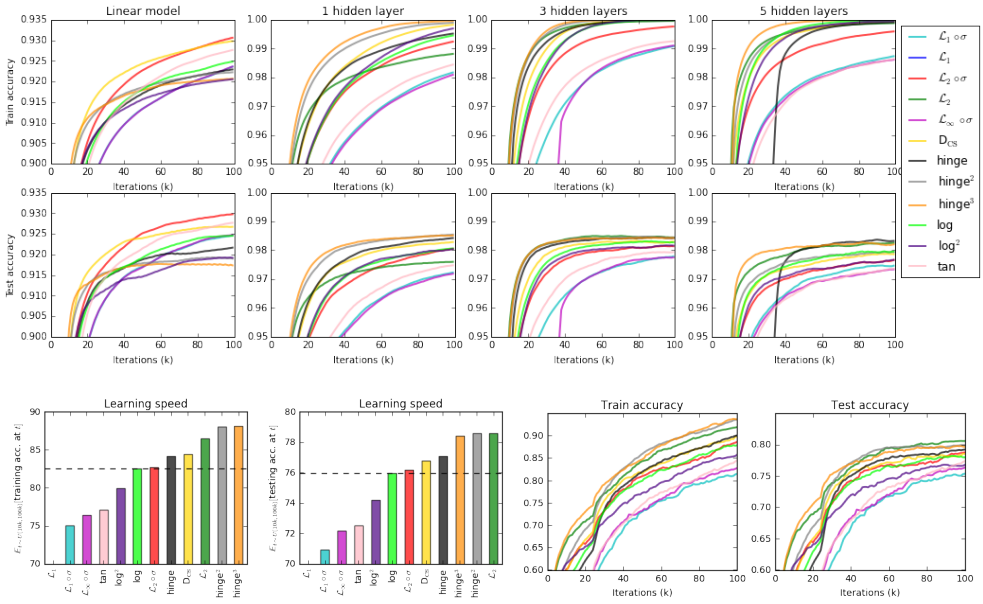


Figure 3. Top two rows: learning curves for MNIST dataset. Bottom row: (left) speed of learning expressed as expected training/testing accuracy when we sample iteration uniformly between 10k and 100k; (right) learning curves for CIFAR10 dataset.

Second interesting observation regards the speed of learning. It appears that (apart from linear models) hinge^2 and hinge^3 losses are consistently the fastest in training, and once we have enough hidden layers (basically more than 1) also $\mathcal{L}_2$. This matches our theoretical analysis of these losses in the previous section. At the same time both expectation losses are much slower to train, which we believe to be a result of their vanishing partial derivatives in heavily misclassified points (Proposition 2). It is important to notice that while higher order hinge losses (especially 2$^{\text{nd}}$) actually help in terms of both speed and final performance, the same property does not hold for higher order log losses. One possible explanation is that taking a square of log loss only reduces model's certainty in classification (since any number between 0 and 1 taken to 2$^{\text{nd}}$ power decreases), while for hinge losses the metric used for penalising margin-outliers is changed, and both L_1 metric (leading to hinge) as well as any other L_p norm (leading to hinge^p) make perfect sense.

Third remark is that pure $\mathcal{L}_1$ does not learn at all (ending up with 20% accuracy) due to causing serious "jumps" in the model because of its partial derivatives wrt. to net output always being either -1 or 1. Consequently, even after classifying a point correctly, we are still heavily penalised for it, while with losses like $\mathcal{L}_2$ the closer we are to the correct classification - the smaller the penalty is.

Finally, in terms of generalisation capabilities margin-based losses seem to outperform the remaining families. One could argue that this is just a result of lack of regularisation in the rest of the losses, however we underline that all the analysed networks use strong dropout to counter the overfitting problem, and that typical L_1 or L_2 regularisation penalties do not work well in deep networks.

For CIFAR10 dataset we used a simple convnet, consisting of 3 layers of convolutions, each of size 5x5 and 64 filters, with ReLU activation functions, batch-normalisation and pooling operations in between them (max pooling after first layer and then two average poolings, all 3x3 with stride 2), followed by a single fully connected hidden layer with 128 ReLU neurons, and final softmax layer with 10 neurons. As one can see in Figure 3, despite completely different architecture than before, we obtain very similar results – higher order margin losses lead to faster training and significantly stronger models. Quite surprisingly – $\mathcal{L}_2$ loss also exhibits similar property. Expectation losses again learn much slower (with the regularised one – training at the level of log loss and unregularised – significantly worse). We would like to underline that this is a very simple architecture, far from the state-of-the art models for CIFAR10, however we wish to avoid using architectures which are heavily overfitted to the log loss. Furthermore, the aim of this paper is not to provide any state-of-the-art models, but rather to characterise effects of loss functions on deep networks.

As the final interesting result in these experiments, we notice that Cauchy-Schwarz Divergence as the optimisation criterion seems to be a consistently better choice than log loss. It performs equally well or better on both MNIST and CIFAR10 in terms of both learning speed and the final performance. At the same time this information theoretic measure is very rarely used in DL community, and rather exploited in shallow learning (for both classification [10] and clustering [13]).

Now we focus on the impact these losses have on noise robustness of the deep nets. We start by performing the following experiment on previously trained MNIST classifiers: we add noise sampled from $\mathcal{N}(0, \epsilon\mathbf{I})$ to each $\mathbf{x}_i$ and observe how quickly (in terms of growing ϵ) network's training accuracy drops (Figure 4). The first crucial observation is that both expectation losses perform very well in terms of input noise robustness. We believe that this is a consequence of what Proposition 1 showed about their probabilistic interpretation – that they lead to minimisation of the expected misclassification, which is less biased towards outliers than log loss (or other losses that focus on maximisation of probability of correct labelling of all samples at the same time). For log loss a single heavily misclassified point has an enormous impact on the overall error surface, while for these two losses – it is minor. Secondly, margin based losses also perform well on this test, usually slightly worse than the expectation losses, but still better than log loss. This shows that despite no longer maximising the misclassification margin while being used in deep nets – they still share some characteristics with their linear origins (SVM). In another, similar experiment, we focus on the generalisation capabilities of the networks trained with increasing amount of label noise in the training set (Figure 4) and obtain analogous results, showing that robustness to the noise of expectation and margin losses is high for both input and label noise for deep nets, while again – slightly different results are obtained for linear models, where log loss is more robust than the margin-based ones. What is even more interesting, a completely non-standard loss function – *Tanimoto loss* – performs extremely well on this task. We believe that its exact analysis is one of the important future research directions.

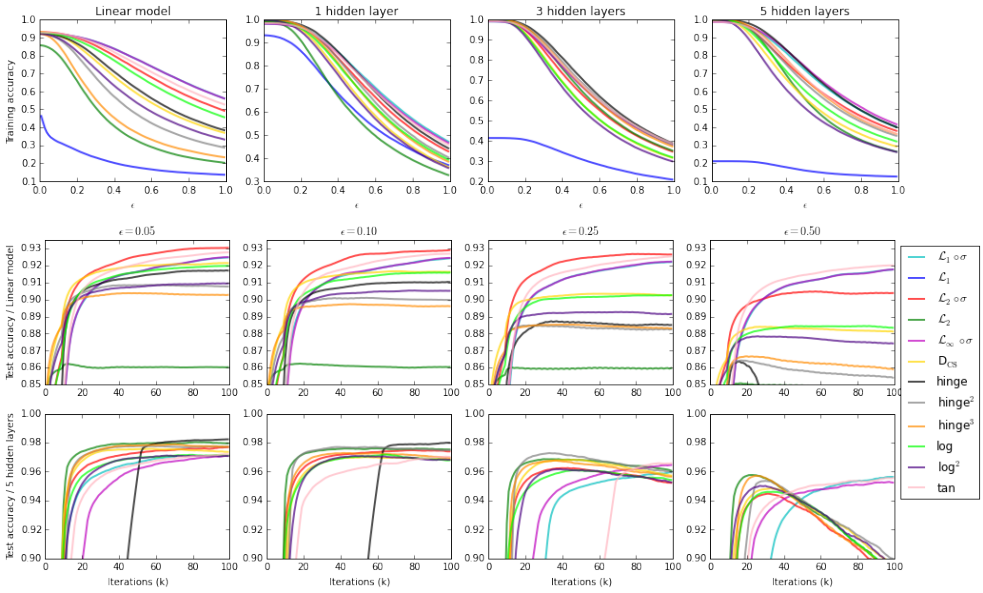


Figure 4. Top row: Training accuracy curves for the MNIST trained models, when presented with training examples with added noise from $\mathcal{N}(0, \epsilon\mathbf{I})$, plotted as a function of ϵ. Middle and bottom rows: Testing accuracy curves for the MNSIT experiment with ϵ of training labels changed, plotted as a function of training iteration. If $\mathcal{L}_1 \circ \sigma$ is not visible, it is almost perfectly overlapped by $\mathcal{L}_\infty \circ \sigma$.

4. Conclusions

This paper provides basic analysis of effects the choice of the classification loss function has on deep neural networks training as well as their final characteristics. We believe the obtained results will lead to a wider adoption of various losses in DL work – where up till now log loss is unquestionable favourite.

In the theoretical section we show that, surprisingly, losses which are believed to be applicable mostly to regression, have a valid probabilistic interpretation when applied to deep network-based classifiers. We also provide theoretical arguments explaining why using them might lead to slower training, which might be one of the reasons DL practitioners have not yet exploited this path. Our experiments lead to two crucial conclusions. First, that intuitions drawn from linear models rarely transfer to highly-nonlinear deep networks. Second, that depending on the application of the deep model – losses other than log loss are preferable. In particular, for purely accuracy focused research, squared hinge loss seems to be a better choice at it converges faster as well as provides better performance. It is also more robust to noise in the training set labelling and slightly more robust to noise in the input space. However, if one works with highly noised dataset (both input and output spaces) – the expectation losses described in detail in this paper – seem to be the best choice, both from theoretical

and empirical perspective.

At the same time this topic is far from being exhausted, with a large amount of possible paths to follow and questions to be answered. In particular, non-classical loss functions such as Tanimoto loss and Cauchy-Schwarz Divergence are worth further investigation.

5. References

[1] Larochelle, H., Bengio, Y., Louradour, J., Lamblin, P., *Exploring strategies for training deep neural networks.* Journal of Machine Learning Research, 2009, **10**(Jan), pp. 1–40.

[2] Krizhevsky, A., Sutskever, I., Hinton, G.E., Imagenet classification with deep convolutional neural networks. In: *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[3] Oord, A.v.d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., Kavukcuoglu, K., *Wavenet: A generative model for raw audio.* arXiv preprint arXiv:1609.03499, 2016.

[4] Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al., *Mastering the game of go with deep neural networks and tree search.* Nature, 2016, **529**(7587), pp. 484–489.

[5] Clevert, D.A., Unterthiner, T., Hochreiter, S., *Fast and accurate deep network learning by exponential linear units (elus).* arXiv preprint arXiv:1511.07289, 2015.

[6] Kingma, D., Ba, J., *Adam: A method for stochastic optimization.* arXiv preprint arXiv:1412.6980, 2014.

[7] Tang, Y., *Deep learning using linear support vector machines.* arXiv preprint arXiv:1306.0239, 2013.

[8] Lee, C.Y., Xie, S., Gallagher, P., Zhang, Z., Tu, Z., Deeply-supervised nets. In: *AISTATS.* vol. 2., 2015, pp. 6.

[9] Choromanska, A., Henaff, M., Mathieu, M., Arous, G.B., LeCun, Y., The loss surfaces of multilayer networks. In: *AISTATS*, 2015.

[10] Czarnecki, W.M., Jozefowicz, R., Tabor, J., Maximum entropy linear manifold for learning discriminative low-dimensional representation. In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, 2015, pp. 52–67.

[11] LeCun, Y., Cortes, C., Burges, C.J., *The mnist database of handwritten digits*, 1998.

[12] Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R., *Dropout: a simple way to prevent neural networks from overfitting.* Journal of Machine Learning Research, 2014, **15**(1), pp. 1929–1958.

[13] Principe, J.C., Xu, D., Fisher, J., *Information theoretic learning.* Unsupervised adaptive filtering, 2000, **1**, pp. 265–319.

Schedae Informaticae Vol. 25 (2016): 61–72
doi: 10.4467/20838476SI.16.005.6186

tfml 2017
theoretical foundations
of machine learning, Kraków

Optimization of ℓ^p-regularized Linear Models via Coordinate Descent

JACEK KLIMASZEWSKI, MARCIN KORZEŃ
Group, Department
Faculty of Computer Science and Information Technology
ul. Żołnierska 49, 71-210, Szczecin, Poland
e-mail: {*jklimaszewski, mkorzen*}*@wi.zut.edu.pl*

Abstract. In this paper we demonstrate, how ℓ^p-regularized univariate quadratic loss function can be effectively optimized (for $0 \leqslant p \leqslant 1$) without approximation of penalty term and provide analytical solution for $p = \frac{1}{2}$. Next we adapt this approach for important multivariate cases like linear and logistic regressions, using Coordinate Descent algorithm. At the end we compare sample complexity of ℓ^1 with $\ell^p, 0 \leqslant p < 1$ regularized models for artificial and real datasets.

Keywords: Classification, Coordinate Descent, Regression, Sparsity

1. Introduction

In the supervised learning there are two (usually numeric) matrices: $\mathbf{X}_{n\times d}$ and $\mathbf{y}_{n\times 1}$, where n stands for number of observations and d represents number of attributes. The goal is to build such a model, that for unseen examples it would predict correct answers. Therefore final model should contain only relevant features, that were selected at training stage. One of the way to do this is to include regularization term in the loss function, which penalizes coefficients in the model.

Received: 11 December 2016 / Accepted: 30 December 2016

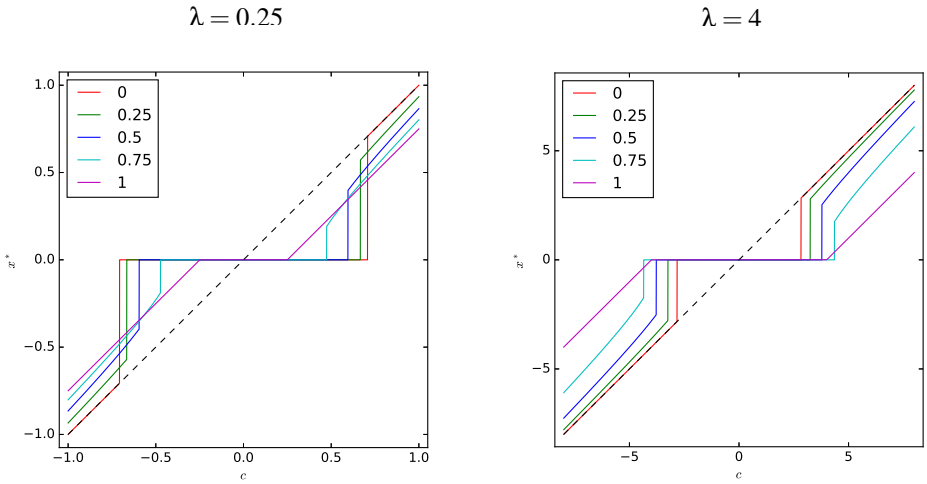


Figure 1. Plot of $x^* = \arg\min_x \frac{1}{2} \cdot (c-x)^2 + \lambda \cdot |x|^p$ for various exponent p and λ.

1.1. Related Work

Historically regularization was used for the first time to solve ill-posed problems [1]. The so called *Tikhonov regularization* (also known as *ridge regression*) penalizes squared ℓ^2-norm of coefficients. It is known that this type of regularization shrinks correlated features, but it also produces dense solutions (all coefficients in the model are non-zero). The same situation is observed for bridge regression (ℓ^p for $p \in (1,2)$) [2], because regularization term is strictly convex.

Next, ℓ^1-norm was tried and it drew a big attention, because it produces sparse solutions [3, 4]. ℓ^1-regularized problem with convex loss function is still convex, but its derivative has discontinuity at origin, which causes certain coefficients to be set exactly to zero. In the Figure 1 it can be seen that solution is shifted from the true coefficient (dashed line), what results in biased models.

When $0 \leqslant p < 1$, ℓ^p-regularized problem with convex loss function becomes non-convex, making optimization more difficult due to many local minima. On the other hand, resulting model has smaller bias. Work by [5] treated non-convex regularization, but local quadratic approximation was used to approximate concavity — this approach is sensitive to initialization, as it can give different solutions for different initial points (once a coefficient is set to zero, it will stay at zero). ℓ^p-"norm" was also mentioned in [6], but authors abandoned using coordinate descent procedure in this case, because they encountered some instability (caused by discontinuity in path of solutions, what is depicted in the Figure 1) and impossibility of converging to the global minimum (using Multi-stage Local Linear Approximation), even for some univariate case. Some non-convex regularizers were also studied in [7] and ℓ^p quasi-norm was considered in [8].

2. The Algorithm for Univariate Case

Consider a function (also known as *proximal operator* [9]):

$$f(x) = \frac{\mu}{2} \cdot (c-x)^2 + \lambda \cdot |x|^p, \tag{1}$$

where $p \in [0,1]$, $c \in \mathbb{R}$ is a true coefficient, $x \in \mathbb{R}$ is its estimate, $\mu > 0$ controls strength of squared error and $\lambda \geq 0$ controls strength of regularization. One may note that μ can be fused with λ and equation (1) can be written in a different way:

$$f(x) = \frac{1}{2} \cdot (c-x)^2 + \rho \cdot |x|^p, \text{ where } \rho = \frac{\lambda}{\mu}, \tag{2}$$

but this form was deliberately omitted for better clarification.

For $p = 1$ equation (1) is still convex and its unique global minimum is given by a soft-thresholding formula [10]:

$$x^* = \operatorname{sgn}(c) \cdot \max\left(0,\ |c| - \frac{\lambda}{\mu}\right). \tag{3}$$

The case $p = 0$ is known as hard-thresholding [10], as it minimizes number of non-zero coefficients. Minimum of equation (1) is either 0 or c[1].

When $p \in (0,1)$, regularization term causes non-convexity, what can be seen in the Figure 2. In this case equation (1) may have 1 minimum (either c for $\lambda = 0$ or 0 for sufficiently large λ) or 2 minima.

2.1. Special Case of $p = \frac{1}{2}$

When $x \geqslant 0$, equation (1) can be transformed into quartic function using substitution $t = x^{\frac{1}{2}}$:

$$g(t) = \frac{\mu}{2} \cdot (c-t^2)^2 + \lambda \cdot t, \tag{4}$$

whose roots (and extrema) can be calculated analytically. Subtracting $\frac{\mu}{2} \cdot c^2$ from equation (4) yields:

$$t \cdot \left(\frac{\mu}{2} \cdot t^3 - \mu \cdot c \cdot t + \lambda\right) = 0. \tag{5}$$

Now we need to find such $\lambda_{\text{critical}}$ for which equation (5) has double root (that coincides with minimum). Using ideas presented in [11] it can be shown that:

$$\lambda_{\text{critical}} = \mu \cdot \left(\frac{2}{3} \cdot c\right)^{\frac{3}{2}}. \tag{6}$$

[1] If $f(\mathbf{x}) = \frac{\mu}{2} \cdot \|\mathbf{c}-\mathbf{x}\|_2^2 + \lambda \cdot \sum_j |x_j|^p$, then each x_j^* can be computed independently (x_j^* is either 0 or c_j). This does not hold in general case $f(\mathbf{x}) = \frac{\mu}{2} \cdot \|\mathbf{c}-\mathbf{A}\mathbf{x}\|_2^2 + \lambda \cdot \sum_j |x_j|^p$.

If $\lambda > \lambda_{\text{critical}}$, then equation (4) has global minimum at zero. Otherwise its stationary points need to be found. Differentiating with respect to t yields cubic equation:

$$t^3 - ct + \frac{\lambda}{2\mu} = 0. \tag{7}$$

In this case equation (7) has 3 real roots (see [11]):

$$\begin{cases} \alpha &= 2 \cdot \sqrt{\frac{c}{3}} \cdot \cos\theta, \\ \beta &= 2 \cdot \sqrt{\frac{c}{3}} \cdot \cos\left(\theta + \frac{2\pi}{3}\right), \\ \gamma &= 2 \cdot \sqrt{\frac{c}{3}} \cdot \cos\left(\theta + \frac{4\pi}{3}\right), \end{cases} \tag{8}$$

where

$$\theta = \frac{1}{3} \cdot \arccos\left(-\frac{\lambda}{4\mu\left(\frac{c}{3}\right)^{\frac{3}{2}}}\right). \tag{9}$$

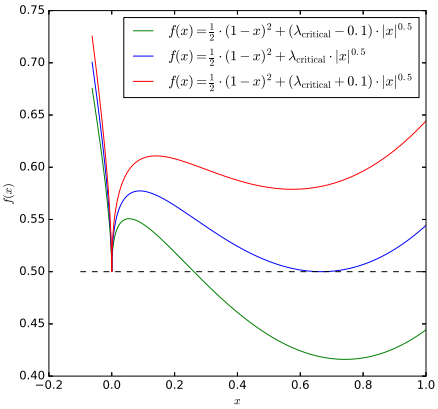


Figure 2. Plot of eq. (1) for different λ.

α is a local minimum of equation (4), γ is a local maximum and β is ignored, because it is negative. Since $t = x^{\frac{1}{2}}$, α need to be squared to obtain x^*. Negative case is handled similarly.

2.2. Algorithm for General Case of $p \in (0,1)$

In general case Newton's method [12] can be used to find minimum of equation (1), because it is differentiable for $x \neq 0$:

$$f'(x) = \mu \cdot (x - c) + \lambda \cdot p \cdot \operatorname{sgn}(x) \cdot |x|^{p-1}, \tag{10}$$

$$f''(x) = \mu + \lambda \cdot p \cdot (p-1) \cdot |x|^{p-2}. \tag{11}$$

If $c = 0$, then $x = 0$ is a global minimum.

$\lambda_{\text{critical}}$ in general case was found in a similar way to case $p = \frac{1}{2}$ — formula $f(x) - \frac{\mu}{2}c^2$ has 2 minima that coincide with roots when $\lambda = \lambda_{\text{critical}}$. To find $\lambda_{\text{critical}}$, the following system of equations has to be solved:

$$\begin{cases} f(x) - \frac{\mu}{2}c^2 = 0 \\ f'(x) = 0 \end{cases} \tag{12}$$

It can be solved via substitution — the solution is:

$$\begin{cases} x = \frac{2-2p}{2-p} \cdot c, \\ \lambda_{\text{critical}} = \frac{\mu|c|}{2-p} \cdot \left(\frac{2-2p}{2-p} \cdot |c|\right)^{1-p}. \end{cases} \tag{13}$$

Now minimization of (1) is much easier: firstly, value $\lambda_{\text{critical}}$ is computed and then condition $\lambda_{\text{critical}} \leq \lambda$ is checked — if it is met, then there is nothing to do, because global minimum

is at zero (we select $x = 0$ even when $\lambda_{\text{critical}} = \lambda$, though global minimum is not unique in that case). Otherwise Newton's method is launched for a starting point $x_0 = c$ (it is very close to the minimum, so only a few iterations are needed).

3. The algorithm for multivariate case

In this section it is shown how Coordinate Descent method is applied to minimize ℓ^p-regularized residual sum of squares (RSS). Then this approach is used to estimate coefficients of the logistic regression model via iteratively reweighted least squares.

3.1. Basic Algorithm for Linear Regression

Coefficients w_j of the linear regression model are estimated by minimization of residual sum of squares (RSS). ℓ^p-regularized loss function has a form (intercept w_0 is not regularized):

$$L(\mathbf{w}) = \frac{1}{2} \cdot \sum_i \left(y_i - w_0 - \sum_j x_{ij} w_j \right)^2 + \lambda \cdot \sum_j |w_j|^p, \tag{14}$$

where y_i is the i-th value of the variable to be predicted and $\mathbf{x}_i$ is the i-th row of the explanatory matrix $\mathbf{X}$. Differentiating equation (14) with respect to w_j yields following formulae:

$$\frac{\partial L}{\partial w_j} = \sum_i x_{ij}^2 \cdot \left(w_j - \frac{\sum_i x_{ij} \cdot \left(y_i - \sum_{k \neq j} x_{ik} w_k \right)}{\sum_i x_{ij}^2} \right) + \lambda \cdot p \cdot \operatorname{sgn}(w_j) \cdot |w_j|^{p-1}, \tag{15}$$

$$\frac{\partial^2 L}{\partial w_j^2} = \sum_i x_{ij}^2 + \lambda \cdot p \cdot (p-1) \cdot |w_j|^{p-2} \tag{16}$$

From equations (15) and (16) it can be seen that $c_j = \frac{\sum_i x_{ij} \cdot \left(y_i - \sum_{k \neq j} x_{ik} w_k \right)}{\sum_i x_{ij}^2}$ and $\mu_j = \sum_i x_{ij}^2$, so:

$$\frac{\partial L}{\partial w_j} = \mu_j \cdot (w_j - c_j) + \lambda \cdot p \cdot \operatorname{sgn}(w_j) \cdot |w_j|^{p-1}, \tag{17}$$

$$\frac{\partial^2 L}{\partial w_j^2} = \mu_j + \lambda \cdot p \cdot (p-1) \cdot |w_j|^{p-2}. \tag{18}$$

Now it is clear that equations (17) and (18) have the same form as equations (10) and (11) respectively. Hence Coordinate Descent procedure can be constructed:

The algorithm is stopped when it exceeds maximum number of iterations or maximum relative difference between $w_j^{(k)}$ and $w_j^{(k+1)}$ falls below $\varepsilon = 10^{-5}$ (or some other value specified by user).

Algorithm 1 Coordinate Descent procedure for ℓ^p-regularized Linear Regression.

while has not converged **do**
  $w_0 \leftarrow \frac{1}{n} \cdot \sum_i \left(y_i - \sum_j w_j x_{ij}\right)$
  for $j \leftarrow 1$ **to** d **do**
    Compute c and $\lambda_{\text{critical}}$ for j-th coordinate.
    if $\lambda_{\text{critical}} \leq \lambda$ **then**
      $w_j \leftarrow 0$
    else
      $w_j \leftarrow \text{NewtonAlgorithm}(\frac{\partial L}{\partial w_j}, \frac{\partial^2 L}{\partial w_j^2}, c)$
    end if
  end for
end while

3.2. Basic Algorithm for Logistic Regression

A similar approach can be adapted for fitting the logistic regression model. Here we consider the case where $\mathbf{y}$ is a binary variable ($y_i \in \{0,1\}$):

$$P(y_i = 1|\mathbf{x}_i, \mathbf{w}) = \frac{1}{1+\exp(-w_0 - \sum_j w_j x_{ij})}, \tag{19}$$

$$P(y_i = 0|\mathbf{x}_i, \mathbf{w}) = 1 - P(y_i = 1|\mathbf{x}_i, \mathbf{w}) = \frac{1}{1+\exp(w_0 + \sum_j w_j x_{ij})}. \tag{20}$$

A common way of estimation of model's coefficients is minimization of the negative log-likelihood function (or, equivalently, maximization of the likelihood function). The ℓ^p-regularized negative log-likelihood function has a form:

$$L(\mathbf{w}) = -\sum_i y_i \cdot \log\left(p\left(\mathbf{x}_i\right)\right) + \left(1 - y_i\right) \cdot \log\left(1 - p\left(\mathbf{x}_i\right)\right) + \lambda \cdot \sum_j |w_j|^p, \tag{21}$$

where $p(\mathbf{x}_i) = P(y_i = 1|\mathbf{x}_i, \mathbf{w})$.

Equation (21) does not have closed-form solution, but it can be locally approximated by quadratic function [13]. Introduction of weights:

$$\alpha_i = p(\mathbf{x}_i) \cdot \left(1 - p(\mathbf{x}_i)\right), \tag{22}$$

$$z_i = w_0 + \sum_j w_j x_{ij} + \frac{y_i - p(\mathbf{x}_i)}{\alpha_i}, \tag{23}$$

allows us to rewrite equation (21) to the form:

$$L(\mathbf{w}) = \frac{1}{2} \cdot \sum_i \alpha_i \cdot \left(z_i - w_0 - \sum_j w_j x_{ij}\right)^2 + \lambda \cdot \sum_{j=1} |w_j|^p. \tag{24}$$

Now minimization of equation (21) is changed to the successive minimization of equation (24) — for $\mathbf{w}^{(k)}$ update weights and compute $\mathbf{w}^{(k+1)}$ until convergence.

Finally a similar algorithm to the Algorithm 1 of fitting logistic regression model is presented below:

Algorithm 2 Coordinate Descent for ℓ^p-regularized Logistic Regression.

while has not converged **do**
  Update α and $\mathbf{z}$ using current $\mathbf{w}$.
  while not converged **do**
    $w_0 \leftarrow \frac{\sum_i \alpha_i \cdot \left(z_i - \sum_j w_j x_{ij}\right)}{\sum_i \alpha_i}$
    for $j \leftarrow 1$ **to** d **do**
      Compute c and $\lambda_{\text{critical}}$ for j-th coordinate.
      if $\lambda_{\text{critical}} \leq \lambda$ **then**
        $w_j \leftarrow 0$
      else
        $w_j \leftarrow \text{NewtonAlgorithm}(\frac{\partial L}{\partial w_j},\ \frac{\partial^2 L}{\partial w_j^2},\ c)$
      end if
    end for
  end while
end while

3.3. Improvements to the Basic Algorithms

Ad hoc implementation of above algorithms is not efficient for large datasets. To improve it, ideas described in [14] were used — i.e. naive updates, pathwise coordinate descent and computation over *active set* of features.

4. Experiments

Experimental part was written in Python. Procedures for estimation of coefficients of ℓ^p-regularized linear and logistic regression were implemented in C++ and called from Python script via *ctypes* package. Vectorized matrices using column-major order were passed to the C++ routines to speed-up computation. Some parts of *glmnet* [14] were used during implementation of our solution. In each experiment tolerance ε was set to 10^{-5}.

We use three test data sets:

1. DataSet#1 is artificial set of size $100 \times 1000(4)$ (consisting of 100 samples with 1000 attributes drawn from multivariate normal distribution and only 4 significant attributes), output is a linear combination of 4 significant variables (in case of logistic regression it was sign of this linear combination);

2. DataSet#2 is artifical set of size $100 \times 1000(32)$ generated similarly;

3. DataSet#3 is a Golub's `Leukemia` dataset [15], preprocessed by [16]. This dataset has 38 training samples and 34 test samples.

4.1. Impact of p on coefficients' paths

The first experiment shows the coefficients' path for linear and logistic regression models for $p = 0, 0.333, 0.667, 1$. The results are presented in the Figures 3 and 4. The vertical dashed line shows value of λ that yields optimal model (with the highest accuracy). The accuracy measure in the case of linear regression is RSS, and for logistic regression it is the accuracy of classification. As one can see, in both cases ℓ^1-regularized regressions include some number of random attributes in the optimal model. In the case $p < 1$ the optimal model selects 4 significant attributes correctly — this shows a qualitative difference between ℓ^1 and $\ell^p, p < 1$ penalty terms. The second observation is that for smaller exponents p coefficients' path is more robust with respect to λ (for $p = 0$ coefficients' paths are piecewise constant).

It can also be noted that in case $d \gg n$ when λ is close to 0, coefficients of logistic regression wander off to $\pm\infty$ in order to achieve probabilities of 0 or 1, but this is natural.

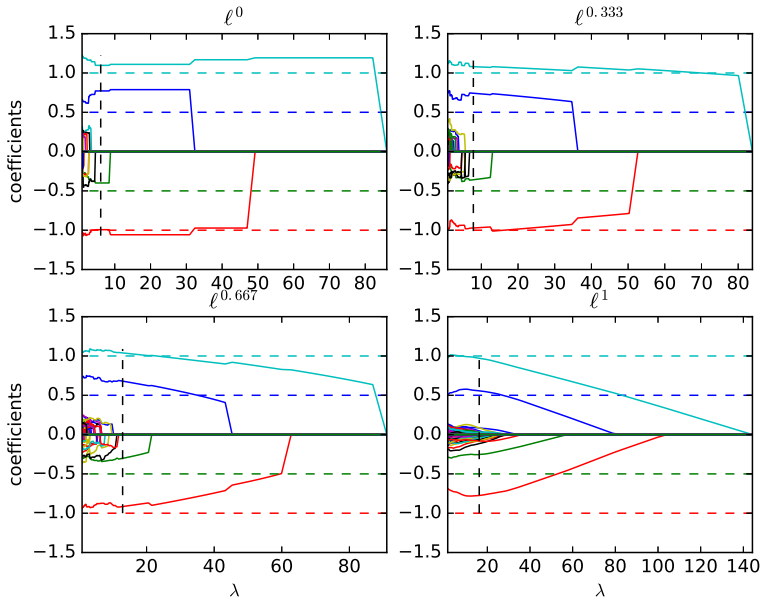


Figure 3. Paths of coefficients for the linear regression for DataSet#1; horizontal dashed lines represent true coefficients, vertical dashed line depicts value of λ for which model has the lowest RSS, tested via 10-fold cross validation procedure.

4.2. Sample complexity of Logistic Regression

In the next experiment we compared influence of exponent p on sample complexity of the model. We varied number of samples in the train set, next we tested each classifier using test set, we took coefficients from the solution's path (computed for 65 values of λ, $\lambda_{\min} = 0.01 \cdot \lambda_{\max}$) and we computed prediction for each solution on the path. This process was

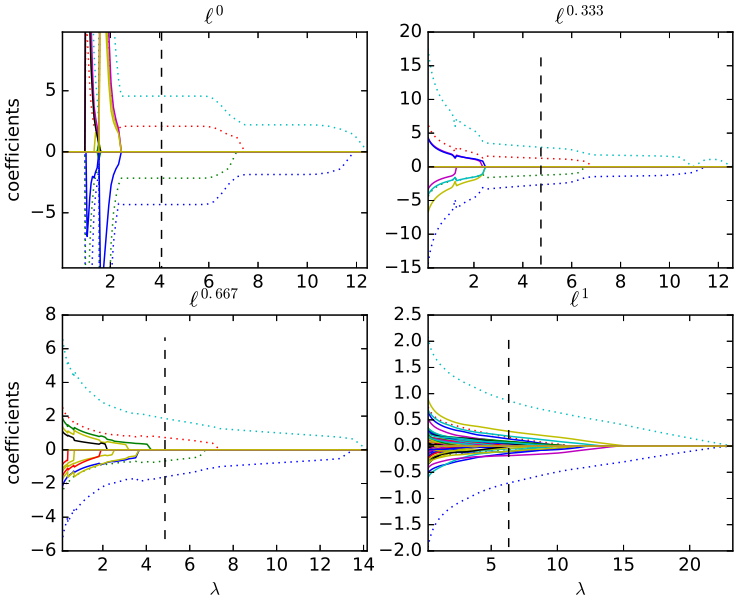


Figure 4. Paths of coefficients for the logistic regression for DataSet#1; vertical dashed line depicts value of λ for which model has the highest classification accuracy, tested via 10-fold cross-validation procedure; dotted line corresponds to relevant features.

repeated 50 times and the accuracy was averaged. Best results for each sample size are presented in the Figure 5 for DataSet#1 and in the Figure 6 for DataSet#2. As the number of training samples rises, accuracies of all models grows. It can be seen that accuracies of ℓ^1- and $\ell^{0.75}$-regularized models are roughly the same in the beginning, but later $\ell^{0.75}$ is superior to ℓ^1. Surprisingly, models for $p \leqslant 0.5$ gave almost the same accuracies. Generally results show that models with smaller exponent p achieve the desired accuracy earlier.

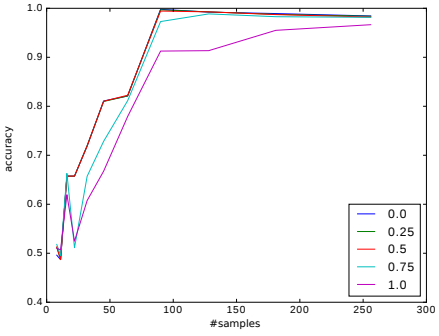


Figure 5. Results of the experiment with 4 relevant features.

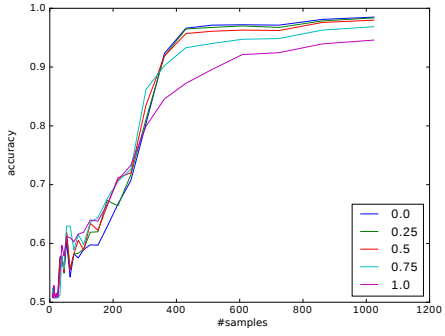


Figure 6. Results of the experiment with 32 relevant features.

4.3. Test on Leukemia dataset

Logistic regression model was trained on preprocessed `Leukemia` dataset using leave-one-out cross-validation for 100 values of λ and $\lambda_{\min} = 0.001 \cdot \lambda_{\max}$. Table 1 presents results of the experiment.

Table 1. Accuracy on the test set.

p	Timing	Accuracy	AUC	#nnz
0.0	1.78	32/34	0.936	2
0.25	16.17	30/34	0.925	1
0.5	19.20	33/34	0.993	1
0.75	15.04	32/34	0.968	2
1.0	7.20	31/34	0.989	7

It can be seen that all models for $p < 1$ gave sparser solution than ℓ^1-regularized model. Also models for $p \geqslant 0.5$ are more accurate (higher area under ROC curve). Although $\ell^{0.5}$-regularized model seems to be superior, it is hard to select the best model in this case, because dataset is relatively small.

5. Conclusions

In the paper we have shown that ℓ^p-regularized regressions can be effectively fitted via adapted version of pathwise coordinate descent algorithm. The results show that in some cases models with penalty function for smaller exponent p can lead to models with some attractive features like:

- sparser solutions;
- more robust coefficients' paths with respect to λ;
- smaller exponents p, close to $p = 0$, have a better selecting properties in the case of independent attributes.

The case of correlated variables, which was in fact omitted in the paper, needs further research, because all considered models should be corrected in the similar way to elastic-net model [17].

6. References

[1] Tikhonov, A.N., *On the stability of inverse problems (in Russian)*. Doklady Akademii Nauk SSSR, 1943, **39**(5), pp. 195–198.

[2] Frank, I.E., Friedman, J.H., *A Statistical View of Some Chemometrics Regression Tools*. Technometrics, 1993, **35**, pp. 109–148.

[3] Williams, P.M., *Bayesian Regularisation and Pruning using a Laplace Prior*. Neural Computation, 1994, **7**, pp. 117–143.

[4] Tibshirani, R., *Regression Shrinkage and Selection Via the Lasso*. Journal of the Royal Statistical Society, Series B, 1996, **58**, pp. 267–288.

[5] Fan, J., Li, R., *Variable Selection via Nonconcave Penalized Likelihood and its Oracle Properties*. Journal of the American Statistical Association, 2001, **96**, pp. 1348–1360.

[6] Mazumder, R., Friedman, J., Hastie, T., *SparseNet: Coordinate Descent With Nonconvex Penalties*. Journal of the American Statistical Association, 2011, **106**(495), pp. 1125–1138.

[7] Nikolova, M., *Analysis of the Recovery of Edges in Images and Signals by Minimizing Nonconvex Regularized Least-Squares*. Multiscale Modeling & Simulation, 2005, **4**(3), pp. 960–991.

[8] Bredies, K., Lorenz, D., Reiterer, S., *Minimization of Non-smooth, Non-convex Functionals by Iterative Thresholding*. Journal of Optimization Theory and Applications, 2015, **165**, pp. 78–112.

[9] Moreau, J.J., *Fonctions convexes duales et points proximaux dans un espace hilbertien.* Comptes Rendus de l'Académie des Sciences (Paris), Série A, 1962, **255**, pp. 2897–2899.

[10] Donoho, D., Johnstone, I., *Ideal Spatial Adaptation by Wavelet Shrinkage*. Biometrika, 1994, **81**, pp. 425–455.

[11] Nickalls, R.W.D., *A New Approach to Solving the Cubic: Cardan's Solution Revealed*. The Mathematical Gazette, 1993, **77**(480), pp. 354–359.

[12] Kincaid, D., Cheney, W., *Numerical Analysis: Mathematics of Scientific Computing*. American Mathematical Society, 2002.

[13] Green, P.J., *Iteratively Reweighted Least Squares for Maximum Likelihood Estimation, and some Robust and Resistant Alternatives*. Journal of the Royal Statistical Society. Series B (Methodological), 1984, **46**(2).

[14] Friedman, J., Hastie, T., Tibshirani, R., *Regularization Paths for Generalized Linear Models via Coordinate Descent*. Journal of Statistical Software, 2010, **33**(1), pp. 1–22.

[15] Golub, T., Slonim, D., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J., Coller, H., Loh, M., Downing, J., Caligiuri, M., Bloomfield, C., Lander, E., *Molecular classification of cancer: class discovery and class prediction by gene expression monitoring.* Science, 1999, **286**(5439), pp. 531–537.

[16] Dettling, M., *BagBoosting for Tumor Classification with Gene Expression Data*. Bioinformatics, 2004, **20**(18), pp. 3583–3593.

[17] Zou, H., Hastie, T., *Regularization and Variable Selection via the Elastic Net*. Journal of the Royal Statistical Society, Series B, 2005, **67**, pp. 301–320.

Schedae Informaticae Vol. 25 (2016): 73–83
doi: 10.4467/20838476SI.16.006.6187

tfml 2017
theoretical foundations
of machine learning, Kraków

Pairwise versus Pointwise Ranking: A Case Study

Vitalik Melnikov[1], Pritha Gupta[1], Bernd Frick[2],
Daniel Kaimann[2], Eyke Hüllermeier[1]
[1]Department of Computer Science
[2]Faculty of Business Administration and Economics
Paderborn University
Warburger Str. 100, 33098 Paderborn
e-mail: {*melnikov,prithag,eyke*}*@mail.upb.de,* {*bernd.frick,daniel.kaimann*}*@upb.de*

Abstract. Object ranking is one of the most relevant problems in the realm of preference learning and ranking. It is mostly tackled by means of two different techniques, often referred to as pairwise and pointwise ranking. In this paper, we present a case study in which we systematically compare two representatives of these techniques, a method based on the reduction of ranking to binary classification and so-called expected rank regression (ERR). Our experiments are meant to complement existing studies in this field, especially previous evaluations of ERR. And indeed, our results are not fully in agreement with previous findings and partly support different conclusions.

Keywords: Preference learning, object ranking, linear regression, logistic regression, hotel rating, TripAdvisor

1. Introduction

Preference learning is an emerging subfield of machine learning that has received increasing attention in recent years [1]. Roughly speaking, the goal in preference learning is to induce preference models from observed data that reveals information

Received: 11 December 2016 / Accepted: 30 December 2016

about the preferences of an individual or a group of individuals in a direct or indirect way; these models are then used to predict the preferences in a new situation.

In general, a preference learning system is provided with a set of items (e.g., products) for which preferences are known, and the task is to learn a function that predicts preferences for a new set of items (e.g., new products not seen so far), or for the same set of items in a different context (e.g., the same products but for a different user). Frequently, the predicted preference relation is required to form a total order, in which case we also speak of a *ranking problem*. In fact, among the problems in the realm of preference learning, the task of "learning to rank" has probably received the most attention in the literature so far, and a number of different ranking problems have already been introduced. Based on the type of training data and the required predictions, Fürnkranz and Hüllermeier [1] distinguish between the problems of object ranking [2, 3], label ranking [4, 5, 6] and instance ranking [7].

The focus of this paper is on object ranking. What we present is an empirical study in which we compare the two most common approaches to this problem: pairwise ranking and pointwise ranking, with the latter being represented by a method called *expected rank regression* [8, 9, 3]. Although we are not the first to conduct experiments of that kind, our study sheds new light on the comparison of these two techniques and helps to better understand their advantages and disadvantages.

The rest of the paper is organized as follows. In the next section, we recall the problem of object ranking as well as the techniques of pairwise and pointwise ranking. The ranking data used for the purpose of our case study is described in Section 3. The design of the experiments and the results obtained are then presented in Section 4, prior to concluding the paper in Section 5.

2. Object ranking

Consider a reference set of objects or items $\mathcal{X}$, and assume each item $\boldsymbol{x} \in \mathcal{X}$ to be described in terms of a feature vector; thus, an item is a vector $\boldsymbol{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d$ and $\mathcal{X} \subseteq \mathbb{R}^d$. Training data consists of a set of rankings $\{O_1, \ldots, O_N\}$, where each ranking O_j is a total order of a subset of $n_j = |O_j|$ items $\boldsymbol{x}_{j_i} \in \mathbb{X}$:

$$O_j : \boldsymbol{x}_{j_1} \succ \boldsymbol{x}_{j_2} \succ \ldots \succ \boldsymbol{x}_{j_{n_j}} \tag{1}$$

The order relation $\succ$ is typically (though not necessarily) interpreted in terms of preferences, i.e., $\boldsymbol{x} \succ \boldsymbol{x}'$ suggests that $\boldsymbol{x}$ is preferred to $\boldsymbol{x}'$.

The goal in object ranking is to learn a *ranking function* that accepts any (query) subset $Q \subseteq \mathcal{X}$ of $n = |Q|$ items as input. As output, the function produces a ranking (total order) O of these items. This prediction is evaluated in terms of a suitable loss function or performance metric; a common choice is the Kendall τ correlation, which counts the number of item pairs $\boldsymbol{x}, \boldsymbol{x}' \in Q$ that are incorrectly ordered by O and normalizes this number (which is between 0 and $n(n-1)/2$) to the range $[-1, +1]$.

2.1. Representation and learning

The ranking function sought in object ranking is a complex mapping from $2^{\mathcal{X}}$ to the set of all total orders over subsets of $\mathcal{X}$. A first question, therefore, is how to represent a "ranking-valued" function of that kind, and a second one is how it can be learned efficiently.

As for the question of representation, a ranking function is typically implemented by means of a scoring function $U : \mathcal{X} \longrightarrow \mathbb{R}$, so that $\boldsymbol{x} \succ \boldsymbol{x}'$ if $U(\boldsymbol{x}) > U(\boldsymbol{x}')$ for all $\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}$. In other words, a ranking-valued function is implicitly represented by a real-valued function. Obviously, U can be considered as a kind of utility function, and $U(\boldsymbol{x})$ as a latent utility degree assigned to an item $\boldsymbol{x}$. Seen from this point of view, the goal in object ranking is to learn a latent utility function on a reference set $\mathcal{X}$. In the following, we shall also refer to U itself as a ranking function.

The representation of a ranking function in terms of a real-valued (utility) function also suggests natural approaches to learning. In particular, two such approaches are prevailing in the literature. The first one reduces the original ranking problem to *regression*; as it seeks a model that assigns appropriate scores to individual items $\boldsymbol{x}$, it is referred to as the *pointwise* approach. The second idea is to reduce the problem to *binary classification*; here, the focus is on pairs of items, which is why the approach is called the *pairwise* approach.

2.2. Pairwise ranking

Given a ranking (1) as training information, the pairwise approach extracts all pairwise preferences $\boldsymbol{x}_{j_i} \succ \boldsymbol{x}_{j_k}$, $1 \leq i < k \leq n_j$, and considers these preferences as examples for a binary classification task. This approach is especially simple if U is a linear function of the form $U(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x}$. In this case, $U(\boldsymbol{x}) > U(\boldsymbol{x}')$ if $\boldsymbol{w}^\top \boldsymbol{x} > \boldsymbol{w}^\top \boldsymbol{x}'$, which is equivalent to $\boldsymbol{w}^\top \boldsymbol{z} > 0$ for $\boldsymbol{z} = \boldsymbol{x} - \boldsymbol{x}' \in \mathbb{R}^d$. Thus, from the point of view of binary classification (with a linear threshold model), $\boldsymbol{z}$ can be considered as a positive and $-\boldsymbol{z}$ as a negative example.

In principle, any binary classification algorithm can be applied to learn the weight vector $\boldsymbol{w}$ from set of examples produced in this way. In the case of logistic regression, the resulting model has a specifically nice interpretation. Given two items $\boldsymbol{x}$ and $\boldsymbol{x}'$, the model produces a probability for the preference $\boldsymbol{x} \succ \boldsymbol{x}'$ and a complementary probability for $\boldsymbol{x}' \succ \boldsymbol{x}$. The former corresponds to the probability of a positive label $y = +1$ for the instance $\boldsymbol{z} = \boldsymbol{x} - \boldsymbol{x}'$, i.e.,

$$\begin{aligned} \mathbf{P}(\boldsymbol{x} \succ \boldsymbol{x}') = \mathbf{P}(y = +1 \,|\, \boldsymbol{z}) &= \frac{1}{1 + \exp(-\boldsymbol{w}^\top (\boldsymbol{x} - \boldsymbol{x}'))} \\ &= \frac{\exp(U(\boldsymbol{x}))}{\exp(U(\boldsymbol{x})) + \exp(U(\boldsymbol{x}'))} \end{aligned}$$

Thus, observed preferences are supposed to follow the Bradley-Terry model of discrete choice [10]: Having to choose between two options $\boldsymbol{x}$ and $\boldsymbol{x}'$, the probability for deciding in favor of either of them is proportional to the (exponential of the) corresponding

utility. The maximum likelihood estimator $\boldsymbol{w}$ then simply maximizes the probability of the observed preferences under this choice model.

2.3. Pointwise ranking

Pointwise ranking methods induce a (utility) function $U : \mathcal{X} \longrightarrow \mathbb{R}$ as well. To do so, however, they fit a regression function to training examples of the form $(\boldsymbol{x}_i, y_i)$. Here, an obvious question concerns the definition of the target values y_i. In the setting of object ranking as introduced above, only *relative* information about the preference of items $\boldsymbol{x}_i$ in comparisons to others is given, but no *absolute* evaluations that could immediately be associated with a single $\boldsymbol{x}_i$.

Obviously, a reasonable target y_i for an item $\boldsymbol{x}_i$ is its (relative) position in $\mathcal{X}$, i.e., $y_i = \#\{\boldsymbol{x} \in \mathcal{X} \,|\, \boldsymbol{x}_i \succ \boldsymbol{x}\}$,[1] because sorting according to these scores yields perfect ranking performance. Again, however, since only rankings O_j of *subsets* of $\mathcal{X}$ are observed, these scores are not part of the training data.

In the method of expected rank regression (ERR), the scores are therefore approximated in terms of their expectation [8, 9, 3]. More specifically, given a ranking O_j of length n_j, an item $\boldsymbol{x}_i$ ranked on position r_i in O_j is assigned the score $y_i = r_i/(n_i+1)$. This is justified by taking an expectation over all (complete) rankings of $\mathcal{X}$ and assuming a uniform distribution. Roughly speaking, the items in O_j are assumed to be distributed uniformly among the whole spectrum of ranks.

2.4. Pairwise versus pointwise ranking

The pointwise approach solves a regression problem on $|O_1| + \ldots + |O_N|$ training examples in total; thus, if $|O_j| \approx K$, the size of the training data is of the order $\mathcal{O}(KN)$. The number of examples created by the pairwise approach for binary classification is of the order $\mathcal{O}(K^2N)$—although, at the cost of a slight loss of information, it could be reduced to $\mathcal{O}(KN)$ as well, namely by only extracting consecutive preferences $\boldsymbol{x}_{j_i} \succ \boldsymbol{x}_{j_{i+1}}$ from (1). In any case, linear regression is simpler and computationally less expensive than methods for binary classification, such as logistic regression. Thus, from the point of view of complexity, the pointwise approach seems to be preferable.

Also note that, while the pointwise approach leaves the rankings O_j intact, the pairwise approach splits a ranking O_j into pairwise comparisons. This necessarily comes with a loss of information, even when generating the full set of comparisons. This is because, statistically, the probability of observing the ranking as a whole is normally different from the probability of observing the set of pairwise preferences independently of each other.

That being said, the assignment of scores y_i in ERR is based on a rather strong and arguably unrealistic assumption. Moreover, these scores do not reflect an inherent

[1] assuming $\mathcal{X}$ is finite

property of an item $\boldsymbol{x}_i$, but instead depend on the context in which $\boldsymbol{x}_i$ is observed. Therefore, one may wonder whether predicting the y_i as a function of item-features is possible at all. Roughly speaking, the pointwise approach could be questioned because it adds information to the training data that is actually not present. This information is unreliable at best and misleading at worst. The pairwise approach, on the other hand, extracts only qualitative information. This information is weaker but indeed valid.

Finally, one may wonder whether a regression approach is suitable for fitting (relative) ranks, because ranks are only measured on an ordinal scale. In this regard, however, one should also note that the regression function U is not required to fit the data well in a numerical sense, i.e., in terms of the squared error loss. Instead, as it is only used for the purpose of ranking, any function that is comonotonic with the ranks is equally good.

Empirically, ERR has indeed been shown to be competitive and sometimes even superior to other ranking methods [8, 9, 3], albeit under experimental conditions that agree with the assumptions underlying this method. This paper is meant to complement these experiments by another case study, in which we systematically control certain properties of the training data in order to see to what extent they affect ERR. In particular, we are interested in scenarios that violate the assumptions of ERR. The hypotheses of this study are as follows:

H1: Due to the disputable way in which target values are produced for training in ERR, this method should in general be inferior to pairwise ranking.

H2: In particular, the shorter the training rankings O_j, the less accurate the approximation of target values in terms of expected ranks, and hence the worse the performance of ERR should be. Likewise, we suspect that the variance of the lengths $n_1, \ldots, n_N$ of the rankings in the training data has a negative influence.

H3: The performance of ERR will also drop due to a violation of the assumption of uniform sampling of positions.

3. TripAdvisor hotel dataset

Our case study deals with the ranking of hotels. The dataset used for performing the experiments was taken from the TripAdvisor website,[2] using a combination of web crawling and web scraping tools, on September 21 and 22, 2014. The dataset contains five rankings for hotels in five major German cities: Düsseldorf (110 hotels), Hamburg (170 hotels), Berlin (363 hotels), Frankfurt (149 hotels), and Munich (194 hotels). These rankings are referred to as *complete rankings* in the rest of this paper (they correspond to the reference set $\mathcal{X}$).

The position of each hotel in the ranking is determined by the so-called *popularity index*, which is computed by TripAdvisor based on the reviews for the hotels.

[2] www.tripadvisor.com

Although the true underlying computational formula (utility function) is unknown, several major factors contributing to this index are mentioned on the website.[3] These include the number of reviews, the age of reviews, and the overall quality of reviews. We used these attributes as features for each hotel in the dataset and complemented them by a set of additional attributes: distance to the city center (real number), number of hotel stars (ordinal), number of pictures on the TripAdvisor website (natural number), number of hotel rooms (natural number), average price per double room (real number), recommendation percent (percentage), number of reviews, five numerical features containing the number of ratings (from very good to very poor) given by the reviewers, and six real-valued features with average rating for different hotel attributes (location, sleep quality, room service, cost benefit, and cleanliness).

4. Experiments

We compare ERR with pairwise ranking based on logistic regression (LR). To guarantee a fair comparison, a linear utility function $U(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x}$ is used in both approaches. Moreover, the preprocessing of the data, including a standardization of all input attributes, was done in exactly the same way.

The general experimental design is as follows: We use one complete ranking (Berlin with 363 hotels) to generate training data in the form of incomplete rankings $O_1, \ldots, O_N$. LR and ERR are trained on this data as described in Sections 2.2. and 2.3., respectively. The models thus obtained are then evaluated on the remaining four cities: The four complete rankings are predicted, the Kendall correlation is determined for each of them individually, and finally the correlations are averaged. All experiments are repeated 100 times.

To test our hypotheses, the sampling procedure was controlled as follows:

- The lengths $n_j = |O_j|$ were sampled (independently) at random from a uniform distribution on $\{K - d, \ldots, K + d\}$. Thus, the average length of an observation is K, and the standard deviation is proportional to d. We chose $K \in \{5, 10, 20, 50, 100, 250\}$ and $d \in \{0, 1, 3, 4, 5, 7, 17, 47, 97\}$.[4]
- To make the results of different experiments comparable, regardless of the parameters K and d, we set $N = 1000/K$. Thus, the total number of hotels included in a sample is always 1000 (on average).
- After the length n_j of a ranking has been obtained, the ranking O_j itself is produced by randomly sampling n_j hotels in Berlin (and keeping their original order). For the sampling procedure, three different scenarios are considered:
  - *Uniform.* In this case, hotels are drawn uniformly at random (without replacement).

[3] https://www.tripadvisor.com/TripAdvisorInsights/n684/tripadvisor-popularity-ranking-key-factors-and-how-improve

[4] Since a length cannot be negative, not all (K, d) combinations are possible.

- *Top hotels.* This procedure has a bias in favor of hotels in the top of the list. First, we randomly pick one hotel from the first 50 in the complete ranking. This hotel is removed, and the next one is chosen among its neighbors, namely the three ones above and below. This procedure is continued until n_j hotels are collected.
- *Two groups.* The same sampling procedure as in the previous case is used (with a smaller neighborhood of 2 instead of 3), but the first hotel is randomly taken from first or the last 50 hotels.

The results of the experiments are summarized in Table 1 in terms of the average Kendall τ correlation and its standard deviation. Moreover, the following loss/gain of performance of ERR relative to LR is shown in the form of heatmaps in Figure 1:

$$\frac{\tau_{LR} - \tau_{ERR}}{\tau_{LR}} \tag{2}$$

The following conclusions can be drawn from these results:

- The pairwise approach (LR) consistently outperforms the pointwise approach (ERR) in all experiments. Moreover, pairwise ranking remains relatively stable across all settings, whereas the performance of ERR is much more sensitive toward the parameters. Thus, our hypothesis H1 is clearly confirmed.
- Our conjecture that ERR benefits from longer rankings is confirmed as well. Indeed, the performance of ERR clearly improves with increasing K. For the variance of the lengths, a clear trend is not visible. A problem here could be that the mean and variance of the length cannot be separated completely: Increasing d will always lead to producing a few rankings that are longer than K, which might be beneficial for ERR. Anyway, our conjecture H2 is confirmed only partially.
- Both approaches perform best in the case of *uniform* sampling. This was to be expected, since uniform sampling produces observations from the complete feature space. In the *top hotels* scenario, the pairwise approach remains rather stable, whereas ERR significantly drops in performance and seems to predict almost at random. A similar picture is obtained for the *two groups* scenario. These results clearly support our conjecture H3, namely that ERR is very sensitive toward deviations from the assumption of uniform sampling.

Table 1. Mean ± standard deviation of Kendall's tau for the *uniform* scenario (top), *top hotels* (middle), and *two groups* (bottom).

Approach	K/d	0	1	3	4	5	7	17	47	97
Pointwise	2	.248 ± .228	-	-	-	-	-	-	-	-
Pairwise	2	.832 ± .010	-	-	-	-	-	-	-	-
Pointwise	5	.390 ± .229	.302 ± .221	.154 ± .204	-	-	-	-	-	-
Pairwise	5	.840 ± .006	.840 ± .005	.839 ± .006	-	-	-	-	-	-
Pointwise	10	.377 ± .179	.432 ± .145	.536 ± .104	.555 ± .125	.567 ± .111	.615 ± .091	-	-	-
Pairwise	10	.841 ± .005	.841 ± .005	.840 ± .005	.840 ± .005	.839 ± .006	.839 ± .005	-	-	-
Pointwise	20	.707 ± .054	.703 ± .053	.700 ± .063	.712 ± .064	.713 ± .048	.719 ± .038	.743 ± .037	-	-
Pairwise	20	.841 ± .004	.841 ± .004	.841 ± .005	.841 ± .005	.840 ± .005	.841 ± .005	.840 ± .005	-	-
Pointwise	50	.786 ± .016	.785 ± .016	.784 ± .019	.786 ± .019	.785 ± .019	.785 ± .017	.788 ± .018	.794 ± .016	-
Pairwise	50	.841 ± .004	.841 ± .004	.841 ± .004	.841 ± .005	.841 ± .004	.842 ± .005	.841 ± .005	.841 ± .004	-
Pointwise	100	.803 ± .009	.804 ± .009	.802 ± .010	.805 ± .009	.804 ± .009	.803 ± .010	.801 ± .010	.802 ± .009	.806 ± .008
Pairwise	100	.841 ± .004	.841 ± .004	.841 ± .004	.841 ± .004	.841 ± .004	.842 ± .003	.841 ± .004	.842 ± .004	.842 ± .004
Pointwise	250	.812 ± .003	.811 ± .003	.812 ± .004	.812 ± .004	.811 ± .004	.812 ± .004	.811 ± .004	.811 ± .004	.812 ± .004
Pairwise	250	.843 ± .003	.842 ± .003	.843 ± .003	.843 ± .003	.842 ± .003	.843 ± .003	.843 ± .003	.842 ± .003	.843 ± .002

Approach	K/d	0	1	3	4	5	7	17	47	97
Pointwise	2	.085 ± .232	-	-	-	-	-	-	-	-
Pairwise	2	.749 ± .079	-	-	-	-	-	-	-	-
Pointwise	5	.088 ± .339	.010 ± .316	.021 ± .287	-	-	-	-	-	-
Pairwise	5	.722 ± .073	.735 ± .068	.734 ± .081	-	-	-	-	-	-
Pointwise	10	.037 ± .291	.028 ± .233	.083 ± .227	.137 ± .285	.067 ± .253	.115 ± .257	-	-	-
Pairwise	10	.704 ± .078	.708 ± .082	.711 ± .075	.675 ± .105	.700 ± .084	.676 ± .087	-	-	-
Pointwise	20	.321 ± .217	.332 ± .224	.327 ± .231	.299 ± .205	.367 ± .203	.304 ± .216	.412 ± .183	-	-
Pairwise	20	.670 ± .090	.673 ± .067	.676 ± .075	.673 ± .072	.683 ± .068	.707 ± .066	.707 ± .050	-	-
Pointwise	50	.587 ± .042	.589 ± .046	.586 ± .047	.585 ± .040	.586 ± .048	.581 ± .046	.579 ± .043	.596 ± .054	-
Pairwise	50	.777 ± .016	.781 ± .014	.776 ± .016	.776 ± .016	.773 ± .019	.774 ± .018	.764 ± .027	.790 ± .018	-
Pointwise	100	.675 ± .013	.677 ± .014	.674 ± .012	.675 ± .012	.676 ± .010	.678 ± .012	.674 ± .018	.654 ± .035	.690 ± .052
Pairwise	100	.762 ± .013	.763 ± .014	.759 ± .014	.765 ± .013	.767 ± .014	.767 ± .012	.782 ± .014	.766 ± .010	.781 ± .010
Pointwise	250	.792 ± .006	.792 ± .005	.794 ± .007	.793 ± .006	.794 ± .007	.794 ± .008	.794 ± .011	.789 ± .013	.782 ± .022
Pairwise	250	.811 ± .004	.811 ± .004	.812 ± .004	.811 ± .004	.811 ± .004	.811 ± .005	.814 ± .005	.816 ± .004	.822 ± .009

Approach	K/d	0	1	3	4	5	7	17	47	97
Pointwise	2	.006 ± .279	-	-	-	-	-	-	-	-
Pairwise	2	.402 ± .210	-	-	-	-	-	-	-	-
Pointwise	5	.015 ± .287	.008 ± .173	.030 ± .139	-	-	-	-	-	-
Pairwise	5	.533 ± .132	.569 ± .124	.543 ± .109	-	-	-	-	-	-
Pointwise	10	.019 ± .243	.004 ± .190	.021 ± .183	.010 ± .155	.023 ± .187	.018 ± .160	-	-	-
Pairwise	10	.595 ± .065	.575 ± .070	.587 ± .062	.588 ± .069	.589 ± .076	.602 ± .069	-	-	-
Pointwise	20	.172 ± .263	.061 ± .211	.076 ± .190	.030 ± .217	.045 ± .206	.048 ± .198	.021 ± .212	-	-
Pairwise	20	.658 ± .051	.646 ± .052	.655 ± .045	.670 ± .050	.663 ± .052	.667 ± .054	.696 ± .044	-	-
Pointwise	50	.350 ± .295	.260 ± .251	.160 ± .233	.153 ± .225	.177 ± .228	.116 ± .254	.110 ± .242	.158 ± .249	-
Pairwise	50	.761 ± .016	.762 ± .016	.760 ± .018	.764 ± .018	.764 ± .017	.762 ± .017	.765 ± .017	.789 ± .014	-
Pointwise	100	.482 ± .249	.423 ± .237	.406 ± .212	.420 ± .217	.342 ± .193	.385 ± .221	.259 ± .272	.270 ± .296	.408 ± .266
Pairwise	100	.803 ± .015	.804 ± .010	.805 ± .009	.805 ± .012	.805 ± .013	.806 ± .011	.807 ± .011	.819 ± .010	.834 ± .008
Pointwise	250	.802 ± .011	.803 ± .012	.802 ± .012	.802 ± .012	.802 ± .012	.805 ± .012	.799 ± .019	.798 ± .023	.789 ± .032
Pairwise	250	.836 ± .008	.836 ± .008	.836 ± .008	.836 ± .009	.836 ± .007	.837 ± .007	.838 ± .006	.839 ± .007	.838 ± .009

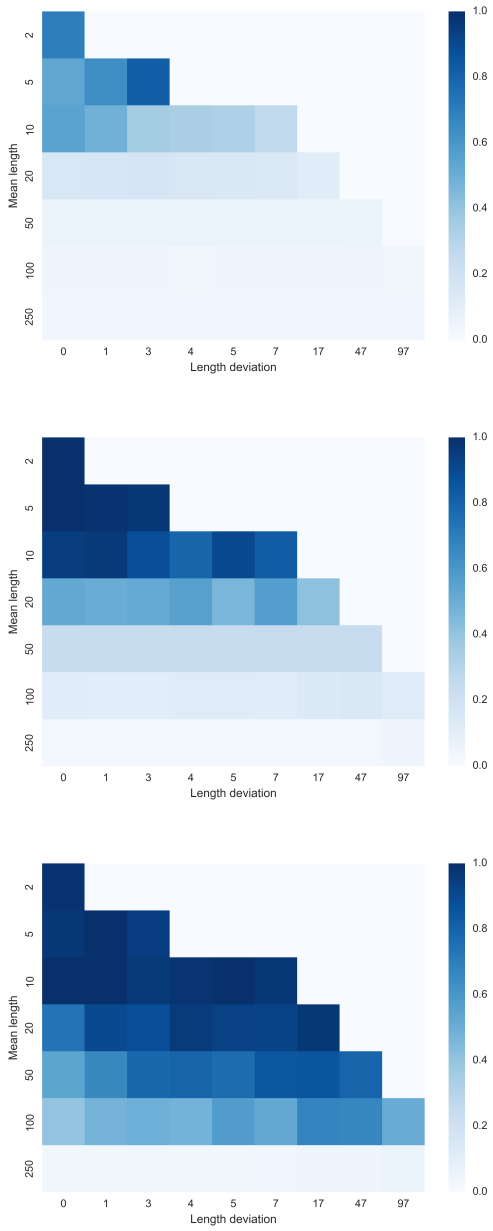


Figure 1. Relative improvement (2) for the *uniform* scenario (top), *top hotels* (middle), and *two groups* (bottom).

5. Conclusion

In summary, the results of our case study convey a picture that to some extent disagrees with previous experimental evaluations of expected rank regression, and which confirms our reservations regarding this approach. ERR seems to be competitive under ideal conditions, namely for sufficiently long rankings that are uniformly distributed across ranks. However, any deviation from these conditions leads to a significant drop in performance. As opposed to this, pairwise ranking shows a much more stable behavior and maintains a consistently strong performance across different experimental settings.

Needless to say, a single case study is necessarily limited in scope. Therefore, the conclusions drawn from the study should of course not be overgeneralized. Instead, we consider them as a starting point for further investigations that are needed to complete the picture and to gain a full understanding of the techniques of pairwise and pointwise ranking.

Acknowledgments

This work has been supported by the German Research Foundation (Deutsche Forschungsgesellschaft, DFG) within the Collaborative Research Centre "On-The-Fly Computing" (CRC 901).

6. References

[1] Fürnkranz, J., Hüllermeier, E., eds. *Preference Learning.* Springer, 2010.

[2] Cohen, W., Schapire, R., Singer, Y., *Learning to order things.* Journal of Artificial Intelligence Research, 1999, **10**(1), pp. 243–270.

[3] Kamishima, T., Kazawa, H., Akaho, S., A survey and empirical comparison of object ranking methods. In Fürnkranz, J., Hüllermeier, E., eds.: *Preference Learning.* Springer 2010 pp. 181–202.

[4] Har-Peled, S., Roth, D., Zimak, D., Constraint classification: a new approach to multiclass classification. In Cesa-Bianchi, N., Numao, M., Reischuk, R., eds.: *Proceedings of the 13th International Conference on Algorithmic Learning Theory*, Springer, 2002, pp. 365–379.

[5] Cheng, W., Hühn, J., Hüllermeier, E., Decision tree and instance-based learning for label ranking. In: *Proceedings of the 26th International Conference on Machine Learning*, Omnipress, 2009, pp. 161–168.

[6] Vembu, S., Gärtner, T., Label ranking: a survey. In Fürnkranz, J., Hüllermeier, E., eds.: *Preference Learning.* Springer 2010 pp. 45–64.

[7] Fürnkranz, J., Hüllermeier, E., Vanderlooy, S., Binary decomposition methods for multipartite ranking. In: *Proc. of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, Springer, 2009, pp. 359–374.

[8] Kamishima, T., Kazawa, H., Akaho, S., Supervised ordering – an empirical survey. In: *Proc. ICDM, 5th IEEE International Conference on Data Mining*, Houston, Texas, 2005, pp. 673–676.

[9] Kamishima, T., Akaho, S., *Supervised ordering by regression combined with Thurstone's model.* Artif. Intell. Rev., 2006, **25**(3), pp. 231–246.

[10] Marden, J., *Analyzing and Modeling Rank Data.* Chapman and Hall, London, New York, 1995.

Schedae Informaticae Vol. 25 (2016): 85–101
doi: 10.4467/20838476SI.16.007.6188

tfml 2017
theoretical foundations
of machine learning, Kraków

Impact of Clustering Parameters on the Efficiency of the Knowledge Mining Process in Rule-based Knowledge Bases

Agnieszka Nowak-Brzezińska[1], Tomasz Rybotycki[2]
[1]University of Silesia, ul. Bankowa 12
40-007 Katowice, Poland,
e-mail: *agnieszka.nowak@us.edu.pl*
[2]IBS PAN, Doctoral Study, ul. Newelska 6
01-447 Warszawa, Polska

Abstract. In this work the subject of the application of clustering as a knowledge extraction method from real-world data is discussed. The authors analyze an influence of different clustering parameters on the quality of the created structure of rules clusters and the efficiency of the knowledge mining process for rules / rules clusters. The goal of the experiments was to measure the impact of clustering parameters on the efficiency of the knowledge mining process in rule-based knowledge bases denoted by the size of the created clusters or the size of the representatives. Some parameters guarantee to produce shorter/longer representatives of the created rules clusters as well as smaller/greater clusters sizes.

Keywords: rule-based knowledge bases, clustering, similarity, visualization

1. Introduction

For the last twenty years, there has been an enormous interest in integrating database and knowledge-based system (KBS) technologies to create an infrastructure for modern advanced applications. The result of it are knowledge bases (KBs) which consist of database systems extended with some kind of knowledge, usually expressed in

Received: 11 December 2016 / Accepted: 30 December 2016

the form of *rules* [1] - logical implications, that can usually be described in form of equation (1):

$$\text{condition}_1 \ \& \ \text{condition}_2 \ \& \ ... \ \& \ \text{condition}_n \implies \text{conclusion}, \tag{1}$$

Such a natural way of knowledge representation makes rules easily understood by experts and knowledge engineers (that are working with KBSs) as well as people not involved in the expert system building (such as data scientists not acquainted with given domain).

1.1. Rules as knowledge representation method

Rules are very specific type of data (knowledge) structure. Usually they are generated from data stored in tabular form (f.e. decision tables). Methods used in order to generate the rules, aims to create so called minimal rules, which means that the rules achieved in this way, have got a short description and cover as many data from the original dataset as possible. Every rule contains two parts: conditional (with at least one premise) and decisional (usually with one conclusion). Sometimes (what makes the analysis more complicated) conclusion of one rule may be a condition in others. In this case it is said that such rules form a chain, and during the inference process they are all processed as a cause and effect chain. Sometimes rules' attributes are weighted therefore the importance of some rules (given as a ordered set of attributes) is higher than others because of difference in weights of their attributes and/or their lengths. What is more, conditional and decisional part of a rule can be also treated differently from each other, f.e. conditional part may have higher priority (greater weights for premises than for a conclusion). All these circumstances make the rules very specific kind of knowledge representation.

1.2. An efficiency of the knowledge mining process

KBs are constantly increasing in volume, thus the knowledge, often stored as a set of rules, is getting progressively more complex and when the rules are not organized into any structure, the KBS isn't as efficient as it could be. There is a growing research interest in searching for methods that could manage large sets of rules using the clustering approach as well as joining and/or reducing rules [2]. Because of many advantages of clustering algorithms [3, 4] it is possible to organize the rules in a smart way. The aim of clustering algorithms is to group the rules into a set of groups (f.e. the hierarchy) of similar rules. To achieve it, some kind of technique, that allows describing similarity between rules, has to be proposed. In the literature there are already a lot of methods of describing similarity between objects, that can be modified

to work with rules as well. When two given rules are said to be the most similar (by given similarity measure in a given step of clustering process), they are meant to be clustered before the others. It influences on the further clustering steps. Similarity measure used to find a pair of rules or groups of rules that are the most similar in a given moment is called an *intra-cluster similarity measure* or *inter-object (inter-rule) similarity measure*. The authors studied different intra-cluster similarity measures [5] and choose the following three measures for experimental validation: Gower's measure [6, 7], Simple Matching Coefficient (SMC) [7] and Jaccard's Index [8] sometimes also called weighted similarity or weighted similarity coefficient [7]. They are further described and anylyzed in Section 2.1.1.

The analysis of the rules' similarity can be based on either premises or conclusions of the rules. In this reasearch rules are divided into a number of groups based on similar premises in order to improve the inference process efficiency [9]. This approach is dictated by the forementioned fact, that conditional parts of the rules are generally longer than their decisional parts, and thus making clustering more complex, accurate and interesting. Moreover, the authors directed their research toward the forward (data driven) inference process where the premises of the rules are the basis of searching. To minimize the number of rules that needs to be read before KBS's anwser to given input is reached, instead of searching within the whole set of rules (as in case of traditional inference processes), only representatives of groups would be compared with the set of facts and/or hypothesis to be proved. The most relevant group of rules is selected and the exhaustive searching is done only within this group. This way, given a set of rules, new knowledge may be derived using a standard forward chaining inference process, which can be described as follows: each cycle of deductions starts with matching the condition part of each rule with known facts. If at least one rule matches the facts asserted into the rule base, it is fired. It's really crucial to find and describe all the factors which influence clustering results and interference efficiency as it'd help in designing or partitioning of KBs in order to maximize KBS's effectiveness.

There is also the other type of cluster similarity measure - so called *inter-cluster* similarity measure. It's used to describe how much each group resembles one another basing on their members (see the details in Section 2.1.2.).

Examining the influence of choosing different similarity measures on the efficiency of clustering algorithms, is the main goal of this research. It's crucial to answer the question if a given similarity measure influences the shape of grouped KB's structure. To have a chance to analyze it, the silimarity measures described in section 2.1. were implemented by the authors. The result of this implementation is the `CluVis` system, described briefly in Section 4. The system allows to examine different similarity measures and methods of hierarchical clustering for any given knowledge base in predefined form described further in [10].

The rest of the paper is organized as follows. We first mention all related efforts in the study of rules clustering in Section 2., where the description of different inter and intra cluster similarity measures were described. Section 3. focuses on two selected visualization algorithms designed for hierarchical strucures. Short description of the authors system `CluVis` can be found in Section 4., whereas Section 5. describes experimental setup, evaluation methodology and the results on public data sets.

2. Rules clustering

Nowadays mankind's knowledge is growing rapidly. People are constantly developing new ways of using this knowledge in practice. As a result KBSs came into being - decision systems, in particular, are prime exmample of that. These systems, often limited to only one domain, tend to store their knowledge in special form. Rule-based knowledge bases (as mentioned in 1.) are most common, because it's easier (for knowledge engineers and experts) to present laws and rudiments of given domain in this shape. Rules stored in knowledge bases are often unorganized as usually there's no reasonable way of doing so, because it's hard (if not impossible) to judge eg. which law or theorem is more important than the other. However, to ensure KBSs' optimal work, it'd be necessary to decrease amount of data that system needs to read to get final conclusion. One way to achive that is proposing some kind of rules partitioning that'd agregate similiar and separate different rules, describing each of such groups with proper description (generalization of it's members) and thus allowing searching through KB from most general groups to specific rule and, ipso facto, successfully decreasing amount of data read during generating KBS's anwser. The same approach may be neccessary during updating KB (deleting redundant rules etc.), where searching through KB is also required. It's possible to find such partitioning using data exploration methods. The most natural approach seems to be clustering.

Clustering is one of the oldest and most common methods of organizing data sets. The goal of clustering is to maximize intra-cluster similarity and minimize inter-cluster similarity, thus creating a partition, where similiar rules are joined into one cluster and rules different from others are singled out. During this unsupervised process similiar objects (according to given similarity measure) are joined into groups thus often decreasing amount of data required to be analyzed in order to extract knowledge from examined data set. Several methods of clustering has been proposed, each of them includes variety of different techniques. Due to forementioned fact selecting proper algorithm is a nontrivial task as there are many factors to be considered. In this work, sets of complex objects known as rules (denoted as Horn's clauses) are being examined. As the authors were aiming to find hierarchy-like partitioning that'd ensure that finding specific rule is achieved through searching through more general groups, hierarchical clustering methods were used. Hierarchical clustering has also one more advantage. It doesn't impose on any special methods of describing clusters similarity and thus can be used for rules clustering without additional modifications. To avoid increasing complexity of research the authors selected one of the most well known algorithms from this popular group of techniques - Classical AHC algorithm - and used to organize several KBs.

2.1. Similarity measures

Measuring similarity or distance between two data points is a core requirement for several data mining and knowledge discovery tasks that involve distance computation.

The notion of similarity or distance for categorical data is not as straightforward as for continuous data. When data consists of objects that agregate both types at once the problem is much more complicated. It is necessary to find a measure, that could deal with this case.

2.1.1. Intra-cluster similarity measures

In this paper, the authors study a variety of similarity measures. Having a set of attributes A and set of sets of their values $V = \bigcup_i V_i$ rules premises and conclusions are build using pairs (a_i, v_j), $a_i \in A, v_j \in V_i$ called descriptors D as a vector of lenght equal to number of attributes in permise and conclusion of given rule, where i-th position denotes the value of i-th attribute for a given premise/conclusion attribute.

In all similarity measures, described in this work, similarity S between two rules r_i and r_j can be denoted as weighted sum of similarities s_k considering k-th common attribute of these rules. This can be written as equation (2):

$$S(r_i, r_j) = \sum_{k:a_k \in (A(r_i) \cap A(r_j))} w_k s_k(r_{ik}, r_{jk}), \tag{2}$$

where $A(r)$ is set of attributes of rule r, w_k is the weight assigned to k-th attribute and r_{ik} and r_{jk} are values of k-th attributes of i-th and j-th rule respectively.

SMC (simple matching coefficient) [7] is the simplest measure of similarity considered in this work. It handles continous attributes the same way it does with categorical attributes, namely (equation (3)):

$$s(r_{ik}, r_{jk}) = \begin{cases} 1 & \text{if } r_{ik} = r_{jk} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

In this case, however, overall similarity of rules S is simple sum, as weight w_k of each attribute is equal to 1. Due to that fact this similarity measure tends to favor rules with more attributes. More advanced form of this measure is Jaccard index [8, 7]. It eliminates forementioned drawback of SMC by setting weight $w_k = \frac{1}{Card(A(r_i) \cup A(r_j))}$, where $Card : V \to \mathbb{N}$ is the cardinality of a set.

Last measure described in this work is Gower's index [6]. This measure is most complicated one, that the authors have used, as it handles categorical data differently from numeriacal data. Similarity considering categorical data is count the same way as in case of Jaccard or SMC. Similarity of continous attributes can be denoted as following (equation (4)):

$$s(r_{ik}, r_{jk}) = \begin{cases} 1 - \frac{|r_{ik} - r_{jk}|}{range(a_k)} & \text{if } range(a_k) \neq 0 \\ 1 & \text{otherwise} \end{cases} \tag{4}$$

where $range(a_k) = max(a_k) - min(a_k)$ is range of k-th common attribute.

All measures described in this subsection were used as a parameter of classical AHC algorithm in order to examine influence they have on resultant structure of clustering. It was shown that f.e. some of them tends to generate structures with larger number of ungrouped rules and that different inter-object similarity measures influences average length of cluster representative, thus allowing one to consider KB partitioning basing on cluster representatives length.

2.1.2. Inter-cluster similarity measures

Apart from the usual choice of similarity functions, linkage criterion must be selected (since a cluster consists of multiple objects, there are multiple candidates to compute the distance to). There are many possible ways of describing inter-cluster similarity. The most popular among them are known as Single Link (SL), Complete Link (CoL), Average Link (AL) and Centroid Link (CL) [11, 3, 4].

Single link is the most head-on approach, as it describes similarity of the clusters basing on their most similar objects, thus in case of rules-clustering one may say, that clusters of rules are as similar as their most common two rules. This method is known to cause undesireable occurence called cluster chaining, wherein long clusters are being created. In general, however, it's not proper partitioning for given dataset and thus another method should be proposed.

The most similiar to SL is method called Complete Link. Both of these methods returns similarity of single pair of rules as similarity of two clusters, however, in case of complete link, clusters are only as similar as their two most disrinct rules. The values of CoL are obviously lower (on average) than in case of SL. It is known in the literature (and also had been shown during experiments in this work) that CoL tends to generate partitionings wherein there's lower number of small groups and bigger number of larger ones. Both of forementioned methods share a common drawback, namely: they are sensitive to noisy data. The reason for that is that they both base their result on single pair of rules instead of considering whole contents of the clusters.

Another inter-cluster similarity measure that was used in this work is Average Link. Let C_i, C_j be two clusters of rules. Then similarity between C_i and C_j can be defined by equation (5):

$$AL(C_i, C_j) = \frac{\sum_{r_i \in C_i} \sum_{r_j \in C_j} S(r_i, r_j)}{\overline{C_i} * \overline{C_j}} \tag{5}$$

In other words AL is described as mean similarity of all rules inside examined clusters. It's sensitive neither to noisy data nor to cluster chaining as it considers all rules that are inside given clusters.

Another measure that considers all rules within given clusters is centroid link. This one however, instead of considering similarity between rules that are inside examined clusters, considers only similarity between two virtual objects, called centroids of clusters. Usually centroids are described as geometrical center of the cluster, however, in case of rules, defining geometrical center of cluster is non-trivial task, thus

representative of this cluster was used instead. It was dictated by assumption that representative of the cluster (further described in subsection 2.2.) is meant to be the most general rule in the cluster - one may say most ,, centered " one. Let $c(C_i)$ be centroid of i-th cluster. Then similarity between clusters C_i and C_j can simply be defined as $CL(C_i, C_j) = S(c(C_i), c(C_j))$.

In previous authors' researches it was noticed that the method of creating the representatives are equally important with the clustering algorithm and similarity measures used to clustering rules. In this work, further researches on this matter were conducted.

2.2. Cluster's representative

It is very important for data to be presented in the most friendly way. Sole visualization of clustering (described further in 3.) is not enough, as it would only reduce the whole pattern discovery to examining an accumulation of shapes, thus some kind of symbolic description has to be proposed. Cluster representatives are the proposed solution for this issue. There are many methods of creating representatives. In this work representative aims to be an average rule of the cluster, basing on cluster's content. Its creation algorithm can be described as follows. Having as input data: cluster C, and a threshold value t [%], find in the cluster's attributes set A only these attributes that can be found in t rules and put them in set A'. Then for each attribute $a \in A'$ check if a is symbolic or numeric. If it is symbolic then count modal value, if numeric - average value and return the attribute-value pairs from A' as cluster's representative. The representative created this way contains attributes that are the most important for the whole cluster (they are common enough). This way the examined group is well described by the minimal number of variables [1]. It should be possible to characterize the clusters using a small number of variables (the number of attributes attained by this method strongly depends on selected threshold value). It is very important to examine the quality of created clusters and to generate the well-formed descriptions for them, what is difficult especially when the objects of clustering are rules.

2.3. Clustering algorithm

As mentioned in section 2. Classical AHC algorithm was used to for rules clustering in this work. It's essential to point out that during each iteration step it joins only two most similiar clusters (not all clusters with similarity exceeding given threshold). Decision of choosing this algorithm over general AHC was dictated by the fact, that selecting rational threshold value, especially when grouping complex objects such as rules, would require deep analysis of each examined data set.

[1] However the authors see the necessity to analyze the more methods for creating clusters' representatives and their influence on the resultant structure efficiency.

Clustering algorithm used in this work agregates all the elements forementioned in this section. As AHC algorithm doesn't specify neither inter-object nor inter-cluster similarity measure, they have to be precised as algorithm's parameters. In this work all measures specified in 2.1.1. and 2.1.2. were used in all possible configurations. During each merging step created cluster is given a description, in form of representative, generated in a way described in 2.2.. It's worth mentioning that representative is created basing solely on rules within given cluster, not on representative of cluster figuring higher in hierarchy.

The grouping is complete after selected number of groups is reached. The number of produced clusters is in range from 1 to N, where N is the number of rules in examined knowledge base. After grouping is finished resultant structure can be visualized using algorithms described further in 3..

3. Knowledge base visualization

The most common way of clustering presentation is tree like structure called dendrogram, however, it has some fatal flaws that makes it inadequate for researches such as presented in this work. Among others, the most vital issue is that it isn't suited for representing large clusterings, as it quickly becomes less readable. This problem concerns many visualization algorithms, but some of them loses their readability much slower e.g. treemaps. In this work two treemap algorithms were used - rectangular treemap [12] and circular treemap [13]. The differences between the two are very distinct for each of them base on different geometrical shapes and has different methods of deploying them. In this work classical slice-and-dice deployment method was used for rectangles as proposed in [12] and method described in [10] was used for circles.

Figure 1 presents the case of using the `CluVis` to cluster 42 rules in a given knowledge base with 70 attributes into 10 groups. Clusters are presented graphically using Circular Treemap (as selected visualization algorithm) and classical AHC (as clustering algorithm) with Gower measure for inter-cluster similarity measure and (CL) Complete link as intra cluster similarity measure. For this particular case the biggest cluster's size is 18 rules (which is 42% of all rules in KB), (denoted as $J33$) with the representative's length equal to 67 descriptors (which is also the maximum length for representatives in general). The other clusters contains the following number of rules: $4, 4, 6, 1, 1, 1, 4, 2, 1$.

Visualizations generated using these algorithms, without any additional features, would only be accumulations of shapes. In some cases such solution would seem sufficient, however, in general, especially when exploring large sets of rules, it is confusing or chaotic. To make exploration of large KBs easier each visualization is responsive. Each shape stores information about cluster it represents. Moreover there is possibility of future examining objects agregated in cluster by selecting it. To ensure that desired cluster is selected, active shape is highlighted. To make visualization even more readable, colors (of the rainbow) were used to mark procentage size of the cluster. Colors and their order were selected in such way to make it easy to remember.

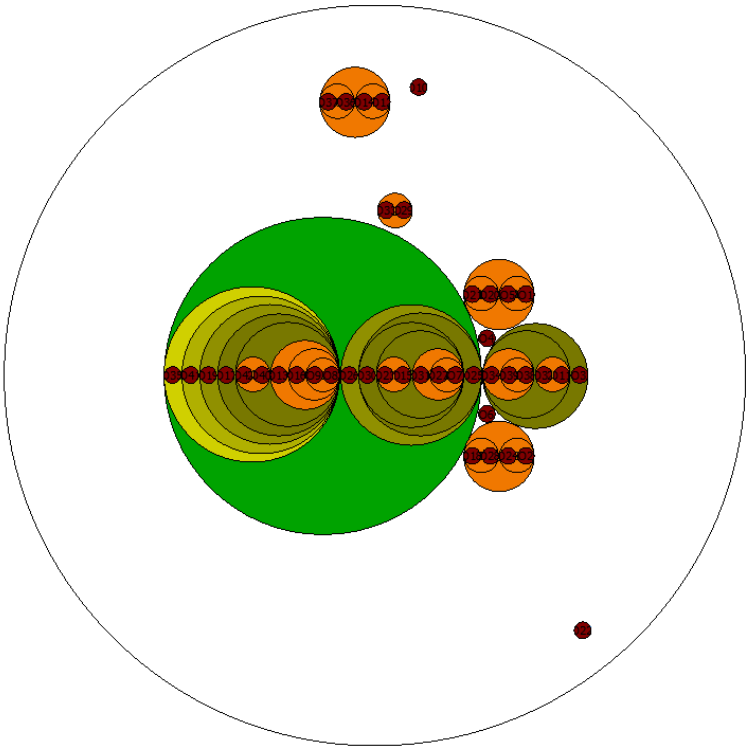


Figure 1. Sample treemap visualization.

4. CluVis

`CluVis` [10] is an application designed to group sets of rules generated by *RSES* [14] and visualizing them using selected treemap methods. It is first application capable of working on raw *KB*s as generated from *RSES*. It was successfully used in previous researches to group and visualize medical knowledge bases generated from artificial data sets available on [15] as well as one generated from real medical data [11]. Moreover, it aggregates functionalities of both clustering and visualization software, making it universal tool for exploring *KB*s. Along its main functionalities (many of which can be seen in figure 2), described in more detail in sections 2. and 3. `CluVis` is capable of generating reports of grouping (to `txt` or special `xml` files which can be opened in such applications as e.g. `Libre Calc`) containing detailed information about each obtained cluster and about clustering in general (number of nodes, min/max/avg representative length...). It's also possible to save generated visualization to `png` file as well as to find best clustering (according to implemented cluster validation indexes - *MDI* and *MDBI* - not discussed in this article). `CluVis` is an open source application written in C++11 using QT graphic libraries. It's available in english and polish and its source code can be downloaded from https://github.com/Tomev/CluVis.

Every XML report file, generated using the authors application, contains the following informations: index of an experiment (`Index`), the name of the knowledge base file (`Name of the base`), the number of attributes (`Attributes number`), of objects (`Objects number`), of nodes in the created hierarchy (`Nodes number`), of created clusters (`Clusters number`). It also says what was the algorithm used to visualize the groups of rules (X or Y), which clustering algorithm it uses, what was the intra-cluster and inter-cluster similarity measure, what part of the rule (conditional or/and decisional) was analyzed as well as the names of the biggest and the smallest cluster (`The biggest cluster` / `The smallest cluster`). Moreover it gives the values for: `Coverage sum`,
`Biggest cluster size`, `Biggest representative length`, `Ungrouped objects`,
`Biggest representative size`, `Smallest representative size`,
`Average representative size`, `MDI\verb`, `MDBI`. It also contains the details data of each cluster, like `Index`, `Cluster's name`, `Cluster's size`,
`Objects percent in cluster`, `Cluster's nodes number`,
`Cluster's nodes percent`, `Cluster's coverage`, `Cluster's coverage percent`,
`Cluster's representative`, `Cluster's representative length`.

Then an example of the cluster representative (which contains 25 of descriptors) may looks like:

```
(history_ringing=f)&(history_fullness=f)&(m_m_sn_gt_500=f)&
(m_s_sn_gt_2k=f)&(boneAbnormal=f)&(history_buzzing=f)&(m_m_gt_2k=f)&
(m_gt_1k=f)&(history_dizziness=f)&(m_s_sn_gt_1k=f)&(airBoneGap=f)&
(history_fluctuating=f)&(m_s_gt_500=f)&(age_gt_60=t)&(air=mild)&
(history_noise=t)&(m_p_sn_gt_2k=f)&(m_m_sn_gt_2k=f)&(history_heredity=f)&
(history_recruitment=f)&(late_wave_poor=f)&(m_at_2k=f)&(m_cond_lt_1k=f)&
(bser=MISSING)&(m_s_sn_gt_3k=f)=>(class=cochlear_age_and_noise)
```

Automatically created KBs (e.g. creating rules from dataset using $LEM2$ algorithm) have a chance to contain some undesired rules (like ones that would never be activated or are simply redundant). It is essential to maintain simplicity of KBs thus some method of eliminating these rules is required. As $CluVis$ is used to transform unorganized KBs into organized ones, presented in form of responsive visualization that enhances readability of KB, it's a perfect tool for this task.

The process of organizing knowledge bases in `CluVis` is as follows. After importing KB into the application and selecting clustering parameters grouping can be performed. First selected file is scanned to see weather it has proper format. In the next step data about attributes is gathered (such information as frequency of each value of each categorical attribute, maximal and minimal values of continous attributes...) as they are used in some similarity measures. Then rules are transformed from lines of text into hashmaps, which are basically vectors [2], which are stored as a singular clusters. Then similarity matrix (which is triangle matrix holding information about similarity between i-th and j-th cluster) is built using similarity measure selected during first phase. Then two the most similiar clusters are joined, and the

[2] The i-th variable value is accessed by it's name in map, not by it's index

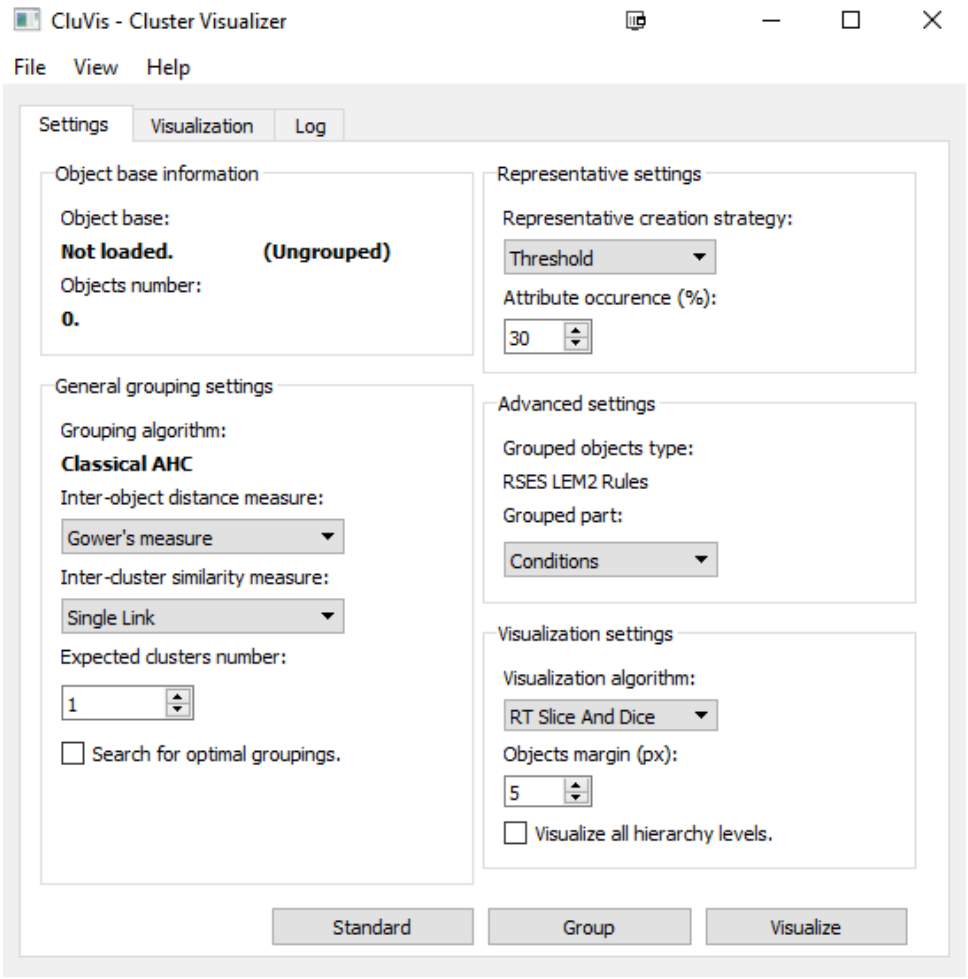


Figure 2. CluVis's GUI.

similarity matrix is updated. During merging of two smaller clusters, their representative is calculated. There may be many possible stop conditions for AHC (e.g. end when similarity of most similiar objects is equal to 0). `CluVis` ends clustering after given (as clustering parameter) number of clusters is reached. The grouping ends when stop contidion is satisfied (selected number of clusters is formed). As soon as grouping is complete visualization can be generated.

Graphical representation of clustering can be performed in two ways - with fully hierarchical view, or not. Visualization is responsive meaning that selecting a cluster and clicking it with right mouse button will generate new visualization representing internal structure of selected cluster. Considering that during each step of used AHC algorithm only two objects are merged usually structure of internals is trivial (large cluster + single object), however, it's nonetheless a method of exploring KB by diving deeper into it. To ensure proper direction of diving, one may generate a report from

each cluster on current visualization and select one that seems most appropriate. Sample visualization was presented on figure 1.

5. Experiments

In this section, an experimental evaluation of 3 similarity measures and 4 clustering methods on 7 different KBs [15] is presented. Decision rules were generated from the original data using `RSES` software and $LEM2$ algorithm [14]. The smallest number of attributes was 5, the greatest 280. The smallest number of rules was 42, the greatest 490.

Table 1. Characteristics of the experimental data.

	AttrN	RulesN	ClustersN	UngroupedN
Arythmia	280	154	12,5±2,52	5,78±5,68
Audiology	70	42	6,92±3,02	3,56±2,85
Autos	26	60	7,89±2,25	3,57±2,92
Balance	5	278	19±9,00	7,46±8,70
Breast	10	125	11±1,01	5,08±3,35
Diab	9	483	28,74±19,13	11,75±13,75
diabetes	9	490	29,5±19,64	13,58±14,87

Table 2. Data gathered during experiments.

	BRS	ARS	WARS	BRL	BCS
Arythmia	151,9±4,6	133,4±11,6	2,1±0,2	147,4±1,5	111,5±41,7
Audiology	67,0±0,4	49,7±1,2	1,5±0,4	66,8±0,5	29,8±7,8
Autos	11,9±1,9	8,6±1,7	3,2±0,6	10,7±0,5	38,3±14,0
Balance	4±0	3,5±0,4	1,4±0,2	4±0	180,2±93,5
Breast	9±0	7,1±1,1	1,4±0,2	9±0	7,6±32,8
Diab	5,4±0,5	3,4±0,8	2,9±0,7	4,8±0,6	314,8±137,2
diabetes	5,6±0,7	3,3±0,8	2,9±0,7	4,9±0,3	335,2±140,7

All the details of analyzed datasets are included in table 1 and 2. The meaning of the columns in table 1 and 2 is following:

AttrN - number of different attributes occuring in permises or conclusions of rules in given knowledge base.

RulesN - number of rules in examined knowledge base.

ClustersN - number of nodes in dendrogram representing resultant structure.

UngroupedN - number of singular clusters in resultant structure of grouping.

BRS - Biggest representative size - number of descriptors used to describe longest representative.

ARS - Average representative size - average number of descriptors used to describe cluster's representatives.

WARS - Weighted Average representative size (AttrNumber) - division of average number of descriptors used to describe cluster's representative in give data set and number of attributes in this data set.

BRL - Biggest representative length - number of descriptors in biggest cluster's representative.

BCS - Biggest cluster size - number of rules in the cluster that contains the most of them.

The performance of different similarity measures was evaluated in the context of knowledge mining using informations like: the number of rules clusters (CN - Clusters number), the number of ungrouped rules (U - Ungrouped objects), the sizes of the biggest cluster ($BiggCluS$ - Biggest cluster size) as well as its representative ($BiggRepS$ - Biggest representative size) and the representative the most specific ($BiggRepL$ - Biggest representative length). More specific means more detailed, containing a higher number of descriptors.

The optimal structure of KBs with rules clusters should contain the well separated groups of rules, and the number of such groups should not be too high. Moreover, the number of ungrouped rules should be minimal. Creating an optimal description of each cluster (representative) is very important because they are used further to select a proper group (and reject all the others) in inference process, in order to mine knowledge hidden in rules (by accepting the conclusion of the given rule as a true fact). The experimental results verifies the initial hypotheses about the inter and intra cluster similarity measures. As can be seen from Tables 3 and 4 no single measure is always superior or inferior. This is obvious since each KB has different characteristics (different number of attributes and/or rules) as well as different types of attributes. The use of some measures however, guarantees achieving more general or more specific representatives for created rules clusters. There are some pairs of measures that exhibit complementary performance, i.e. one performs well where the others perform poorly and vice-versa.

Table 3. Influence of inter-cluster similarity measures on respective values.

	CN	BiggCluS	BiggRepL	U	BiggRepS
Gower	16,6±14,1	157,2±147,4	35,7±51,2	8,1±9,9	36,4±52,3
SMC	16,4±14,3	155,6±143,6	35,4±50,8	5,6±6,7	36,3±52,4
WSMC	22,1±16,5	211,5±152,8	5,7±2,0	6,1±9,1	6,0±1,9

Table 3 and figure 3 show that choosing different intra-cluster similarity measures does not influence the overall efficiency (the exception is Jaccard's index).

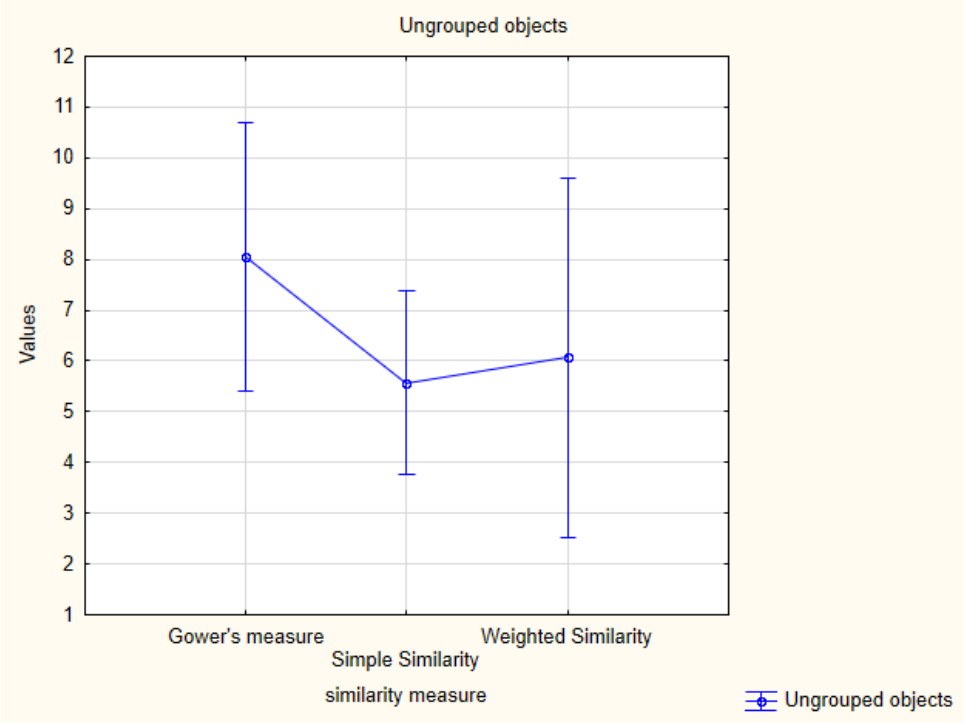


Figure 3. Ungrouped rules - interval plot for inter-cluster similarity measures

Table 4 (with figure 4) however shows that the size of the biggest cluster and the number of ungruped rules depends on the inter-cluster similarity measures. The experiments confirmed that the SL method produces straggling clusters, called chaining, where clusters may be forced together due to single elements being close to each other, even though many of the elements in each cluster may be very distant from each other. CoL, on the other hand, tends to find compact clusters.

Table 4. Influence of intra-cluster similarity measures on respective values.

	SL	CL	AL	CoL
CN	16,5±14,2	16,5±14,1	16,6±14,1	16,6±14,1
U	12,1±11,8	2,1±3,1	4,5±5,2	10,4±10,9
BigCluS	213,5±166,2	84,0±89,8	157,2±139,8	167,8±142,6
BigRepL	35,4±50,4	35,1±50,6	35,4±50,4	35,4±50,5
BigRepS	36,0±51,8	36,4±51,5	36,5±51,5	36,5±52,2

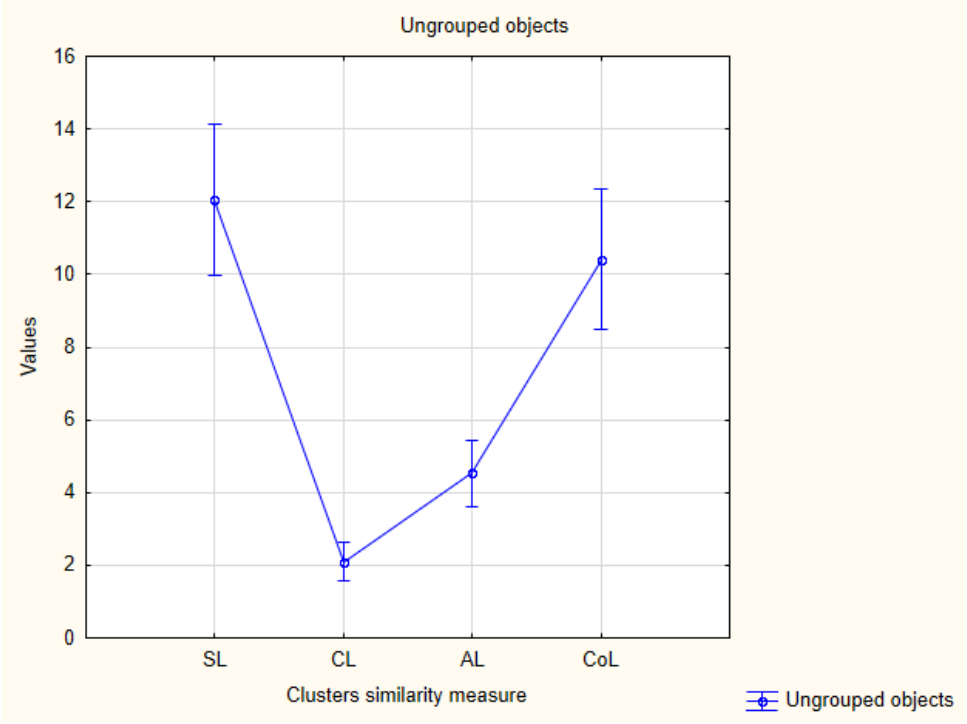


Figure 4. Ungrouped rules - interval plot for intra-cluster similarity measures

6. Summary

This article presents how exploration of complex KBs can be performed using clustering and visualization of rules clusters and presents the application of clustering as a knowledge extraction method from real-world data. Clustering a large set of objects (rules in this case) is not enough when exploring such an enormous amount of data in order to find some hidden knowledge in it. The extraction of valuable knowledge from large data sets can be difficult or even impossible. Modularization of KBs (by clustering) helps to manage the domain knowledge stored in systems using the described method of knowledge representation because it divides rules into groups of similar forms, context, etc. The authors analyze an influence of different clustering parameters on the quality of created structure of rules clusters and the efficiency of the knowledge mining process for rules / rules clusters. In the course of the experiments, three different similarity measures and four clustering measures have been examined in order to verify their impact on the size of the created clusters and the size of the representatives. The experiments have revealed that there is a corelation between the parameters used in the clustering process and future efficiency levels of the knowledge mined from such structures: some parameters guarantee to produce shorter/longer representatives of the created rules clusters as well as smaller/greater clusters sizes.

The authors propose to use clusters of rules and visualize them using treemap algorithms and hope that this two-phase way of rules representation allows the domain experts to explore the knowledge hidden in these rules quicker and more efficiently than before. In the future, the authors plan to extend the software's functionality, especially in the context of parameters used in clustering and visualizing procedures, as well as importing other types of data sources. It would be easier then to support human experts in their everyday work by using the created software (`CluVis`) in work with many expert systems.

7. References

[1] Mulawka, J.J., *Systemy Ekspertowe.* Wydawnictwo Naukowo-Techniczne, Warszawa, 1996.

[2] Latkowski, R., Mikołajczyk, M., Data decomposition and decision rule joining for classification of data with missing values. In: *Rough Sets and Current Trends in Computing.* vol. 3066 of *Lecture Notes in Computer Science.*, Springer Berlin Heidelberg, 2004, pp. 254–263.

[3] Morzy, T., *Eksploracja danych. Metody i algorytmy.* Wydawnictwo Naukowe PWN, 2013.

[4] Wierzchoń, S.T., Kłopotek, M.A., *Algorithms of Cluster Analysis.* Wydawnictwo IPI PAN, Warszawa, 2015.

[5] S. Boriah, V. Chandola, V.K., Similarity measures for categorical data: A comparative evaluation. In Chid Apte, Haesun Park, K.W., Zaki, M.J., eds.: *Proceedings of the 2008 SIAM International Conference on Data Minning*, Society for Industrial and Applied Mathematics, 2008, pp. 243–254.

[6] Gower, J.C., *A general coefficient of similarity and some of its properties.* Biometrics, 1971, **27**, pp. 857–871.

[7] T. Jach, A.N.B., Wnioskowanie w systemach z wiedz niepewn. In: *Studia Informatica.* vol. 32 No 2A. Wydawnictwo Politechniki Ślskiej 2011 pp. 377–389.

[8] Jaccard, P., *tude comparative de la distribution florale dans une portion des alpes et des jura.* Bulletin de la Socit Vaudoise des Sciences Naturelles, 1901, **37**, pp. 547–579.

[9] Nowak-Brzezińska, A., *Mining rule-based knowledge bases inspired by rough set theory.* December 2016, ,, **148**(no. 1–2), pp. 35–50.

[10] Rybotycki, T., *Visualization of hierarchical structures in rule-based knowledge bases.* March 2015, ,.

[11] Nowak-Brzezińska, A., Rybotycki, T., *Visualization of medical rule-based knowledge bases.* Journal of Medical Informatics & Technologies, 2015, **24**, pp. 91–98.

[12] Shneiderman, B., *Tree visualization with tree-maps: 2-d space-filling approach.* January 1992, ,, **11**, pp. 92–99.

[13] Wetzel, K., *Pebbles - using circular treemaps to visualize disk usage.* http://lip.sourceforge.net/ctreemap.html, 2004.

[14] Bazan, J.G., Szczuka, M.S., Wroblewski, J., A new version of rough set exploration system. In: *Rough Sets and Current Trends in Computing.* vol. 2475 of *Lecture Notes in Computer Science.*, Springer Berlin Heidelberg, 2002, pp. 397–404.

[15] Lichman, M., *Machine learning repository.* http://archive.ics.uci.edu/ml, 2013 Accessed in October 2016.

Schedae Informaticae Vol. 25 (2016): 103–115
doi: 10.4467/20838476SI.16.008.6189

tfml 2017
theoretical foundations
of machine learning, Kraków

Towards Learning Word Representation

Magdalena Wiercioch
Faculty of Mathematics and Computer Science
Lojasiewicza 6, 30-348 Kraków
e-mail: *magdalena.wiercioch@ii.uj.edu.pl*

Abstract. Continuous vector representations, as a distributed representations for words have gained a lot of attention in Natural Language Processing (NLP) field. Although they are considered as valuable methods to model both semantic and syntactic features, they still may be improved. For instance, the open issue seems to be to develop different strategies to introduce the knowledge about the morphology of words. It is a core point in case of either dense languages where many rare words appear and texts which have numerous metaphors or similies. In this paper, we extend a recent approach to represent word information. The underlying idea of our technique is to present a word in form of a bag of syllable and letter n-grams. More specifically, we provide a vector representation for each extracted syllable-based and letter-based n-gram, and perform concatenation. Moreover, in contrast to the previous method, we accept n-grams of varied length n. Further various experiments, like tasks-word similarity ranking or sentiment analysis report our method is competitive with respect to other state-of-the-art techniques and takes a step toward more informative word representation construction.

Keywords: representation learning, n-gram model, NLP

1. Introduction

Continuous word representations (embeddings or distributed representations) are found useful for many Natural Language Processing problems such as information

Received: 11 December 2016 / Accepted: 30 December 2016

retrieval or character recognition [1, 2]. Since their quality is strictly connected with aspects of specific language that is being analyzed, each explored issue in this field may also lead to improvement of the particular task where given representation is applied.

Various attempts have been made to investigate learning continuous representations of words in Natural Language Processing [3, 4]. Most of the earliest approaches for learning continuous vectors are based on latent semantic derivations [5, 6]. In particular, its subdomain called distributional semantics where analyzing relationships between a set of documents are considered have been studied extensively in vast majority of papers [7]. In last years neural network researchers have focused on this field [8, 9]. The common drawback of these techniques is the fact they associate a completely distinct vector to each word of the vocabulary. In consequence, the word characteristic information is lost. Take for instance some of dense (highly inflected) languages, i.e. Serbo-Romanian, Romanian which create a challenge for researchers since these languages are seen by linguistics as richly inflected [10]. What is more, although English is not considered as complex, it may be demanding to learn satisfactory representation for corpuses with many rhetorical devices.
On the other hand, the idea of applying more detailed information connected with a given word to a model was presented a few years ago. One of the first approaches to learn representations using fragments of words was character fourgrams-based method introduced by Schütze [4]. In 2003 Bilmes and Kirchhoff investigated factored language models, where a word is viewed as a vector of k factors, such as stems, morphological classes, data-driven clusters [11]. Also, several approaches which rely on a morphological decomposition have been proposed [12, 13]. There is a series of papers which describe models built using recurrent neural networks [14, 15]. Yet another class of methods makes use of convolutional neural networks working on characters. Let us give just a few examples of usage: text classification [16], part-of-speech tagging [17, 18], language modeling [19], sentiment analysis [20] or text normalization [21]. Recently, the concept of using subwords to form a representation appeared [22, 23]. Another work [24] suggests to guide word-embeddings with morphologically annotated data and shows achievement using German in a case study. Also, many papers study syllable-based n-gram methods to model language [25, 26].

In this work, we explore another way to learn word representation using combined character and syllable-level approach. Inspired by the recent works - on continuous bag-of-word model by Mikolov et al. [27] and on using subword information by Bojanowski et al. [28], we show that combining multiple n-grams types enables to capture more word-specific features. This paper is an extension to words vector model proposed by Bojanowski et al. [28]. *Our main goal is to check how extra added syllable information to subword vector representation changes the overall reliability of the model.* What is more, the previous paper uses a very simple scheme where only n-grams of length between 3 and 6 are explored. We does not make such limitations and in consequence our model is able to distinguish short words as well. In order to evaluate our approach, we compare several types of continuous representations, including those made available by other researchers. The evaluation tasks - word similarity ranking analogies and analogy analysis prove the method to be valuable. We achieve improvement for Romanian corpus, too.

The most vital advantage of the proposed model is an attempt to describe word

more precisely. For instance, according to our approach "in" has a different position (representation) in word space if it appears as a word itself, and another two locations in case it appears in two independent fragments of another word, e.g. "painting".

2. Model architecture

In this section, we present model to learn specific representation that takes words fragments into account. The proposed representation is an extension of the idea introduced by Bojanowski et al. [28]. Since the model demonstrated by Bojanowski itself is derived from continuous Skip-gram (SG) model introduced by Mikolov et al. in 2013 [8], we first explain how SG works.
Generally, training phase of the Skip-gram model aims at finding word representation that is useful for predicting the surrounding words in a corpus. Let us denote $W = \{w_1, w_2, \ldots, w_S\}$ as the sequence of training words - vocabulary, S - size of vocabulary. The goal of the Skip-gram model is to maximize the average log probability

$$l(W) = \sum_{t=1}^{S} \sum_{c \in C_t} \log p(w_c|w_t),$$

where C_t refers to the context, i.e. the words which surround w_t.
The probability of observing a context word w_c given w_t is parametrized using the word vectors. Given a scoring function s, which maps pairs of (word, context) to value in $\mathbb{R}$, a possible choice to define the probability of a context word is the softmax.

In a basic form the probability of the output context word $Context$ having input $Word$ is defined using the softmax function

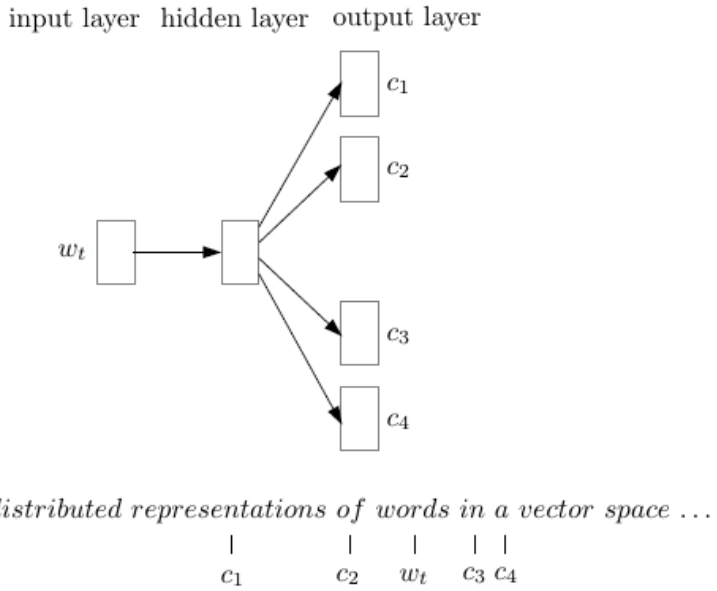
$$p(Context|Word) = y_c = \frac{e^{w_c^\top w_t}}{\sum_{j=1}^{S} e^{w_j^\top w_t}},$$

where w_c, w_t, w_j are vector representations of words and y_c is the output of the c-th neuron of the output layer. In practice this formula is not used because of the computational costs. However, an efficient alternative to softmax is Negative sampling, a simple version of Noise Contrastive Estimation (NCE) [29]. While NCE requires from the model to differentiate data from noise by means of logistic regression, Negative Sampling aims only at obtaining high quality representation. Thus, in terms of neural probabilistic language model one may formulate the conditional distribution corresponding to context word c

$$P_\theta^c = \frac{e^{s_\theta(w_t^\top w_c)}}{\sum_{j=1}^{S} e^{s_\theta(w_j^\top w_c)}},$$

where $s_\theta()$ is called scoring function that assesses how the word w_t is compatible with the context w_c. The parametrization for the scoring function is done by taking the

Figure 1. The Skip-gram model architecture with 4 context words considered. The learnt representation enables to predict surrounding words given the current input word "words".



scalar product between word and context embeddings: $s(Word, Context) = w_t^\top w_c$[1].

Since the rest of the work uses the concept of letter n-grams and syllable n-grams, they shall be explained.

Letter n-grams

In our paper a letter n-gram is a contiguous sequence of n characters from a given string. As explained in Subsection "Fragmentation model", we use n-grams of several different lengths. In order to distinguish the beginning of a word and the ending of word we append blank spaces to the beginning and the end of the word. Let's take word 'TEST' for example. Note that here the underscore (_) represents the blank space. The following n-grams should be expected.

$$bigrams : \_T, TE, ES, ST, T\_$$

$$trigrams : \_TE, TES, EST, ST\_, T\_\_$$

$$4-grams : \_TES, TEST, EST\_, ST\_\_, T\_\_\_$$

Syllable n-grams

The syllable n-gram is seen as a contiguous sequence of n syllables in a given string. In general, the task of defining the syllable raises some controversy [30, 31]. This work employs the procedure proposed by Daelemans et al. in [32].

[1] Both w_c and w_t are vector representations in $\mathbb{R}^d$.

Fragmentation model

We notice two possible extensions of Bojanowski et al. approach [28]. Firstly, as the authors suggest, the Skip-gram model ignores the internal structure of words. So, they associate a vector representation z_g to each n-gram g. However, we claim it may be insufficient for short and rare words. In this section, we thus propose to extend such a representation by taking syllable n-grams into consideration. Given a word w, let us denote by $G_w = \{1, \ldots, G\}$ the set of letter n-grams which appear in w (as Bojanowski et al. done). Similarly, let $H_w = \{1, \ldots, H\}$ to be the set of syllable n-grams which appear in w. Now, we associate a vector representation z_g to each letter n-gram g and a vector representation z_h to each syllable n-gram h. The new word representation is considered as the direct concatenation of the two vector representations of its n-grams (letter and syllables):

$$z_{new} = [z_g, z_h].$$

In consequence, the scoring function is

$$s(w, c) = \sum_{new \in G_w \cup H_w} z_{new}^{\top} v_c.$$

Secondly, in the model demonstrated by Bojanowski n-grams of length k are only considered, where $3 \leq k \leq 6$. Our analysis show it may negatively affect the final representation reliability. Thus, the upgraded model makes use of n-grams of varied length n.

3. Experiments

Table 1. Spearman's correlation coefficient for the word similarity task.

dataset	RNNLM	NCE	CBoW	Sg	Ft	our
WS353 (en)	0.42	0.45	0.48	0.47	**0.5**	**0.5**
SimVerb-3500 (en)	0.44	0.46	0.44	**0.47**	**0.47**	**0.47**
Sim999 (en)	0.44	0.45	0.45	**0.46**	0.45	0.45
RG65 (en)	0.39	0.4	0.43	0.46	0.46	**0.47**
SGS130 (en)	0.45	0.48	0.5	0.49	**0.5**	**0.5**
YP130 (en)	0.43	0.45	0.44	0.47	**0.48**	**0.48**
Gur30 (ge)	0.45	0.46	0.49	**0.51**	**0.51**	**0.51**
Gur65 (ge)	0.45	0.47	0.52	0.54	0.54	**0.55**
ZG222 (ge)	0.5	0.53	0.53	0.55	**0.56**	**0.56**
RO353 (ro)	0.51	0.55	0.57	0.59	0.59	**0.61**

We conducted a series of experiments to compare the performance of our approach with several strong baseline representations learned on a fixed dataset on different

Table 2. Semantic analogies task results. The accuracy specified as %.

dataset	RNNLM	NCE	CBoW	Sg	Ft	our
WS353 (en)	15.3	24.2	0.23.8	**28**	27.5	27.5
SimVerb-3500 (en)	20.1	26.7	30.6	**34.5**	**34.5**	34
Sim999 (en)	18.3	21.2	**29.8**	24.3	24.8	24.8
RG65 (en)	29.7	35.2	39.1	**42**	**42**	**42**
SGS130 (en)	35.2	41.3	47	**56.1**	**56.1**	**56.1**
YP130 (en)	46.4	42.6	43.6	**56.3**	**56.3**	**56.3**
Gur30 (ge)	37.2	61.2	38.7	**46.7**	**46.7**	**46.7**
Gur65 (ge)	39.8	34.2	44.7	**46.9**	46	46
ZG222 (ge)	41.7	36.2	55.3	**52.6**	**52.6**	52.2
RO353 (ro)	43.9	50	46.6	**60.4**	60.1	60.1

Table 3. Syntactic analogies task results. The accuracy specified as %.

dataset	RNNLM	NCE	CBoW	Sg	Ft	our
WS353 (en)	24.7	30.2	33.5	**40.9**	40.2	40.2
SimVerb-3500 (en)	31.6	33.9	37.2	**52**	**52**	**52**
Sim999 (en)	26	32	**55**	49.8	49.3	49.5
RG65 (en)	35.6	40.2	40.7	**48.9**	**48.9**	**48.9**
SGS130 (en)	38.4	**59**	43.2	49.6	49.6	49.6
YP130 (en)	32.3	37.8	45.8	**50.3**	**50.3**	**50.3**
Gur30 (ge)	30.1	35.2	40.9	**49.3**	**49.3**	**49.3**
Gur65 (ge)	24	35.7	47.3	**62.5**	**62.5**	**62.5**
ZG222 (ge)	38.7	45.3	56.9	**67.2**	**67.2**	67.1
RO353 (ro)	30.6	41.7	**59.2**	53.1	53.1	53.1

tasks. In our experiments we used benchmarks of three languages, i.e. English, German and Romanian. For English, we evaluated word vectors on the following datasets: WS353 [33], SimVerb-3500 [34], Sim999 [35], RG65 [36], SGS130 [37], YP130 [38]. For German, the models were compared on datasets: Gur30, Gur65 [39], ZG222 [40]. For Romanian, the translated version of WS353 was used [41] (RO353). The data contains word pairs along with human-assigned similarity judgements. We compared our approach with 5 baseline representations. These include a model based on recurrent neural network (RNNLM) from 2010 [42] and a method trained using Noise Contrastive Estimation (NCE) presented in [43]. We also took into account two log bilinear methods by Mikolov, i.e. Continuous Bag of Words (CBoW) and mentioned here previously Skip-gram (SG) [8]. Finally, the implementation of the model proposed by Bojanowski et al. (Ft) was examined [28]. Apart from NCE case, we used publicly available codes of mentioned models.

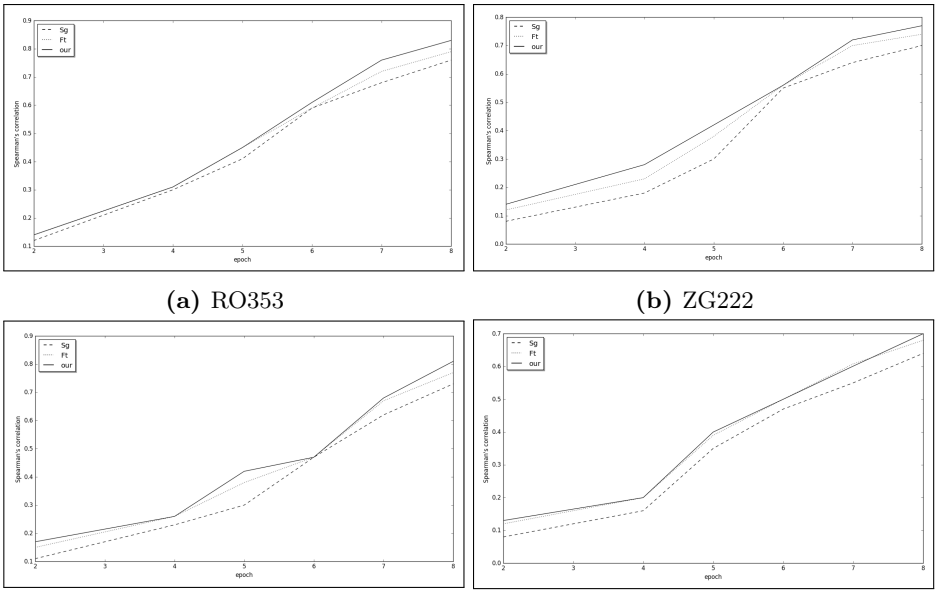
Setup details

In order to provide a reliable comparison, all the methods were trained on the same datasets. For the baseline methods we used default settings presented in papers with a

couple of exceptions. They include a context window of 6 words (both left and right). Additionally, the learning rate was fixed to 3×10^3 and the vector representations had the dimension 200.

Similarity judgement task

The most widely used method of representation quality evaluation is Spearman's rank correlation coefficient [44]. It enables to assess how well the given representations capture word similarity. For instance, "popular" and "famous" are supposed to be closer each other than "trendy" and "fruit". Thus, according to standard techniques, we calculated cosine distance between word pairs in datasets and reported Spearman's rank correlation coefficient between the rankings obtained from the models and human rankings. Table 1 yields the results for the word similarity task. It can be observed that our method slightly outperformed the baseline models in 3 cases. Two of them refer to German and Romanian, so it suggests the proposed technique better describes dense languages (note that German is much more dense than English).



(a) RO353 **(b)** ZG222

(c) SimVerb-3500 **(d)** WS353

Figure 2. The plots of performance versus training epoch for word similarity task.

Word analogy task

Another methods of evaluation are so-called analogy tasks [27]. They enable to assess syntactic and semantic relations between words. In practice, there are sets of questions and each question contains a missing word. The goal is to predict this word. The example of semantic question could be "brother" $\longleftrightarrow$ "sister" ; "grandson" $\longleftrightarrow$ "granddaughter", where the word "granddaughter" has to be predicted. According to [27], it is sufficient to obtain the vector $v = a_{brother} - a_{sister} + a_{grandson}$. We assume the answer is correct if the calculated vector v has high cosine similarity if compared to the good answer. The results for semantic and syntactic analogy tasks are listed

Table 4. Spearman's correlation coefficient for the word similarity task with a much more bigger corpus (200M tokens) and different length of vector representation.

representation length	Sg	Ft	our
200	0.61	**0.65**	**0.65**
300	0.64	0.69	**0.73**
400	0.66	0.71	**0.75**
500	0.7	0.76	**0.81**
600	0.72	0.83	**0.86**

Table 5. Semantic analogies task results with a much more bigger corpus (200M tokens) and different length of vector representation. The accuracy specified as %.

representation length	Sg	Ft	our
200	60.2	65.7	**70.1**
300	63.1	66.4	**68.1**
400	69.7	73.5	**75.2**
500	73.4	77.2	**81.1**
600	77	82.3	**84.8**

in Table 2 and 3, respectively. In fact, our method did not overcome any competing model. Nevertheless, it gave similar results to other Skip-gram based approaches. It shows it may be worth to explore the method's performance on more dense languages.

As one may observe, the previous experiments showed that the most significant results were achieved by Sg, Ft and our approach. Thus, we explored these methods more deeply in the next analysis. First of all, during the experiments we noticed that different models converged at different rates. Figure 2 plots the performance of the word similarity task on selected datasets after a specified number of epochs (2-8). The chart demonstrates that the all three models converge quickly to a satisfactory level of performance. Nevertheless, it appears that our approach yields more reliable results. This suggests that if training was done on more data, the representation could work better. Inspired by this observation, a few other experiments were carried out. We evaluated 3 representations, i.e. Sg, Ft, and our approach. The following tasks were undertaken: word similarity, syntactic and semantic analogies. They were trained on Wikipedia sets which contain 200M tokens [2]. The summary results of evaluations that consider vector space dimensions from 200 to 600 are presented in Table 4, 5 and 6. It is interesting to note that our model is in the vast majority of cases better than Sg and Ft. It performs favorably for either word similarity task (Table 4) or semantic analogies task (Table 5). Although the efficiency of our model on the syntactic analogies task is not strong, it provides some improvements, see Table 6. All in all, the results suggest that our approach benefits from a bigger corpus.

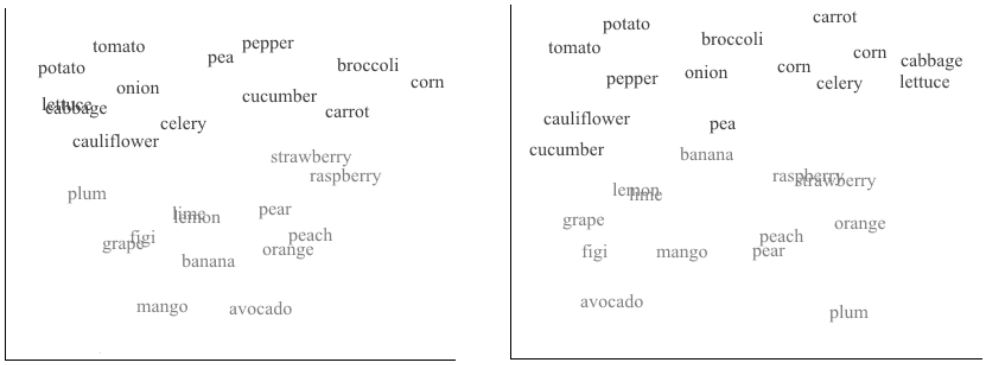
Finally, for our approach and Skip-gram method proposed by Bojanowski et al. we projected the learned word representations into two dimensions using the t-SNE

[2] `https://dumps.wikimedia.org/`

Table 6. Syntactic analogies task results with a much more bigger corpus (200M tokens) and different length of vector representation. The accuracy specified as %.

representation length	Sg	Ft	our
200	55.4	57.1	**57.6**
300	58.2	**59.3**	59
400	64.3	**67.8**	69.3
500	69.1	**73.2**	73
600	72.6	76.9	**77.1**

Figure 3. Two dimensional projections of our method and Bojanowski-based (right) word representations. Words associated with "fruit" are colored in grey, words associated with "vegetable" are colored in black. We can see that "fruit" and "vegetable" words are clustered correctly. However, our approach performs slightly better.



tool [45]. Figure 3 shows projections of the words related to the concept fruit vs. vegetable. The visual inspection demonstrates that all words were assigned to their groups correctly. However, the position of "peach" and "orange" seems to be more adequate if our model is considered.

4. Conclusion

In this paper, we propose method to learn word representations that considers fragments of words, including syllables and characters to built the model. We showed that our method outperforms state-of-the-art approaches on dense languages when tasks such as word similarity ranking or syntactic and semantic analogies are taken into consideration.

Acknowledgment

This research was partially supported by National Centre of Science (Poland) Grants No. 2016/21/N/ST6/01019.

5. References

[1] Miller, S., Guinness, J., Zamanian, A., Name tagging with word clusters and discriminative training. In: *Proceedings of HLT*, 2004, pp. 337–342.

[2] Vitz, P.C., Winkler, B.S., *Predicting the judged similarity of sound of english words.* Journal of Verbal Learning and Verbal Behavior, 1973, **12**(4), pp. 373 – 388.

[3] Rumelhart, D.E., Hinton, G.E., Williams, R.J., Neurocomputing: Foundations of research. MIT Press 1988 pp. 696–699.

[4] Schütze, H., Dimensions of meaning. In: *Proceedings of the 1992 ACM/IEEE Conference on Supercomputing.* Supercomputing '92, Los Alamitos, CA, USA, IEEE Computer Society Press, 1992, pp. 787–796.

[5] Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R., *Indexing by latent semantic analysis.* JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE, 1990, **41**(6), pp. 391–407.

[6] Hofmann, T., Probabilistic latent semantic indexing. In: *Proceedings of the 22Nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* SIGIR '99, New York, NY, USA, ACM, 1999, pp. 50–57.

[7] Baroni, M., Lenci, A., *Distributional memory: A general framework for corpus-based semantics.* December 2010, ,, **36**(4), pp. 673–721.

[8] Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J., Distributed representations of words and phrases and their compositionality. In Burges, C.J.C., Bottou, L., Welling, M., Ghahramani, Z., Weinberger, K.Q., eds.: *Advances in Neural Information Processing Systems 26.* Curran Associates, Inc. 2013 pp. 3111–3119.

[9] Pennington, J., Socher, R., Manning, C.D., Glove: Global vectors for word representation. In: *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543.

[10] Rehm, G., Uszkoreit, H., *The Romanian Language in the Digital Age.* Springer Publishing Company, Incorporated, 2012.

[11] Bilmes, J.A., Kirchhoff, K., Factored language models and generalized parallel backoff. In: *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: Companion Volume of the Proceedings of HLT-NAACL 2003–short Papers - Volume 2*. NAACL-Short '03, Stroudsburg, PA, USA, Association for Computational Linguistics, 2003, pp. 4–6.

[12] Botha, J.A., Blunsom, P., Compositional Morphology for Word Representations and Language Modelling. In: *Proceedings of the 31st International Conference on Machine Learning (ICML)*, jun 2014, ,, *Award for best application paper*.

[13] Luong, M.T., Socher, R., Manning, C.D., Better word representations with recursive neural networks for morphology. In: *CoNLL*, Sofia, Bulgaria, 2013.

[14] Mikolov, T., Sutskever, I., Deoras, A., Le, H.S., Kombrink, S., Cernocky, J., *Subword language modeling with neural networks*. preprint (http://www. fit. vutbr. cz/imikolov/rnnlm/char. pdf), 2012.

[15] Sutskever, I., Martens, J., Hinton, G.E., Generating text with recurrent neural networks. In: *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 2011, pp. 1017–1024.

[16] Zhang, X., Zhao, J., LeCun, Y., Character-level convolutional networks for text classification. In: *Advances in Neural Information Processing Systems*, 2015, pp. 649–657.

[17] Ling, W., Luís, T., Marujo, L., Astudillo, R.F., Amir, S., Dyer, C., Black, A.W., Trancoso, I., *Finding function in form: Compositional character models for open vocabulary word representation*. arXiv preprint arXiv:1508.02096, 2015.

[18] dos Santos, C.N., Gatti, M., Deep convolutional neural networks for sentiment analysis of short texts. In: *COLING*, 2014, pp. 69–78.

[19] Kim, Y., Jernite, Y., Sontag, D., Rush, A.M., *Character-aware neural language models*. arXiv preprint arXiv:1508.06615, 2015.

[20] dos Santos, C.N., Zadrozny, B., Learning character-level representations for part-of-speech tagging. In: *ICML*, 2014, pp. 1818–1826.

[21] Chrupała, G., Normalizing tweets with edit scripts and recurrent neural embeddings. In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Baltimore, Maryland, June 2014, ,, pp. 680–686.

[22] Luong, M.T., Manning, C.D., *Achieving open vocabulary neural machine translation with hybrid word-character models*. arXiv preprint arXiv:1604.00788, 2016.

[23] Sennrich, R., Haddow, B., Birch, A., Neural machine translation of rare words with subword units. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*, 2016.

[24] Cotterell, R., Schütze, H., Morphological word-embeddings. In: *Proc. of NAACL*, 2015.

[25] Sakamoto, N., Yamamoto, K., Nakagawa, S., Combination of syllable based n-gram search and word search for spoken term detection through spoken queries and iv/oov classification. Dec 2015, ,, pp. 200–206.

[26] Wechsler, M., Munteanu, E., Schäuble, P., New techniques for open-vocabulary spoken document retrieval. In: *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '98, New York, NY, USA, ACM, 1998, pp. 20–27.

[27] Mikolov, T., Chen, K., Corrado, G., Dean, J., *Efficient estimation of word representations in vector space*. arXiv preprint arXiv:1301.3781, 2013.

[28] Bojanowski, P., Grave, E., Joulin, A., Mikolov, T., *Enriching word vectors with subword information*. arXiv preprint arXiv:1607.04606, 2016.

[29] Gutmann, M.U., Hyvärinen, A., *Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics*. Journal of Machine Learning Research, 2012, **13**(Feb), pp. 307–361.

[30] Crystal, D., *Dictionary of linguistics and phonetics*. vol. 30. John Wiley & Sons, 2011.

[31] Mayer, T., Toward a totally unsupervised, language-independent method for the syllabification of written texts. In: *Proceedings of the 11th Meeting of the ACL Special Interest Group on Computational Morphology and Phonology*, Association for Computational Linguistics, 2010, pp. 63–71.

[32] Daelemans, W., van den Bosch, A., Generalization performance of backpropagation learning on a syllabification task. In: *Proceedings of the 3rd Twente Workshop on Language Technology*, Universiteit Twente, Enschede, 1992, pp. 27–38.

[33] Finkelstein, L., Gabrilovich, E., Matias, Y., Rivlin, E., Solan, Z., Wolfman, G., Ruppin, E., Placing search in context: The concept revisited. In: *Proceedings of the 10th international conference on World Wide Web*, ACM, 2001, pp. 406–414.

[34] Gerz, D., Vulić, I., Hill, F., Reichart, R., Korhonen, A., SimVerb-3500: A Large-Scale Evaluation Set of Verb Similarity. In: *EMNLP*, 2016.

[35] Hill, F., Reichart, R., Korhonen, A., *Simlex-999: Evaluating semantic models with (genuine) similarity estimation*. Computational Linguistics, 2016.

[36] Rubenstein, H., Goodenough, J.B., *Contextual correlates of synonymy*. October 1965, ,, **8**(10), pp. 627–633.

[37] Szumlanski, S.R., Gomez, F., Sims, V.K., A new set of norms for semantic relatedness measures. In: *ACL (2)*, 2013, pp. 890–895.

[38] Yang, D., Powers, D.M., *Verb similarity on the taxonomy of WordNet*. Masaryk University, 2006.

[39] Gurevych, I., Using the structure of a conceptual network in computing semantic relatedness. In: *International Conference on Natural Language Processing*, Springer, 2005, pp. 767–778.

[40] Zesch, T., Gurevych, I., Automatically creating datasets for measures of semantic relatedness. In: *Proceedings of the Workshop on Linguistic Distances*. LD '06, Stroudsburg, PA, USA, Association for Computational Linguistics, 2006, pp. 16–24.

[41] Hassan, S., Mihalcea, R., Cross-lingual semantic relatedness using encyclopedic knowledge. In: *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3-Volume 3*, Association for Computational Linguistics, 2009, pp. 1192–1201.

[42] Mikolov, T., Karafiát, M., Burget, L., Cernockỳ, J., Khudanpur, S., Recurrent neural network based language model. In: *Interspeech*. vol. 2., 2010, pp. 3.

[43] Mnih, A., Teh, Y.W., *A fast and simple algorithm for training neural probabilistic language models*. arXiv preprint arXiv:1206.6426, 2012.

[44] Spearman, C., *The proof and measurement of association between two things*. American Journal of Psychology, 1904, **15**, pp. 88–103.

[45] Maaten, L.v.d., Hinton, G., *Visualizing data using t-sne*. Journal of Machine Learning Research, 2008, **9**(Nov), pp. 2579–2605.

Schedae Informaticae Vol. 25 (2016): 117–126
doi: 10.4467/20838476SI.16.009.6190

tfml 2017
theoretical foundations
of machine learning, Kraków

Uniform Cross-entropy Clustering

Maciej Brzeski, Przemysław Spurek[1]
Faculty of Mathematics and Computer Science
Jagiellonian University, Łojasiewicza 6, 30-348 Kraków, Poland
e-mail: *maciej.brzeski@doctoral.uj.edu.pl* , *przemyslaw.spurek@uj.edu.pl*

Abstract. Robust mixture models approaches, which use non-normal distributions have recently been upgraded to accommodate data with fixed bounds. In this article we propose a new method based on uniform distributions and Cross-Entropy Clustering (CEC). We combine a simple density model with a clustering method which allows to treat groups separately and estimate parameters in each cluster individually. Consequently, we introduce an effective clustering algorithm which deals with non-normal data.

Keywords: Clustering, Cross-entropy, Uniform distribution

1. Introduction

Clustering plays a basic role in many parts of data engineering, pattern recognition, and image analysis. One of the most important clustering methods is the density approach [1, 2]. Most of such algorithms are based on Gaussian Mixture Model [3], which uses Expectation-maximization (EM) procedure [4]. The mixture components describe individual clusters in the data space. Gaussian components are traditionally successful in detecting elliptic clusters [3, 4, 5]. However, groups of a different shapes require a solution with involved components of other distributions.

Received: 11 December 2016 / Accepted: 30 December 2016
[1] The work of this author was supported by the National Science Centre (Poland) Grant No. 2015/19/D/ST6/01472.

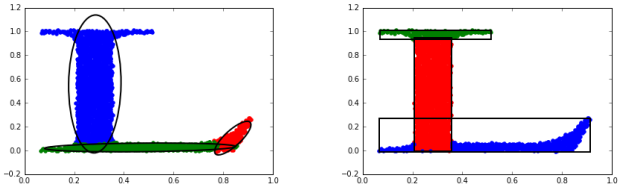
The growing need for more flexible tools to analyze datasets that exhibit non-normal features, including asymmetry, multimodality, heavy tails, and fixed bounds, has led to intense development of non-normal model-based methods. The mixture model-based clustering literature has focused on the development of mixture distributions with more flexible parametric components like split distributions [6, 7], skew distributions [8, 9, 10] and some other non-elliptical approaches [11, 12, 13].

The same situation occurs in the case of clustering non-negative or in some way limited data. To take into account such a feature, components should have a limited support. Therefore, we use the uniform distribution, which well covers clusters in the shape of rectangle. Estimation of data models with the bounded support including uniform ones was studied in various domains: clustering [14], individual state-space and regression models [15, 16] as well as mixture models [17]. In mixture-based clustering approach the challenging task is updating parameters of uniform components. Intuitively, the prior chosen bounds of the uniform distribution are only expandable, but they are not floating (limited support). Therefore, estimating a uniform mixture is very hard.

In this paper we construct a new clustering model Uniform Cross-Entropy Clustering (UCEC), which try to solve these problems. First of all, we use simple multidimensional uniform distribution, see Fig. 1. More precisely, we use uniform pdf for independent variables, which is a product of univariate marginal pdfs, and the distribution will have generally the rectangle support. Furthermore, simpler optimization procedure known as Cross Entropy Clustering (CEC) [18] is used instead of EM.

A goal of CEC is to optimally approximate the scatter of data set $X \subset \mathbb{R}^d$ by the function which is a small modification of EM (for more information see Section 2.). It occurs that at the small cost of having a minimally worse density approximation [18], we gain the ease of using more complicated density models. The method is capable of the automatic reduction of unnecessary clusters (contrary to EM each group has its cost). Moreover, we can treat clusters separately which is more effective from a numerical point of view.

This paper is arranged as follows. First the theoretical background of UCEC method is presented. We introduce the cost function which we need to minimize. Moreover, we present three strategies to escape from local minima to reach a better minimum. In the last part numerical experiments are presented.



(a) GMM (b) UCEC

Figure 1. The result of our approach and classical GMM in the case of the L-type dataset.

2. Theoretical background of UCEC

In this section the UCEC method will be presented. First, we introduce the cost function which will be optimized by the algorithm.

Our approach is based on the CEC [18]. Therefore, we start with a short introduction to the method. Since CEC is similar to EM in many aspects, let us first recall that, in general, EM aims to find $p_1, \ldots, p_k \geq 0$, $\sum_{i=1}^{k} p_i = 1$ and $f_1, \ldots, f_k$ Gaussian densities (where k is given beforehand and denotes the number of densities for which the convex combination builds the desired density model) such that the convex combination $f = p_1 f_1 + \ldots + p_k f_k$ optimally approximates the scatter of our data X with respect to the MLE cost function

$$\mathrm{MLE}(f, X) = -\sum_{\mathrm{x} \in X} \ln(p_1 f_1(\mathrm{x}) + \ldots + p_k f_k(\mathrm{x})). \tag{1}$$

A goal of CEC is to minimize the cost function, which is a minor modification of that given in (1) by substituting the sum with the maximum:

$$\mathrm{CEC}(f, X) = -\sum_{\mathrm{x} \in X} \ln(\max(p_1 f_1(\mathrm{x}), \ldots, p_k f_k(\mathrm{x}))). \tag{2}$$

Instead of focusing on the density estimation as its main task, CEC aims directly at the clustering problem. It occurs that a small cost of having a minimally worse density approximation [18], we obtain numerical efficient method. We can often use the Hartigan approach to clustering, which is faster and typically finds better minimums. This is an advantage, roughly speaking, because the models do not mix with each other since we take the maximum instead of the sum.

To apply CEC, we need to introduce the cost function which we want to minimize. To do so, let it be recalled that by the cross-entropy of data set $X \subset \mathbb{R}^d$ with respect to density f is given by

$$H^{\times}(X \| f) = -\frac{1}{|X|} \sum_{\mathrm{x} \in X} \ln(f(\mathrm{x})).$$

In the case of splitting $X \subset \mathbb{R}^d$ into $X_1, \ldots, X_k$ so that we describe elements of X_i using a function from the family of all multidimensional uniform densities $\mathcal{U}(\mathbb{R}^d)$.

As it was mentioned, we use simple multidimensional uniform distributions. Let us start from one dimensional density

$$U(x; a, b) = \begin{cases} \frac{1}{b-a} & x \in [a, b] \\ 0 & x \notin [a, b] \end{cases},$$

for $a, b \in \mathbb{R}$.

For a dataset $X \subset \mathbb{R}$ the maximum likelihood (ML) estimation of parameters $a, b \in \mathbb{R}$ of uniform distribution is given by maximal and minimal elements of X [19].

In our work we use multidimensional uniform distribution, which is product of univariate marginal pdfs.

Definition 2..1. *For a vector* $\mathrm{x} \in \mathbb{R}^d$ *the multidimensional uniform distribution is given by*

$$U_d(\mathrm{x};\mathrm{a},\mathrm{b}) = \prod_{j=1}^{d} U(x_j; a_j, b_j),$$

for $\mathrm{x} = [x_1, \ldots, x_d] \in \mathbb{R}^d$, $\mathrm{a} = [a_1, \ldots, a_d] \in \mathbb{R}^d$, $\mathrm{b} = [b_1, \ldots, b_d] \in \mathbb{R}^d$, *where* $a_i < b_i$ *for* $i = 1, \ldots, d$.

Similar to the one dimensional case the maximum likelihood estimators are given by by maximal and minimal elements of X [19].

Theorem 2..1. *Let* $X = \{\mathrm{x}_1, \ldots, \mathrm{x}_n\}$ *be a random sample from* $U_d(\mathrm{x};\mathrm{a},\mathrm{b})$. *Then the maximum likelihood estimators of* a *and* b *are*

$$\hat{\mathrm{a}} = \min(X) = [\min(X^1), \ldots, \min(X^d)],$$
$$\hat{\mathrm{b}} = \max(X) = [\max(X^1), \ldots, \max(X^d)].$$

The support of uniform density distributions $U_d(\mathrm{x};\mathrm{a},\mathrm{b})$ is hyperrectangle

$$supp(U_d(\mathrm{x};\mathrm{a},\mathrm{b})) = \{U_d(\mathrm{x};\mathrm{a},\mathrm{b}) \neq 0, x \in \mathbb{R}^d\} = [a_1, b_1] \times \ldots \times [a_d, b_d].$$

Therefore, for given uniform distribution $U_d(\mathrm{x};\mathrm{a},\mathrm{b})$ volume of his support is equal to $V_{U_d(\mathrm{x};\mathrm{a},\mathrm{b})} = |b_1 - a_1| \cdot \ldots \cdot |b_d - a_d|$. Now we are ready to present the cost function, which will be used in our algorithm

$$E(X_1, \ldots, X_k; \mathcal{U}(\mathbb{R}^d)) = \sum_{i=1}^{k} p_i \cdot \left(-\ln(p_i) + H^{\times}(X_i \| \mathcal{U}(\mathbb{R}^d))\right), \tag{3}$$

where $p_i = \frac{|X_i|}{|X|}$ and $H^{\times}(X_i \| \mathcal{U}(\mathbb{R}^d)) = \inf_{f \in \mathcal{U}(\mathbb{R}^d)} H^{\times}(X_i \| f)$.

The aim of CEC is to split dataset X into subsets X_i which minimize the function given in (3). It is easy to see that in the case of one cluster X, the cross-entropy is equivalent to the log-likelihood function:

$$H^{\times}(X \| U_d(\mathrm{a},\mathrm{b})) = -\tfrac{1}{|X|} \sum_{\mathrm{x} \in X} \ln(U_d(\mathrm{a},\mathrm{b})) = -\tfrac{1}{|X|} \ln(L(X;\mathrm{a},\mathrm{b})).$$

Consequently, we can minimize cross-entropy by maximizing log-likelihood. This approach allows us to fit optimal parameters in each cluster and minimize the cost function (3).

In the case of uniform distributions the formula for negative log-likelihood function is given as follows

$$H^{\times}(X \| U_d(\mathrm{a},\mathrm{b})) = -\frac{1}{|X|} \ln(L(X;\mathrm{a},\mathrm{b})) = -\frac{1}{|X|} \sum_{\mathrm{x} \in X} \ln(U_d(\mathrm{x},\mathrm{a},\mathrm{b})) = \ln\left(V_{U_d(\mathrm{x};\mathrm{a},\mathrm{b})}\right).$$

Therefore, our cost function depends on the volume of the support of densities which describes clusters:

$$E(X_1, \ldots, X_k; \mathcal{U}(\mathbb{R}^d)) = \sum_{i=1}^{k} p_i \left(-\ln(p_i) + \ln(V_{U_d(\mathrm{x};\mathrm{a}_i,\mathrm{b}_i)})\right),$$

where $\mathrm{a}_i = \min(X_i), \mathrm{b}_i = \max(X_i)$.

Let us now introduce the algorithm step by step. The UCEC method starts from an initial clustering, which can be obtained randomly or with use of the k-means++ approach.

In our work we use the Hartigan method [20, 21, 22]. The aim of Hartigan method is to find partition $X_1, \ldots, X_n$ of X which cost function (3) is as close as possible to the minimum by subsequently reassigning membership of elements from X.

To explain Hartigan approach more precisely we need the notion of group membership function $gr : \{1, \ldots, n\} \to \{0, \ldots, k\}$, which describes the membership of i-th element, where 0 value is a special symbol which denotes that x_i is as yet unassigned. In other words: if $gr(i) = l > 0$, then x_i is a part of the l-th group, and if $gr(i) = 0$ then x_i is unassigned.

Basic idea of Hartigan is relatively simple – we repeatedly go over all elements of X and apply the following steps:

- if the chosen element x_i is unassigned, assign it to the first nonempty group;
- reassign x_i to these group, which decrease cost function;
- check if no group needs to be removed/unassigned, if this is the case unassign its all elements;

until no group membership has been changed.

To implement Hartigan approach for discrete measures we still have to add a condition when we unassign given group. For example in the case of Uniform clustering in R^d to avoid overfitting we cannot consider clusters which contain less then $d + 1$ points. In practice while applying Hartigan approach on discrete data we usually removed clusters which contained less then three percent of all data-set.

Observe that in the crucial step in Hartigan approach we compare the cross-entropy after and before the switch, while the switch removes a given set from one cluster and adds it to the other. It means that to apply efficiently the Hartigan approach in clustering it is essential to update parameters.

To calculate cost function we need to calculate minimum and maximum for every dimension of new clusters. If we use simple arrays to keep data, it will take $O(d \cdot k \cdot n^2)$ time per loop (k is number of clusters, d dimension of data). So we use BST tree for every dimension, which gives us min and max in $O(\ln n)$ time. Moreover, we can calculate maximum only after switching point - for change cost function we can just take maximum of current maximums and added point. It enable to decrease time per loop to $O(n \cdot d(\ln n + k))$.

In classical Hartigan approach we switch elements one by one. In the case of uniform distribution this approach is ineffective since the algorithms stacks in local minimums. The effect is caused by the finite support of uniform distributions. In most cases switching one point does not decrease the size of hyperrectangle. Therefore, we consider three possible scenarios of switching points. The classical UCEC switch only one point. In the second version random UCEC (UCEC-r), we sometimes randomly move some points to another class and minimize it using Hartigan again. It gives better results, but it is time consuming, especially when we apply many random switches.

In the third version multi-points movement UCEC (UCEC-m), we move subsets of points which lie in the borders of supports of uniform densities. The motivation for the solution comes from the observation that for two clusters $X_1, X_2 \subset R^d$ which supports have nonempty intersection, it is profitable to add all points from intersection to the same cluster.

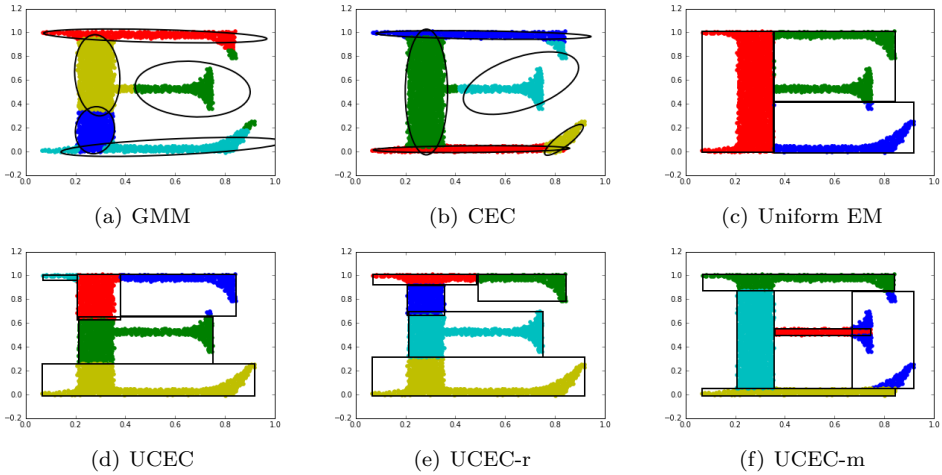
Theorem 2..2. *Let* $X_1, X_2 \subset \mathbb{R}^d$ *such that* $X_1 \cap X_2 = \emptyset$, $X_1 \cup X_2 = X$ *be given. Let*

$$X_\cap = X \cap supp(U_d(\mathrm{a}_1, \mathrm{b}_1)) \cap supp(U_d(\mathrm{a}_2, \mathrm{b}_2)) \neq \emptyset$$

where $\mathrm{a}_1 = \min(X_1), \mathrm{b}_1 = \max(X_1), \mathrm{a}_2 = \min(X_2), \mathrm{b}_2 = \max(X_2)$ *be such that* $X_\cap \subset X_1$ *and* $E(X_1, X_2, \mathcal{U}(\mathbb{R}^d)) \leq E(X_1 \setminus X_\cap, X_2 \cup X_\cap, \mathcal{U}(\mathbb{R}^d))$. *Then*

$$E(X_1, X_2, \mathcal{U}(\mathbb{R}^d)) \leq E(Y_1, Y_2, \mathcal{U}(\mathbb{R}^d)),$$

for any other clustering such that $\min(X_1) = \min(Y_1), \max(X_1) = \max(Y_1), \min(X_2) = \min(Y_2), \max(X_2) = \max(Y_2)$.



(a) GMM (b) CEC (c) Uniform EM

(d) UCEC (e) UCEC-r (f) UCEC-m

Figure 2. The effect of different clustering algorithms in the case of E-type dataset.

Proof. Cross entropy is equal to:

$$\begin{aligned} E(X_1, X_2, \mathcal{U}(\mathbb{R}^d)) =& p \cdot \left(-\ln(p) + V_{U_d(\mathrm{x};\mathrm{a}_1,\mathrm{b}_1)}\right) + (1-p) \cdot \left(-\ln(1-p) + V_{U_d(\mathrm{x};\mathrm{a}_2,\mathrm{b}_2)}\right) \\ =& p \cdot \ln\left(\frac{V_{U_d(\mathrm{x};\mathrm{a}_1,\mathrm{b}_1)}}{p}\right) + (1-p) \cdot \ln\left(\frac{V_{U_d(\mathrm{x};\mathrm{a}_2,\mathrm{b}_2)}}{1-p}\right). \end{aligned}$$

We consider only such clustering which does not change maximal and minimal values in cluster. Therefore, value of a cost function depends only on p. We can consider a simpler function

$$f(p) = p \cdot \ln\left(\frac{V_1}{p}\right) + (1-p) \cdot \ln\left(\frac{V_2}{1-p}\right).$$

where V_1, V_2 are constant. By analyzing the first derivative of f

$$f'(p) = \ln\left(\frac{V_1}{p}\right) - p \cdot \frac{p}{V_1} \cdot \frac{V_1}{p^2} - \ln\left(\frac{V_2}{1-p}\right) + (1-p) \cdot \frac{1-p}{V_2} \cdot \frac{V_2}{(1-p)^2} = \\ = \ln\left(\frac{V_1}{p}\right) - \ln\left(\frac{V_2}{1-p}\right),$$

we obtain that f has one local maximum and no local minimums. Therefore minimum is at one of ends of domain. As a simple corollary we obtain that $E(X_1, X_2, \mathcal{U}(\mathbb{R}^d))$ obtain minimum when all points from $X_\cap$ are in one cluster. □

data	cl.	scoring function	GMM	CEC	U-EM	UCEC	UCEC-r	UCEC-m
D	4	avg l-l	0,479	**0,588**	-0,057	0,487	0,541	0,579
		AIC	-3206	**-3974**	424	-3259	-3625	-3884
		BIC	-3089	**-3943**	540	-3143	-3509	-3768
	5	avg l-l	0,506	**0,736**	-0,070	0,483	0,457	0,606
		AIC	-3377	**-4973**	504	-3224	-3045	-4057
		BIC	-3230	**-4931**	590	-3077	-2898	-3910
	6	avg l-l	0,627	**0,798**	-0,055	0,473	0,600	0,679
		AIC	-4192	**-5387**	378	-3149	-4008	-4545
		BIC	-4014	**-5338**	403	-2972	-3830	-4367
E	4	avg l-l	0,440	0,695	0,284	0,368	0,393	**0,839**
		AIC	-2176	-3487	-1384	-1813	-1941	**-4186**
		BIC	-2065	-3452	-1274	-1702	-1831	**-4074**
	5	avg l-l	0,609	0,829	0,329	0,488	0,534	**0,972**
		AIC	-3019	-4159	-1627	2410	-2589	**-4846**
		BIC	-2880	-4118	-1546	-2270	-2449	**-4706**
	6	avg l-l	0,631	0,829	0,379	0,645	0,677	**1,073**
		AIC	-3117	-4159	-1878	-3191	-3348	**-5345**
		BIC	-2948	-4118	-1797	-3021	-3179	**-5176**
L	4	avg l-l	1,121	1,273	0,506	0,961	1,243	**1,351**
		AIC	-4460	-5097	-2010	-3816	-4950	**-5381**
		BIC	-4353	-5063	-1959	-3710	-4843	**-5274**
	5	avg l-l	1,164	1,315	0,504	1,141	1,255	**1,419**
		AIC	-4623	-5262	-2005	-4528	-4986	**-5644**
		BIC	-4489	-5223	-1955	-4394	-4851	**-5510**
	6	avg l-l	1,160	1,332	0,504	1,077	1,287	**1,432**
		AIC	-4597	-5327	-2005	-4264	-5106	**-5685**
		BIC	-4435	-5283	-1955	-4101	-4944	**-5523**

Table 1. The results of classical algorithms in the case of letter-type data.

In natural way it is impossible to verify all possibles subsets which lies in the borders of clusters. But we can take advantage by using Theorem 2..2 Therefore

instead of considering one point we will use all elements which lie in the intersection of supports of considered clusters. k-d trees [23] can be used to increase performance and enable faster search.

Thanks to above modifications and suitable data structures (like k-d trees or BST trees) we obtain effective algorithm for clustering datasets by uniform distributions.

3. Experiments

In this section, we present a comparison of our method with different scenario (UCEC, UCEC-r, UCEC-m) and classical clustering algorithms k-means, GMM, CEC and uniform EM.

In the first example we use letters type datasets (D, E and L), see Fig. 2. To compare the results, we use the standard Bayesian Information Criterion $BIC = ?2LL + k\ln(n)$ and Akaike Information Criterion $AIC = ?2LL + 2k$, where k is the number of parameters in the model, n is the number of points, and LL is a maximized value of the log-likelihood function. We need a number of parameters which are used in our model. The UCEC model uses two scalars for minimal and maximal value for each dimensions $k \cdot 2d$. The results of our experiment are presented in Table 1. In the case of letters which contains uniform distributions on rectangles (letters E and L) our approach (UCEC-m) gives the best results. On the other hand, if data contains curve types structures (letter D) classical approaches fit data with higher precision.

data	scoring function	k-m	GMM	CEC	U-EM	UCEC	UCEC-r	UCEC-m
iris	a-rand	0,730	0,758	**0,901**	0,512	0,772	0,772	0,772
	avg l-l	-	-2,058	**-1,208**	-3,608	-2,270	-2,27	-2,27
	AIC	-	670	**386**	1134	733	733	733
	BIC	-	748	**422**	1213	811	811	811
cancer	a-rand	0,491	**0,755**	0,000	0,068	0,458	0,439	0,545
	avg l-l	-	-3,308	-11,5	-18,70	-3,212	-3,074	**-3,069**
	AIC	-	4006	13193	21520	3898	3740	**3735**
	BIC	-	4532	13454	22046	4423	4266	**4260**
seeds	a-rand	**0,717**	0,679	0,630	0,515	0,632	0,640	0,671
	avg l-l	-	-1,409	**6,079**	-3,080	-1,276	-1,273	-1,311
	AIC	-	680	**-2493**	1382	624	623	639
	BIC	-	827	**-2393**	1529	771	770	786
wine	a-rand	0,371	**0,915**	0,023	0,203	0,571	0,421	0,383
	avg l-l	-	-18,5	**-17,32**	-22,07	-19,97	-19,91	-19,93
	AIC	-	6750	**6351**	8016	7268	7247	7256
	BIC	-	7005	**6644**	8271	7522	7502	7510

Table 2. The results of classical algorithms in the case of data from UCI repository.

In the second example we use real datasets with labels from UCI repository. In

the experiment we use BIC, AIC measures for verify which model fits data best. On the other hand, we use adjusted rand index to check which model is able to recover reference clustering. The results of our experiment one presented in Table 2. Results of recovering clustering for UCEC are comparable with k-means and worse than GMM.

4. Conclusions

In the paper we construct UCEC, a fast clustering algorithm which describes components by using uniform distributions. In our algorithm we use a data structure like k-d trees or BST trees which allows to implement effective from a numerical optimization point of view, algorithm. Therefore, we obtain a flexible tool for analyzing data with finite support. Moreover, due to its nature UCEC automatically removes unnecessary clusters and therefore can be successfully applied in typical situations where the correct number of groups is not known.

5. References

[1] Jain, A., *Data clustering: 50 years beyond K-means.* Pattern Recognition Letters, 2010, **31**(8), pp. 651–666.

[2] Levin, M.S., *Combinatorial clustering: Literature review, methods, examples.* Journal of Communications Technology and Electronics, 2015, **60**(12), pp. 1403–1428.

[3] McLachlan, G., Krishnan, T., *The EM algorithm and extensions.* vol. 382. John Wiley & Sons, 2007.

[4] McLachlan, G., Peel, D., *Finite mixture models.* John Wiley & Sons, 2004.

[5] Tabor, J., Misztal, K., Detection of elliptical shapes via cross-entropy clustering. In: *Pattern Recognition and Image Analysis.* vol. 7887., Jun 2013, ,, pp. 656–663.

[6] Elguebaly, T., Bouguila, N., *Background subtraction using finite mixtures of asymmetric gaussian distributions and shadow detection.* Machine vision and applications, 2014, **25**(5), pp. 1145–1162.

[7] Spurek, P., *General split gaussian cross–entropy clustering.* Expert Systems with Applications, 2017, **68**, pp. 58–68.

[8] Lee, S.X., McLachlan, G.J., *Finite mixtures of canonical fundamental skew t-distributions.* Statistics and Computing, 2015, pp. 1–17.

[9] Lin, T.I., Ho, H.J., Lee, C.R., *Flexible mixture modelling using the multivariate skew-t-normal distribution.* Statistics and Computing, 2014, **24**(4), pp. 531–546.

[10] Vrbik, I., McNicholas, P., *Analytic calculations for the em algorithm for multivariate skew-t mixture models.* Statistics & Probability Letters, 2012, **82**(6), pp. 1169–1174.

[11] Browne, R.P., McNicholas, P.D., *A mixture of generalized hyperbolic distributions.* Canadian Journal of Statistics, 2015.

[12] Śmieja, M., Wiercioch, M., *Constrained clustering with a complex cluster structure.* Advances in Data Analysis and Classification, pp. 1–26.

[13] Spurek, P., Tabor, J., Byrski, K., *Active function cross-entropy clustering.* Expert Systems with Applications, 2017, **72**, pp. 49–66.

[14] Banfield, J.D., Raftery, A.E., *Model-based gaussian and non-gaussian clustering.* Biometrics, 1993, pp. 803–821.

[15] Jirsa, L., Pavelková, L., Estimation of uniform static regression model with abruptly varying parameters. In: *Informatics in Control, Automation and Robotics (ICINCO), 2015 12th International Conference on.* vol. 1., IEEE, 2015, pp. 603–607.

[16] Pavelková, L., Kárnỳ, M., *State and parameter estimation of state-space model with entry-wise correlated uniform noise.* International Journal of Adaptive Control and Signal Processing, 2014, **28**(11), pp. 1189–1205.

[17] Nagy, I., Suzdaleva, E., Mlynárová, T., Mixture-based clustering non-gaussian data with fixed bounds. In: *Proceedings of the IEEE International conference Intelligent systems IS.* vol. 16., 2016, pp. 4–6.

[18] Tabor, J., Spurek, P., *Cross-entropy clustering.* Pattern Recognition, 2014, **47**(9), pp. 3046–3059.

[19] Casella, G., Berger, R.L., *Statistical inference.* vol. 2. Duxbury Pacific Grove, CA, 2002.

[20] Hartigan, J.A., *Clustering algorithms*, 1975.

[21] Śmieja, M., Tabor, J., Spherical wards clustering and generalized voronoi diagrams. In: *Data Science and Advanced Analytics (DSAA), 2015. 36678 2015. IEEE International Conference on*, IEEE, 2015, pp. 1–10.

[22] Telgarsky, M., Vattani, A., Hartigan's method: k-means clustering without voronoi. In: *AISTATS*, 2010, pp. 820–827.

[23] Bentley, J.L., *Multidimensional binary search trees used for associative searching.* Communications of the ACM, 1975, **18**(9), pp. 509–517.

Schedae Informaticae Vol. 25 (2016): 127–138
doi: 10.4467/20838476SI.16.010.6191

tfml 2017
theoretical foundations
of machine learning, Kraków

Word Embeddings for Morphologically Complex Languages*

Grzegorz Jurdziński
Department of Theoretical Computer Science
Faculty of Mathematics and Computer Science of the Jagiellonian University
ul. prof. Stanisława Łojasiewicza 6, 30-348 Kraków
e-mail: *grzegorz.jurdzinski@student.uj.edu.pl*

Abstract. Recent methods for learning word embeddings, like GloVe or Word2Vec, succeeded in spatial representation of semantic and syntactic relations. We extend GloVe by introducing separate vectors for base form and grammatical form of a word, using morphosyntactic dictionary for this. This allows vectors to capture properties of words better. We also present model results for word analogy test and introduce a new test based on WordNet.

Keywords: machine learning, word embeddings, natural language processing, morphology

1. Introduction

Word embedding methods assign vectors in continuous n-dimensional space to words from a language. These can be used for various tasks, such as information retrieval [1], document classification [2], question answering [3], named entity recognition [4] and parsing [5].

Most word vector methods are supposed to cluster words that have similar meaning and their performance was evaluated based on experiments testing distance or

Received: 11 December 2016 / Accepted: 30 December 2016

* First version of this work was prepared as Bachelor Thesis at Institute of Computer Science of University of Wrocław. Its preparation was supervised by Jan Chorowski Ph.D., Institute of Computer Science, Faculty of Mathematics and Computer Science of the University of Wrocław, ul. Joliot-Curie 15, 50-383 Wrocław (email: *jan.chorowski@cs.uni.wroc.pl*)

angle between pairs of words. [6] introduced a more complex evaluation scheme. It is based on word analogies that examine finer structure of word vector space on various dimensions of difference. For example, the analogy *"king is to queen as man is to woman"* should be encoded in vectors by equation *vector("king") - vector("queen") = vector("man") - vector("woman")*. Indeed many of mentioned word embedding methods produce representations encoding such relations well.

Models like Word2Vec [7] and GloVe [8] receive worse scores on syntactic part of this test. The idea of our solution is to produce separate vectors for the base form of a word and the set of tags describing its grammatical form (called tagset in further part of this work). For example for Polish word *"jabłek"* (genitive case of word *"apples"*) its base form is *"jabłko"* (*"apple"*) and its tagset is *"subst:pl:gen:n2"* (describing that it is a plural form of a noun in genitive case with neuter grammatical gender). Such decomposition of a word can be obtained using morphosyntactic dictionaries (e.g. Polimorfologik for Polish). Then, during learning, vector for each word is represented as sum of vectors of its base form and tagset. One of benefits of such approach is giving the model a possibility to gather more information about rare words. Models mentioned above treat occurrences of the same word in different grammatical forms as separate words. For example there is no direct connection between *"jabłkami"* and *"jabłku"*. Out model has a common base form vector for all forms of *"jabłko"* so it is able to make use out of each occurrence of word regardless its grammatical form.

1.1. Related work

Word embedding was analyzed and implemented in various works. We shortly describe some important examples below.

Feedforward Neural Net Language Model (NNLM) [9] uses word vectors as its parameters. The network itself models the language – that means than when fed with N words (where N is a fixed, chosen number) it produces a probability distribution over all words from the language. For each word it should be the probability of appearing after the N given words. Part of the neural network is a shared matrix of word vectors. NNLM consist of input, projection, hidden and output layers. In the input layer N previous words are encoded using 1-of-V coding (where V is size of the vocabulary), then it is projected to a projection layer P that has dimensionality $N \cdot D$ (where D is the dimensionality of word vectors), using a shared projection matrix. That means that each 1-of-V vector is replaced with a word vector from the shared matrix and then all of them create one big $N \cdot D$ vector. Between the projection and hidden layers there is a dense connection and results of the hidden layer are used by the output layer to compute a probability distribution over all words in the vocabulary using the softmax function.

Word2Vec [10, 7] implements two models – Continuous Bag-of-Words Model (CBOW) and Continuous Skip-gram Model (Skip-gram). Both models were based on NNLM. They consist of input, projection and output layers. CBOW, after projecting words to their vectors, averages them to one vector (NNLM was joining them to create bigger vector) and uses an output layer to predict the word. Also, instead of predicting next

word, after given N words, it predicts middle word given words within certain range. Skip-gram model, instead of predicting a single word based on context, predicts the context itself. Authors of Word2Vec made many improvements with respect to the complexity of the model. They have replaced softmax with its efficient approximations – they have tested Hierarchical Softmax and Negative Sampling [7].

Problem of representing morphology in word embeddings was already tackled before. One of the examples are morphoRNN [11] and [12]. Both of these works split words into morphemes and learn separate vectors for theme. For example word *"unfortunately"* would be split into *"un"*, *"fortunate"* and *"ly"*. To get a vector for a word [12] sum up vectors of its morphemes. They also learn a separate vector for each word and add it to its morphemes vectors, so for example vector for word *"greenhouse"* would be *vector("greenhouse") + vector("green") + vector("house")* (final vector for the word *"greenhouse"* and the vector used to compute it are two different vectors). [11] present more complex way of combining morpheme's vectors. To produce a word vector a small neural network is used. At each step one affix and word stem are combined. A new vector is produced from stem vector ($x_{\texttt{stem}}$) and affix vector($x_{\texttt{affix}}$) as follow: $p = f(W_m[x_{\texttt{stem}}; x_{\texttt{affix}}] + b_m)$. Vectors for stem and affix are combined into bigger vector and multiplied by the matrix of parameters (W_m) and bias vector (b_m) is added. f is an element-wise activation function (tanh for example). So for example a vector for a word *("unfortunately")* would be computed in two steps. First vector for *"unfortunate"* would be computed by combining vectors for *"un"* and *"fortunate"*. Then vectors for *"unfortunate"* and *"ly"* would be combined to give vector for *"unfortunately"*.

The main difference between these two works and the approach we present is the way of splitting the words – instead of looking at the morphemes the word consists of we take its base form and the set of the tags describing its grammatical form.

GloVe was introduced by [8]. Since our work is based on it, we will describe it precisely in next section.

2. GloVe model

GloVe model, introduced by [8], utilizes two approaches to the problem – matrix factorization methods and shallow window-based methods. First uses a large matrix that captures statistical information about a corpus, e.g. rows can correspond to terms, columns to documents in the corpus and cells are numbers of occurrences of term in document. Such approach is used by [13] in latent semantic analysis (LSA). Examples of shallow window-based methods are NNLMs and Word2Vec. Their approach is to learn word representation which aim is to predict word given a local context window of a few, typically $5 - 15$ words.

Before learning vectors a co-occurrence matrix X is created by counting word co-occurrences in a corpus. For that the context window size is being chosen – let's denote it as ws. We will say that a word j occurs in context of a word i if it occurs in the corpus within ws distance from i. $X \in \mathbb{N}^{V \times V}$ (where V is size of vocabulary –

number of different words in corpus) is a word-word co-occurrence matrix, that means X_{ij} is number of occurrences of word j in context of word i. We will call word j a *context word*.

Word vectors are learned based on the matrix X. For each word there are two separate vectors in GloVe – the *word vector* and the *context vector*. Let w_i be the word vector of word i and $\tilde{w}_i$ be the context vector of word i. Analogically there are two biases for each word – b_i and $\tilde{b}_i$.

Authors of GloVe claim that word vectors should satisfy the following equation ([8] give full justification for this formula):

$$w_i^T \tilde{w}_j + b_i + \tilde{b}_j = \log(X_{ij}) \tag{1}$$

For learning vectors equation 1 is treated as least squares problem. This results in a cost function defined as following:

$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right) \left(w_i^T \tilde{w}_k + b_i + \tilde{b}_k - \log X_{ik}\right)^2 \tag{2}$$

where $f(X_{ij})$ is a weighting function. Main reason for introducing it is to prevent rare co-occurrences from influencing the model too strongly. According to Pennington et al. weighting function should satisfy the following properties:

1. $f(0) = 0$. If f is viewed as a continuous function, it should vanish as $x \to 0$ fast enough that the $\lim_{x\to 0} f(x) \log^2 x$ is finite.

2. $f(x)$ should be non-decreasing so that rare co-occurrences are not overweighted.

3. $f(x)$ should be relatively small for large values, so that frequent co-occurrences are not overweighted.

Authors of GloVe were using the following weighting function:

$$f(x) = \begin{cases} (x/x_{\max})^\alpha & \text{if } x < x_{\max} \\ 1 & \text{otherwise} \end{cases} \tag{3}$$

where $x_{\max}$ and α are left as free parameters. They have discovered that $x_{\max}$ does not influence the model strongly and were using $x_{\max} = 100$ for all their tests. They have also found that $\alpha = 3/4$ gives slight improvement over $\alpha = 1$.

The cost function (2) is minimized using the AdaGrad algorithm [14] – variation of regular Gradient Descent. At each step a pair of words i and j is being chosen and the cost of approximating X_{ij} is being computed. Every parameter (vectors and biases) is then being updated using gradient. A full iteration of the algorithm goes over all pairs of words. The program is being run for a fixed number of iterations.

Resulting vectors create many clusters – words with similar meaning or grammatical form are grouped together. Unfortunately often syntactics of words is not represented by these clusters so words considering similar subject but with completely different grammatical form are close.

3. Extentded GloVe model

The main idea of our extension of GloVe model is to replace word vectors with vectors for base forms and tagsets – for each word its vector will be the sum of vectors of its base and grammatical form. It requires finding the base form and the tagset for each word in the vocabulary and changing the model itself. Such approach gives less freedom to the model – regular GloVe could easily place word vectors in most suitable place, here it is not always possible since base forms vectors and tagsets vectors are shared by various words – but it also gives extra information that model can use.

3.1. Obtaining base forms and tagsets for words

For obtaining base forms and tagsets for words we have used the morphosyntactic dictionary Polimorfologik [15]. For each word it defines its base form and a set of tags describing its syntactic form. If its ambiguous, several tagsets, separated with plus sign, are defined. An example line of Polimorfologik looks like this:

`kot;kotami;subst:pl:inst:m1+subst:pl:inst:m2`

The example word is *"kotami"* (*"cats"* in instrumental case). The first column is the base form – `kot` (*"cat"*) here. The second column is the word itself – `kotami`. The third column – `subst:pl:inst:m1+subst:pl:inst:m2` – is the set of tags describing grammatical form of the word. Two tags (`subst:pl:inst:m1` and `subst:pl:inst:m2`) are separated by the `+` sign due to the ambiguouity of the grammatical form of the word. Both contain parts describing that it is a noun (`subst`), in plural form (`pl`), in instrumental case (`inst`). The difference between them is the part responsible for grammatical gender of the word (`m1` and `m2`). The word *"kotami"* in Polish can be interpreted as as in two of three possible musculine grammatical genders – personal and animate.

We have prepared scripts extracting base forms and tagsets for words from vocabulary, putting them in separate files and creating a file that for each word contains line with its base form and tagset. For words that were not found in the Polimorfologik we have the word itself to be its base form and assigned an empty grammatical tag. Therefore all unknown words are assigned to the same grammatical tagset. Please note that this doesn't preclude learning correct embeddings for words not found in Polimorfologik – in fact, since the base form of such words is unique their embedding is not shared with other words and the model is free to place them whenever is appropriate in the embedding space so the results for these words should be similar to the regular GloVe.

3.2. Learning separate vectors

Introducing separate vectors for base forms and tagsets required changes in model. First denote J_{ij} as single element of a sum from equation 2 for words i and j. Let $\text{fdiff} = f\left(X_{ij}\right)\left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)$ and $w_{i,k}$ be k^{th} element of vector w_i. Then we have

$$\begin{aligned}
\frac{\partial J_{ij}}{\partial w_{i,k}} &= \frac{\partial}{\partial w_{i,k}} f(X_{ij})(w_i^T \tilde{w}_j + b_i + \tilde{b}_j + \log X_{ij})^2 \\
&= 2 \cdot \text{fdiff} \cdot \frac{\partial}{\partial w_{i,k}} \left(\sum_{l=1}^{n} w_{i,l} \tilde{w}_{j,l} + b_i + \tilde{b}_j - \log X_{ij} \right) \\
&= 2 \cdot \text{fdiff} \cdot \tilde{w}_{j,k}
\end{aligned} \tag{4}$$

where n is dimensionality of vectors.

For learning separate vectors for base forms and tagsets we replaced word vectors with sums of vectors of its forms, resulting with following cost function:

$$J^{(2)} = \sum_{i,j=1}^{V} f(X_{ij})((w_i + v_i)^T(\tilde{w}_j + \tilde{v}_j) + (b_i^w + b_i^v) + (\tilde{b}_j^w + \tilde{b}_j^v) + \log X_{ij})^2 \tag{5}$$

where w_i and v_i are vectors of base form and tagset of word i (analogically for context word and biases).

Let $J_{ij}^{(2)}$ be analogical to J_{ij} for equation 5 and fdiff to the previous definition. Than we have:

$$\begin{aligned}
&\frac{\partial J_{ij}^{(2)}}{\partial w_{i,k}} = \frac{\partial}{\partial w_{i,k}} f(X_{ij})((w_i + v_i)^T(\tilde{w}_j + \tilde{v}_j) + (b_i^w + b_i^v) + (\tilde{b}_j^w + \tilde{b}_j^v) + \log X_{ij})^2 \\
&= 2 \cdot \text{fdiff} \cdot \frac{\partial}{\partial w_{i,k}} \left(\sum_{l=1}^{n} (w_{i,l} + v_{i,l})(\tilde{w}_{j,l}\tilde{v}_{j,l}) + (b_i^w + b_i^v) + (\tilde{b}_j^w + \tilde{b}_j^v) + \log X_{ij} \right) \\
&= 2 \cdot \text{fdiff} \cdot \frac{\partial}{\partial w_{i,k}} \left(\sum_{l=1}^{n} (w_{i,l} + v_{i,l})(\tilde{w}_{j,l}\tilde{v}_{j,l}) \right) \\
&= 2 \cdot \text{fdiff} \cdot \frac{\partial}{\partial w_{i,k}} (w_{i,k}\tilde{w}_{j,k} + w_{i,k}\tilde{v}_{j,k} + v_{i,k}\tilde{w}_{j,k} + v_{i,k}\tilde{v}_{j,k}) \\
&= 2 \cdot \text{fdiff} \cdot \frac{\partial}{\partial w_{i,k}} (w_{i,k}\tilde{w}_{j,k} + w_{i,k}\tilde{v}_{j,k}) = 2 \cdot \text{fdiff} \cdot (\tilde{w}_{j,k} + \tilde{v}_{j,k})
\end{aligned} \tag{6}$$

(analogically for other vectors and biases).

We have set $x_{\max} = 100$ and $\alpha = 3/4$ since these were the values that gave best results for GloVe.

By separating vectors responsible for meaning and grammar of words the model can easily group one type of vectors by semantics of words and second one by syntactics.

4. Experiments

4.1. Corpora and training details

For Polish language we have trained my model on a 2016 Wikipedia dump with over 350 billion words. We lowercased the corpus and removed punctuation marks with simple script, built a vocabulary of words appearing in the corpus at least 20 times (resulting with over 400 million words vocabulary). Then we built the co-occurrence matrix X using a window size 10.

For all experiments we set $x_{\max} = 100$ and $\alpha = 3/4$. The model was trained using asynchronous AdaGrad, stochastically sampling non-zero elements from X (co-occurrences are randomly shuffled after counting and before starting learning), with initial learning rate set to 0.05. We have trained our model for vectors of size 50, 100, 200 and 300.

4.2. Word analogies test

It has been observed that words with the same grammatical form or similar meaning tend to spread among subspace of original vector space. This suggests that some part of vectors might be responsible for certain features of words. Following this observation [6] has proposed new technique to measure quality of word vector representations. It intends not only to check whether similar words are close to each other but also to investigate multiple degrees of similarity.

Similarities of word vector representations apply not only to syntactic properties of words but also to semantic. [10] describe word offset technique that uses simple algebraic operations applied to vectors . For example, as we have mentioned in the introduction, *vector("king") - vector("man") + vector("woman")* results with a vector that is closest to the vector representation of word *queen*.

To examine these regularities Mikolov et al. has introduced the *word analogy test*. It consists both of semantic and syntactic tests divided into categories. Each file consists of one type of regularity. In each line there are two pairs of words

For our needs we have translated files with these tests to Polish and extended with tests for regularities that are absent in English. Examples of each categories are presented in Table 1. We have avoided multi-word entities (like *New York*). Overall there are 8869 semantic and 10000 syntactic questions.

We evaluate vectors accuracy for all question types and for each question type separately (semantic, syntactic). Question is assumed to be answered correctly if and only if the nearest word to the vector computed suing method described above is exactly the word from question.

GloVe model vectors score better results in the semantic part of the test. Learning separate vectors for base and grammatical forms of words gives the model extra

Type of relationship	Word Pair 1		Word Pair 2	
Common capital city	Ateny	Grecja	Oslo	Norwegia
All capital cities	Astana	Kazachstan	Harare	Zimbabwe
Currency	Angola	kwanza	Iran	rial
City-in-state	Chicago	Illinois	Stockton	Kalifornia
Man-Woman	brat	siostra	wnuk	wnuczka
Adjective to adverb	niezwykły	niezwykle	pozorny	pozornie
Opposite	stały	niestały	świadomy	nieświadomy
Comparative	zły	gorszy	miękki	miększy
Superlative	duży	największy	bliski	najbliższy
Nationality adjective	albania	albański	japonia	japoński
Past tense	tańczy	tańczył	zmniejsza	zmniejszał
Plural nouns	ptak	ptaki	butelka	butelki
Grammatical gender	jadł	jadła	znalazł	znalazła
Nominative-genitive	kalafior	kalafiora	pierwiastek	pierwiastka

Table 1. Examples of word analogies divided into types

information about morphology. It results with much better scores for syntactic part of the word analogy test but it also lowers scores for semantic part.

We have experimented with two ways of computing vectors for each word – adding vectors of its base form and tagset (BF + TS) and adding also vectors of the word itself (the one computed for the regular GloVe; BF + TS + word). Similar technique to the second one was used in [12]. Results are presented in Table 2.

Model	Dim.	Sem. [%]	Syn. [%]	Tot. [%]
GloVe	50	25.57	18.97	22.36
BF+TS	50	13.92	37.17	25.23
BF+TS+word	50	17.46	24.61	20.94
GloVe	100	46.47	25.36	36.20
BF+TS	100	22.10	41.46	31.59
BF+TS+word	100	31.57	37.77	34.59
GloVe	200	56.96	28.43	43.09
BF+TS	200	21.69	44.21	32.64
BF+TS+word	200	42.83	46.24	**44.49**
GloVe	300	**57.64**	28.64	43.54
BF+TS	300	20.15	43.46	31.49
BF+TS+word	300	40.97	**48.01**	44.39

Table 2. Accuracy for word analogy test

These results show that when splitting a word into its base and grammatical form the model tends to loose information about semantics of the word for its syntactic. Possible reason for that is lower freedom of placing vectors for the extended model than in the regular one. Both base forms vectors and tagsets vectors are shared by many various words what makes it much harder for the model to place word vector in most convenient location. Grammatical properties represented by tagsets are always

BF weight	TS weight	word weight	Sem. [%]	Syn. [%]	Tot. [%]
Regular GloVe			**57.64**	28.64	43.54
0.6	0.4	0	14.53	59.27	36.29
0.7	0.3	0	11.55	55.31	32.83
0.8	0.2	0	9.42	51.18	29.73
0.4	0.3	0.3	39.22	57.51	48.11
0.3	0.3	0.4	51.87	50.53	**51.22**
0.4	0.2	0.4	46.68	49.35	47.98
0.4	0.4	0.2	30.60	**61.57**	45.66

Table 3. Weighted vector sums accuracy for dimensionality 300

strictly defined and base forms can be ambiguous what might make tagsets vectors „dominate" base forms vectors. For this reason we have tested assigning different weights to base form, tagset and word vectors when summing these vectors. Results for vectors of dimensionality 300 are presented in Table 3. For comparison we have also included results for the regular GloVe in the first row.

Our model outperforms regular GloVe in the syntactic part of the test for any weights. Even though regular GloVe is still better for the semantic part of the test, our model receives almost equal score for some weights. What is also worth noticing is the fact that for weights 0.3, 0.3, 0.4 our model get best total result out of all combinations we have tested and also very good scores for both semantic and syntactic parts of the test. That gives us a model that balances between semantic and syntactic performance well.

4.3. Wordnet

For this test we have used the polish wordnet - *Słowosieć* [16]. Wordnet is a semantic dictionary reflecting lexical system of polish language. Entries of wordnet are connected by relations creating net. For example *car* is a type of *vehicle*, consists of *wheels*, *engine* and others and its synonyms include *automobile* and *truck*. *Słowosieć* has been created together by computer scientists and linguists.

For our tests we have used the shortest path distance in wordnet graph as word similarity measure. We have defined test set, consisting of 72 popular words and 64 rare words. For each of them we have found 5-20 closest word vectors. Then we have computed the average distance of their base forms (wordnet contains only base forms of words) in wordnet graph from starting word.

Like in the word analogy test, we have tested regular GloVe vectors, sums of base and grammatical forms and sums of all three. Since the wordnet test examines only semantic features of word vectors we have also tested it for base form vectors (for each word we have assigned vector of its base form as its vector). Results are presented in Table 4. We have tested it for 10 closest words.

As it was expected, best results were scored for base form vectors – this test examines only semantics of words so tagset vectors only "disturb". The model we

Model	Dim.	Popular words	Rare words	Total
GloVe	100	2.612	1.824	2.272
BF+TS	100	2.904	2.476	2.674
BF+TS+word	100	2.907	2.215	2.563
BF	100	0.593	0.605	0.599
GloVe	200	2.559	1.471	2.088
BF+TS	200	2.730	2.390	2.550
BF+TS+word	200	2.696	2.453	2.583
BF	200	0.434	**0.546**	**0.491**
GloVe	300	2.541	1.686	2.175
BF+TS	300	2.701	2.273	2.473
BF+TS+word	300	2.797	2.416	2.617
BF	300	**0.396**	0.595	0.498

Table 4. Accuracy for wordnet test for 10 closest words

have implemented has the advantage of gaining information about the meaning of word regardless of its grammatical form so no matter in what case the word occurs, it contributes to learning of the meaning of all of its forms.

5. Conclusion

In our work we have presented, implemented and tested methods to enhance word vectors performance. Feeding model with additional information about base and grammatical forms of words allows it to extract semantic and syntactic information better. It also allows the model to use the corpus more efficiently – occurrences of rare words in different forms are connected and we are able to gather more information about them. We have also translated to Polish set of questions for words analogy test prepared by [10] for testing my model and we have introduced new test basing on *Słowosieć* [16].

5.1. Future work

Several improvements can still be tested. One of them can be a syntactic tagging of corpus for word disambiguation. In our work for each word we have chosen its base and grammatical form not regarding its context. If its grammatical form was ambiguous a tagset was created.

Testing another models, like Skip-gram and CBOW [10], might also help to investigate capabilities of splitting word vectors. Although models implemented in Word2Vec and GloVe are somehow similar, the first ones are more complex what might give the

opportunity to capture language structure better.

Another methods of constructing word vectors from vectors of its base forms and tagsets could be tested too. [11] construct word vectors with small neural network, what gives the model more flexibility and makes the process of combining vectors learnable. Another, simpler method could be concatenating two vectors into one bigger (so if vectors of base forms and tagsets would be from $\mathbb{R}^n$ the resulting vector would be from $\mathbb{R}^{2n}$).

Word embeddings are useful as a way of preprocessing data. When used in bigger model it is hard to tell how the vectors affect the model exactly and which of its parts should be improved. Due to this fact the performance of our vectors on downstream tasks could be also tested and discussed in future. [17] present more methods for evaluating word embeddings.

Acknowledgements

The authors would like to acknowledge the support of the National Science Center (Poland) grant Sonata 8 2014/15/D/ST6/04402 and thank the Wroclaw Center for Networking and Supercomputing for donating computer time.

6. References

[1] Manning, C.D., Raghavan, P., Schütze, H., *Introduction to Information Retrieval.* Cambridge University Press, 2008.

[2] Sebastiani, F., *Machine learning in automated text categorization.* ACM computing surveys (CSUR), 2002, **34**(1), pp. 1–47.

[3] Tellex, S., Katz, B., Lin, J., Fernandes, A., Marton, G., Quantitative evaluation of passage retrieval algorithms for question answering. In: *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, 2003, pp. 41–47.

[4] Turian, J., Ratinov, L., Bengio, Y., Word representations: a simple and general method for semi-supervised learning. In: *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, 2010, pp. 384–394.

[5] Socher, R., Bauer, J., Manning, C.D., Ng, A.Y., Parsing with compositional vector grammars. In: *ACL (1)*, 2013, pp. 455–465.

[6] Mikolov, T., Yih, W.t., Zweig, G., Linguistic regularities in continuous space word representations. In: *HLT-NAACL*. vol. 13., 2013, pp. 746–751.

[7] Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J., Distributed representations of words and phrases and their compositionality. In: *Advances in Neural Information Processing Systems 26*, 2013, pp. 3111–3119.

[8] Pennington, J., Socher, R., Manning, C.D., Glove: Global vectors for word representation. In: *EMNLP*. vol. 14., 2014, pp. 1532–43.

[9] Bengio, Y., Ducharme, R., Vincent, P., Jauvin, C., *A neural probabilistic language model*. Journal of Machine Learning Research, 2003, **3**(Feb), pp. 1137–1155.

[10] Mikolov, T., Chen, K., Corrado, G., Dean, J., *Efficient estimation of word representations in vector space*. CoRR, 2013, **abs/1301.3781**.

[11] Luong, T., Socher, R., Manning, C.D., Better word representations with recursive neural networks for morphology. In: *Proceedings of the Seventeenth Conference on Computational Natural Language Learning, CoNLL 2013, Sofia, Bulgaria, August 8-9, 2013*, 2013, pp. 104–113.

[12] Botha, J.A., Blunsom, P., Compositional morphology for word representations and language modelling. In: *ICML*, 2014, pp. 1899–1907.

[13] Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R., *Indexing by latent semantic analysis*. Journal of the American Society for Information Science, 1990, **41**(6), pp. 391.

[14] Duchi, J., Hazan, E., Singer, Y., *Adaptive subgradient methods for online learning and stochastic optimization*. Journal of Machine Learning Research, 2011, **12**(Jul), pp. 2121–2159.

[15] Miłkowski, M., Polimorfologik. `https://github.com/morfologik/polimorfologik` 2016.

[16] Maziarz, M., Piasecki, M., Szpakowicz, S., Approaching plWordNet 2.0. In: *Proceedings of the 6th Global Wordnet Conference*, January 2012, ,.

[17] Schnabel, T., Labutov, I., Mimno, D.M., Joachims, T., Evaluation methods for unsupervised word embeddings. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, 2015, pp. 298–307.

Schedae Informaticae Vol. 25 (2016): 139–151
doi: 10.4467/20838476SI.16.011.6192

tfml 2017
theoretical foundations
of machine learning, Kraków

A Translation Evaluation Function based on Neural Network

Ameur Douib, David Langlois, Kamel Smaïli
Université de Lorraine, Loria, Campus Scientifique,
BP 239, 54506 Vandoeuvre-lès-Nancy, France
e-mail: *ameur.douib@inria.fr, david.langlois@loria.fr, kamel.smaili@loria.fr*

Abstract. In this paper, we study the feasibility of using a neural network to learn a fitness function for a machine translation system based on a genetic algorithm termed GAMaT. The neural network is learned on features extracted from pairs of source sentences and their translations. The fitness function is trained in order to estimate the BLEU of a translation as precisely as possible. The estimator has been trained on a corpus of more than 1.3 million data. The performance is very promising: the difference between the real BLEU and the one given by the estimator is equal to 0.12 in terms of Mean Absolute Error.

Keywords: Statistical Machine Translation, Genetic algorithm, Quality estimation, Neural network

1. Introduction

Nowadays, a lot of Statistical Machine Translation (SMT) systems use a Beam-search algorithm [1] in order to retrieve the best possible translation by taking into account different scores provided by several models: language, translation, distortion, etc. Starting with an empty set, the solution building process consists in producing incrementally a set of complete solutions from partial ones provided by a translation table (TT). Because the translation is built incrementally, it is then difficult to challenge a previous decision of translation, which can eliminate a partial hypothesis, even if it

Received: 11 December 2016 / Accepted: 30 December 2016

could propose a good final solution.
An alternative to this algorithm is to start with a complete translation hypothesis and try to refine it in order to retrieve the best solution. With complete translation hypotheses, it is possible to revisit each part of the research space and modify it, if necessary.
GAMaT [2] is a new decoder for SMT based on a genetic algorithm. It has the advantage to refine several complete solutions in an iterative process and produce acceptable solutions. In fact, a possible solution is encoded as a chromosome, where the chromosome encloses several information (the source sentence segmented into phrases, a translation hypothesis also segmented into phrases, and alignment between source and target segments). Then, from a population of chromosomes, we estimate their fitness (score) in order to keep them, or not, for next generations. To do so, the fitness is a combination of several scores measuring how the different segments of a chromosome are coherent with each others. Nine scores corresponding to nine features are combined to score translations [2]. A weight proportional to the impact of the feature on the evaluation function is assigned to each feature. This combination has been held by a log-linear approach. In GAMaT, the weights corresponding to the nine scores are provided by Moses [1]. According to the BLEU [3] metric results, the translation performance is good, but not better than Moses.
To get away from Moses and to propose a relevant solution for GAMaT, we propose to learn the function of the chromosome evaluation by using a neural network (NN) which predicts a BLEU value of the translation represented in a chromosome. In other words, we would like the fitness function to be correlated to BLEU. The NN is learned on the nine features mentioned before. We opted for BLEU metric because it is commonly used in the MT community to evaluate translations. This kind of learning algorithm is used in the Quality Estimation (QE) community [4], in order to estimate the translation quality without access to the reference translation.
The article is structured as follows. We present the related works in Section 2. In Section 3 we describe the chromosome features. In Section 4 we present the NN used to learn the fitness function. Then, in Section 5 we describe how we generate the dataset for the NN. We give the results of several configurations in Section 6. Finally, we conclude and give some analyses and perspectives.

2. Related works

In this paper, we propose a new translation evaluation function for phrase-based SMT decoders, and which is applied for our genetic-based decoder GAMaT [2]. This function is learned, using a neural network, on nine chromosome features and correlated with the BLEU value of the translation enclosed in the chromosome.
Therefore, our work can be classed at the intersection of two research disciplines; The former is the optimization of decoder parameters for machine translation [5, 6]. Where, for the majority of decoders, the objective function combines log-linearly a set of translation features to evaluate the translation hypotheses. Optimising the weights

of this function allows a better translation accuracy. In the machine translation community to optimise these weights, the proposed algorithms are largely based on a grid search algorithm [6]. Where the goal is to find the best set of weights which minimise a loss function adapted for the translation process [5].
The latest domain is the Quality Estimation (QE). The main goal in this area is to estimate the translation quality without access to the reference translation [4, 7]. To this end, in QE community, machine learning algorithms are trained on features extracted from pairs of source sentences and their translations, and they are correlated with an evaluation metric, which can be a metric with binary values (good/bad), or a metric with continuous values (BLEU, TER, ...etc.). This can be done by different machine learning algorithms such as SVM regression [8] or a Recurrent Neural Network [9].
In this work, we use a neural network such as in QE community (see Section 4.), to combine optimally the features used in the log-linear approach (see Section 3.). The learned function estimates the quality of the translation by predicting its BLEU value, without having access to the reference translation, which is the case at decoding time.

3. The features of the chromosomes

In order to evaluate the relevance of a chromosome, we need to evaluate it by combining its different features, such as what was done in [2]. The features are log-linearly combined as follows:

$$Score(c) = \sum_{i} \lambda_i \times \log(h_i(e, f)) \tag{1}$$

Where f is the source sentence and e is the translation represented in the chromosome c. λ_i is the weight of h_i determined by Moses and h_i is the score related to the i^{th} feature. The value of each weight defines the influence of the corresponding feature in the final score. Nine features related to the construction of a chromosome have been used and are described in the following.

- F_1: a language model feature, which estimates how the translation e is linguistically correct in the target language. In practice, F_1 is estimated by the likelihood $P(e)$ of a translation yielded by the classical n-gram. In our experiments, n is set to 3.

- F_2: the second feature concerns the translation probability. Given a chromosome c, F_2 is based on the alignment presented in c. This alignment links each source phrase to a target phrase. Then, F_2 is the product of translation scores (from the source language towards the target one, given by the translation table) between linked source and target phrases. This feature is called the direct translation probability.

- F_3: this feature concerns the inverse translation probability, which is equivalent to the previous one, except that the translation scores are from the target

language towards the source one.

- F_4: this feature estimates the quality of a pair of segments at word level [10]. It is defined as the product of lexical probabilities inside one segment and over all the segments of the source sentence. This feature is called a direct lexical probability.

- F_5: symmetrically to F_4, an inverse lexical probability is estimated.

- F_6: this one concerns the length of the target sentence in the chromosome to produce. In fact, the translation should not be too much longer than the source sentence. This feature is set with the difference between the length of the source and the translation in terms of words.

- F_7: to reinforce the previous feature, a length model is trained [2] that assigns a probability to a pair of sentences depending on their lengths. In other words, this feature is estimated as the probability that a source sentence with a length l_s, will produce a translation of length l_t.

- F_8: longer sequences are linguistically more informative than smaller ones. Therefore, a chromosome with a few number of phrases should give a better translation. This feature is called the phrase penalty and is estimated as an exponential function of the number of phrases: e^k, where k is the number of phrases in the chromosome.

- F_9: it is the cost of permutations in the translation at the phrase level, when the target phrases are picked out of order [1].

To estimate some of the previous features, we need a translation table, which contains pairs of source and translation phrases with their associated probabilities.

4. A machine learning algorithm for prediction

As presented in the introduction, the goal of this work is to propose a new chromosome fitness function which combines optimally the features previously presented, and which must be correlated with BLEU. To do so, we decided to train a supervised neural network which takes as input the nine features of a chromosome, and as label in output the BLEU value of the corresponding translation (Figure 1). The learned fitness function is supposed to predict the BLEU of the translation of a chromosome.

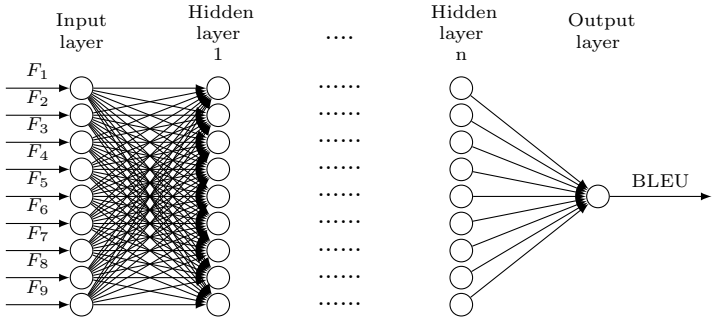


Figure 1. The neural network architecture

We experimented different configurations of the neural network by varying the number of layers and the number of neurons. We used the Sigmoid function for the neuronal activation. The experiments have been conducted by using Keras, a Deep Learning library [11].

5. The dataset for Neural Network

To learn the neural network, a training corpus is necessary. In the following, it will be composed by an important number of pairs $< v_c, \alpha >$. Where v_c is a vector of 9 features which characterize a chromosome c and α represents the BLEU value of the translation hypothesis represented in the chromosome c. To this end, we used the French-English parallel corpus of the 9^{th} task workshop on SMT [12]. We used this corpus to produce the translation table TT which is necessary to build the chromosomes and also to calculate some features of chromosomes.
The corpus contains pairs of French sentences and their English reference translations. We split it into two sets: the former, containing 1,323,382 pairs, is used to build the TT handled by GIZA++ [13] and the language model; while the latter set (C_{NN}), containing 165,422 pairs, is used to produce chromosomes.
For each source sentence in C_{NN} we built a set of chromosomes, e.g. a set of hypotheses with the segmentation of source and target sentences, and the alignment between source and target segments. We have to produce chromosomes representing perfect, good and bad translations in order to diversify the dataset because the fitness must predict correctly the translation quality of chromosomes, whatever this quality is. To produce chromosomes, we used several functions of GAMaT [2] which take a source sentence and produce one or more chromosomes.

The methods used to build chromosomes are presented in the following:

- From a source sentence, the builder of chromosomes proposes a segmentation based on the longest phrase within it. This longest phrase can be found anywhere in the source sentence. This phrase is picked up from the TT. The seg-

mentation process is iteratively repeated until the whole sentence is segmented. Then, each source phrase is translated in the target language by choosing the most likely translation from TT. Source sentence and hypothesis have same phrase ordering. This produces one chromosome.

- The process is the same as the previous method, but the source sentence is segmented from left to right. This produces one chromosome.
- The process is the same as the previous method, but the source sentence is segmented from right to left. This produces one chromosome.
- Here, the source sentence is randomly segmented, where all the produced segments must exist in the TT. This random segmentation is done several times, in order to increase the number of chromosomes. This allows to increase the size of the training corpus.
- In this one, the builder performs similarly such as in the previous point, except that instead of selecting the best translation for a source phrase, it chooses randomly from TT one from the possible translations of this phrase.
- The producer proposes chromosomes such as in the previous point, except that the alignment between a source phrase and a target segmentation is not monotone: target phrases are mixed. This allows to build several bad chromosomes.
- For each source sentence, the producer of chromosomes keeps the best solution proposed by GAMaT. This allows to produce for each source sentence one chromosome with a good quality translation.
- Similarly to the previous one, the producer of chromosomes keeps the best solution proposed by Moses for the source sentence.

Using these methods, we produced 1,5 million of chromosomes, from 165,422 source sentences. For each of them, we produced a set v_c of 9 features, and we calculated α, which is the BLEU value in this work. The BLEU estimates the similarity between the hypothesis and the reference translation. This produced a training corpus for the NN composed of a set of pairs $< v_c, \alpha >$.

6. Experiments and results

In this section, we describe our experiments. We describe the different training and test corpus we used, and how we measured the performance of the neural network.

6.1. Evaluation metric

To evaluate the precision of the prediction function, the Mean Average Error (MAE) criterion [14] is calculated on the test set. The error is the difference between the real

BLEU and the predicted value:

$$MAE_{BLEU} = \frac{1}{n}\sum_{i=1}^{n} |BLEU_r(c_i) - BLEU_p(c_i)| \quad (2)$$

Where $BLEU_r(c_i)$ is the real BLEU value of the translation in the chromosome c_i, and $BLEU_p(c_i)$ is the predicted BLEU value for the same translation. n represents the size of the test set.

6.2. Training and test corpus

In order to build the training and test corpus, we followed the process we described in Section 5. We segmented this corpus into a training and a test part respectively 90% and 10%. We show in Figure 2 the distribution of this corpus according to the BLEU value. This figure shows the proportion of chromosomes for five BLEU intervals: [0,0.25[(very bad translation according to reference), [0.25,0.5[, [0.5,0.75[, [0.75,1[and BLEU=1 (perfect translation). The bar *Unbalanced training set* corresponds to the proportions of the different qualities of translations presented above. The bar referenced as *Test set* corresponds to the proportions of the test corpus for which we keep the same distribution as in the training corpus. This corpus is not balanced, this could be a drawback, because the NN should estimate the translation quality with the same precision, whatever the correctness of a chromosome is. That is why, we trained also the NN with a balanced training corpus (the bar *Balanced training set*). In GAMaT, at the translation time, the distribution presented above is not necessarily respected because the set of chromosomes (population) evolves at each iteration. Therefore, we achieve also experiments with another test corpus (bar *GAMaT test set* in Figure 2) built from a translation corpus used in [2]. This test corpus is representative of chromosomes obtained at end of the real translation process. This test corpus contains 50,000 chromosomes built from 1,000 source sentences. It contains very few perfect translations, and relatively more very bad translations compared to *Test set*.

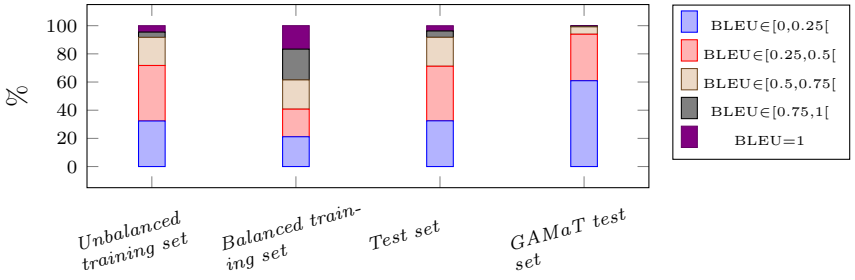


Figure 2. Distribution of data in the training sets and test sets.

In the following, we present the performance of the estimator on *Test set* and *GAMaT test set*, trained with *Unbalanced training set* and *Balanced training set*.

6.3. Impact of the neural network architecture

In this section, we study the performance of the NN according to the number of layers (1 to 7) and neurons (8, 16, 32 and 64) in each layer. Figure 3 shows the MAE performance on *Test set* according to the number of layers, and to the number of neurons. In the left curve, for each number of layers the plot is the average performance according to the number of neurons. In the right curve, for each number of neurons the plot is the average performance according to the number of layers. The figure shows that the best results are achieved for 4 layers and 32 neurons by layer.

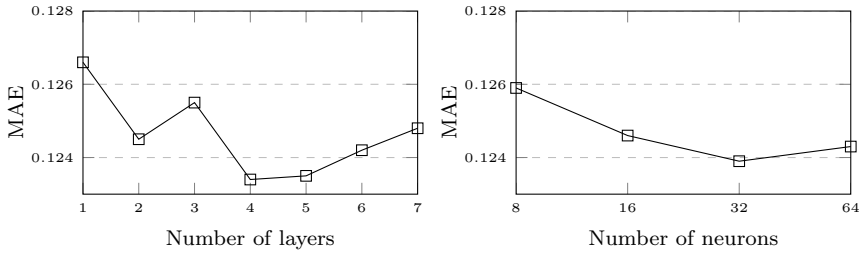


Figure 3. The influence of the number of layers and neurons in the MAE score.

As these values are only averages, we give in Figure 4 the performance for each couple (number of layers, number of neurons by layer). This figure shows that the performance are better when the number of neurons grows with the number of layers. In the following experiments, we keep the best average configuration: 4 layers, and 32 neurons by layer.

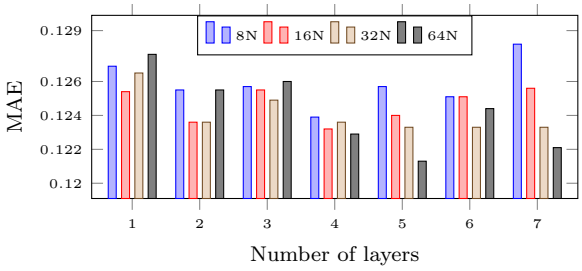


Figure 4. Details of the influence of the number of layers and neurons in the MAE score.

6.4. Impact of the size and quality of the training corpus

In machine learning, the size of the training data is crucial, and the test data should not be very different from the training data. To respect this constraint, we increased the training set, from 225,000 to 2,250,000 with and without paying attention to the proportion of the quality of the set of chromosomes (Figures 5 and 6). Figure 5-(a)

shows the performance on *Test set* and *GAMaT test set* trained with several sizes of the *Unbalanced training set*.

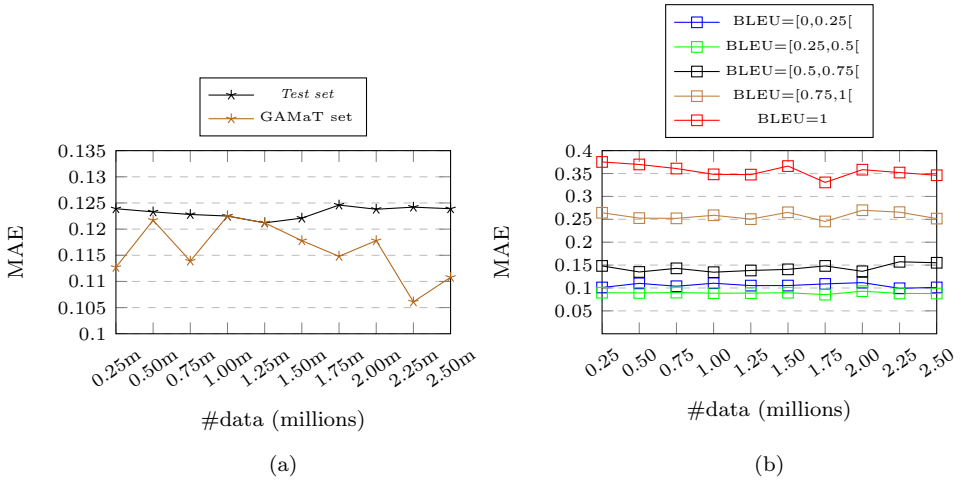


Figure 5. The influence of the number of data in *Unbalanced training set*.

The performance on the corpus *Test set* is not affected by the size of the training corpus according to the results of Figure 5. On the contrary, the curve for *GAMaT test set* is more chaotic but the performance of the estimator tends to be better, as the size of the training increases. Figure 5-(b) shows that the estimator produces bad results for chromosomes for which the BLEU is greater than 0.75 and estimates correctly the others. This is probably due to the fact that bad chromosomes are more numerous than the good ones in *Test set*.
In Figure 6-(a), we did the same experiments, but with *Balanced training set*. The performance with good chromosomes is henceforth better, since the training corpus is more balanced.

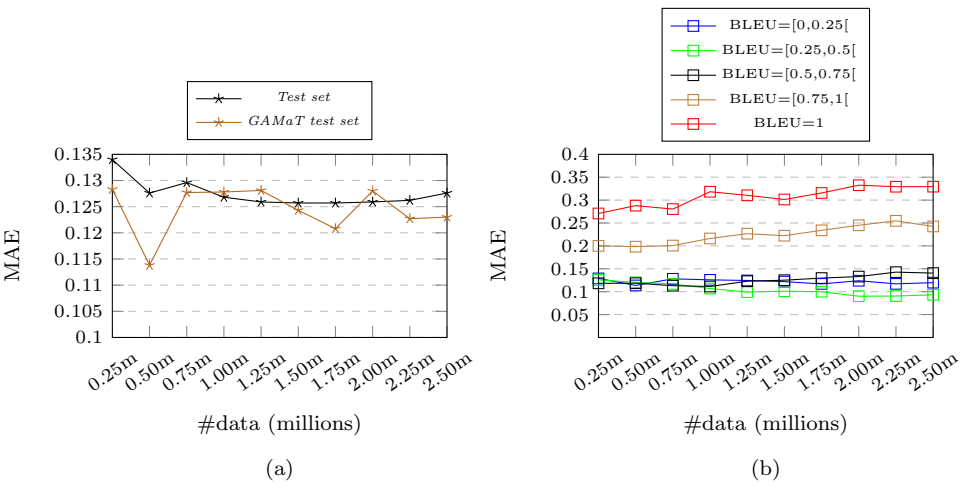


Figure 6. The influence of the number of data in *Balanced training set*.

As in the previous figure, we give more details about MAE performance for test subsets according to translation quality in Figure 6-(b). This figure shows that balancing the training corpus allows to improve the prediction performance for good quality translations, but this is not sufficient to improve the overall performance because these good translations are not numerous in the *Test set*.

6.5. Results and Discussion

The results in the different campaigns of Quality Estimation are very close to each other [4]. Our results obey to this rule. Two main results emerge from this study. For a training corpus not selected smartly, in other words for the *Unbalanced training set*, the estimator needs 2.25 million of chromosomes to reach the best result 0.1061 (Table 1) on *GAMaT test set*. While for the *Test set*, we need only 1.25 million of chromosomes for a MAE of 0.1212. In spite of these low scores (lower is better for MAE), the estimator evaluates badly the best hypotheses of translation, for which the BLEU is greater than 0.75 (see Figure 5-(b)). When we use *Balanced training set*, we need only 0.5 million of chromosomes to reach the best performance for *GAMaT test set* with a MAE of 0.1138. The best result for the *Test set* is obtained for 1.5 million of chromosomes which leads to a MAE of 0.1257. With this *Balanced training set*, the best hypotheses are better evaluated than with the first training set (see Figure 6-(b)), but we lose in the quality estimation of the bad hypotheses, which are more numerous in the real translation process. This explains the fact that the performance of *Unbalanced training set* is better than those of *Balanced training set*.

		Test set	
Training set	**# of data**	**Test**	**GAMaT**
Unbalanced	1.25m	**0.1212**	0.1212
	2.25m	0.1242	**0.1061**
Balanced	1.50m	**0.1257**	0.1243
	0.50m	0.1276	**0.1138**

Table 1. Best MAE scores for *Unbalanced training set* and *Balanced training set*.

To study the behavior of the proposed evaluation function in a real translation process, we used it as a chromosome evaluation function in our SMT decoder (GAMaT) in order to translate a set of 1.000 source sentences. In Table 2 we present some translation results according to the BLEU metric [3]. For each training size, the fitness function we used is the one which led to the lowest MAE score. The best translation performance are obtained when we trained the neural network on 1.250.000 data with 5 hidden layers and 64 neurons in each layer. This configuration allowed us to achieve 21.05 in BLEU.

	Best configuration used		
Size of training set (Millions)	**# of hidden layers**	**# of neurones**	**BLEU**
0.25	5	32	**17.91**
0.50	7	20	**18.34**
0.75	6	64	**19.80**
1.00	5	64	**20.15**
1.25	5	64	**21.05**
1.50	6	64	**18.56**
1.75	3	64	**16.27**
2.00	4	64	**17.94**
2.25	5	64	**17.98**

Table 2. Translation performance according to BLEU using the proposed function.

Through these results, we notice the fact that the size of the training corpus leading to the best performance in terms of BLEU is the same than in terms of MAE (see Table 1). The best configuration uses 5 hidden layers and 64 neurones, and these number of layers and neurones lead also to the best configuration in terms of MAE (see Figure 4).
Therefore, improving the quality of the learned function ensures to improve the final translation quality.

7. Conclusion and perspectives

In this paper, we investigated the use of a new function to evaluate chromosomes in GAMaT. The function is an estimator of BLEU for a chromosome. This estimator has been trained by a neural network which is learned on nine features extracted from a set of chromosomes built by GAMaT.
The experiments show that the amount of the training data, and their distribution in terms of translation quality, impact the precision of the function. The prediction function achieved convincing results. In fact, the MAE calculated on a test set of 100,000 chromosomes is around 0.12. Also, the use of this function in GAMaT such as a fitness function shows that improving the fitness in terms of MAE improves the translation accuracy in real translation conditions.
The presented method uses BLEU for evaluating a translation, but could be extended to other measures which might be combined in order to get a more robust fitness function. To improve this prediction function, sampling techniques should be used to select more representative data for the training.

References

[1] Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., et al., Moses: Open source toolkit for statistical machine translation. In: *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*, Association for Computational Linguistics, 2007, pp. 177–180.

[2] Douib, A., Langlois, D., Smaili, K., Genetic-based decoder for statistical machine translation. December 2016, ,, Nous n'avons pas encore la date officielle de publication.

[3] Papineni, K., Roukos, S., Ward, T., Zhu, W.J., BLEU: a method for automatic evaluation of machine translation. In: *Proceedings of the 40th annual meeting on association for computational linguistics*, Association for Computational Linguistics, 2002, pp. 311–318.

[4] Bojar, O., Chatterjee, R., Federmann, C., Haddow, B., Hokamp, C., Huck, M., Logacheva, V., Pecina, P., eds. *Proceedings of the Tenth Workshop on Statistical Machine Translation*. Association for Computational Linguistics, September 2015, ,.

[5] Neubig, G., Watanabe, T., *Optimization for statistical machine translation: A survey*. Computational Linguistics, 2016.

[6] Och, F.J., Minimum error rate training in statistical machine translation. In: *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, Association for Computational Linguistics, 2003, pp. 160–167.

[7] B., O., et al., eds. *Proceedings of the First Conference on Machine Translation*. Association for Computational Linguistics, August 2016, ,.

[8] Langlois, D., Loria system for the wmt15 quality estimation shared task. In: *Proceedings of the Tenth Workshop on Statistical Machine Translation*, Lisbon, Portugal, September 2015, ,, pp. 323–329.

[9] Kim, H., Lee, J.H., A recurrent neural networks approach for estimating the quality of machine translation output. In: *Proceedings of NAACL-HLT*, 2016, pp. 494–498.

[10] Koehn, P., Och, F.J., Marcu, D., Statistical phrase-based translation. In: *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, Association for Computational Linguistics, 2003, pp. 48–54.

[11] Bergstra, J., Bastien, F., Breuleux, O., Lamblin, P., Pascanu, R., Delalleau, O., Desjardins, G., Warde-Farley, D., Goodfellow, I., Bergeron, A., et al., Theano: Deep learning on gpus with python. In: *NIPS 2011, BigLearning Workshop, Granada, Spain*, Citeseer, 2011.

[12] Bojar, O., Buck, C., Federmann, C., Haddow, B., Koehn, P., Leveling, J., Monz, C., Pecina, P., Post, M., Saint-Amand, H., et al., Findings of the 2014 workshop on statistical machine translation. In: *Proceedings of the Ninth Workshop on Statistical Machine Translation*, Association for Computational Linguistics Baltimore, MD, USA, 2014, pp. 12–58.

[13] Och, F.J., Ney, H., *A systematic comparison of various statistical alignment models*. Computational linguistics, 2003, **29**(1), pp. 19–51.

[14] Willmott, C.J., Matsuura, K., *Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance*. Climate research, 2005, **30**(1), pp. 79–82.

Schedae Informaticae Vol. 25 (2016): 153–164
doi: 10.4467/20838476SI.16.012.6193

tfml 2017
theoretical foundations
of machine learning, Kraków

Data Selection for Neural Networks

Miroslaw Kordos
Department of Computer Science and Automatics
University of Bielsko-Biala
Willowa 2, 43-309 Bielsko-Biala
e-mail: *mkordos@ath.bielsko.pl*

Abstract. Several approaches to joined feature and instance selection in neural network leaning are discussed and experimentally evaluated in respect to classification accuracy and dataset compression, considering also their computational complexity. These include various versions of feature and instance selection prior to the network learning, the selection embedded in the neural network and hybrid approaches, including solutions developed by us. The advantages and disadvantages of each approach are discussed and some possible improvements are proposed.

Keywords: Neural Networks, Data Selection, Feature Selection, Instance Selection

1. Introduction

There are three main purposes of data selection: limiting the dataset size and thus accelerating the model learning process, removing noise from the data and thus improving the model predictive capabilities and making the data interpretation easier by humans [1, 2]. In this paper we discuss how data selection can be addressed in neural network learning. Since datasets consists of instances and the instances consist of features, the dataset size can be reduced by feature selection, instance selection or both. Moreover, the selections can be performed as well prior to neural network

Received: 11 December 2016 / Accepted: 30 December 2016

learning as by the neural network itself during the learning process. Our purpose is to show some interesting properties of data selection obtained with different feature or different instance selection methods, propose some improvements and discuss how to choose the optimal method.

Typically in data selection we fist obtain a little improvement of the prediction accuracy as we remove some data and then the accuracy slightly but constantly drops as more data is removed. Then after exceeding some limit the prediction accuracy begins constantly dropping and if we want to go further with compression we must choose some point depending on our preferences on the compression-accuracy plot.

Feature selection prior to model learning can be done either with filters or with wrappers. Wrappers really wrap the learning algorithm so it is not strictly done before the learning process but rather before deciding which will be the optimal configuration of the final model used for prediction. For the experiments we carefully chose some filter methods (section 2.).

Instance selection is discussed in this paper is more details than feature selection and we examine some possible improvements and parameterization of instance selection methods (section 3.).

We tried to answer the question: what is better feature selection or instance selection or both, and if both then how feature and instance selection influence each other and how they should be applied together for the best results (section 4.).

Another approach is not to perform any preliminary data selection but let the neural network select the relevant data itself. That includes selecting features (section 5.), instances (section 6.) and both of them (section 7.). That of course required us to make some adjustments of the neural learning process and error function.

Finally we experimentally compare the discussed approaches and their results (section 8.) and present the conclusions from this study (section 9.).

2. Feature Selection Before Network Learning

A detailed discussion of feature filters and wrappers can be easily found in literature [3, 4]. When the expert knowledge is available it can be used to make some preliminary feature selection [5, 6]. To select the methods we are going to use, we first performed some preliminary experiments with different feature selection and different instance selection methods using the RapidMiner software [7] and then we chose the methods that were among the best in term of the balance between compression, classification accuracy and running time.

W tested the following feature filters available in RapidMiner: Information Gain, Information Gain Ratio, Deviation, Chi Squared, Gini Index, Uncertainty, SVM, PCA, Correlation, Rule, Relief. We also tested three wrappers: forward selection, backward selection and evolutionary selection. Although the backward selection wrapper and evolutionary selection were able to discover more informative feature subsets, resulting in bit higher classification accuracy with the same number of features, their execution time was between two and four orders of magnitude longer, what in the case of the biggest data sets was totally impractical for our purposes. The results of the filter evaluation are presented in Table 1 in terms of the average classification

accuracy over the 10 datasets obtained in 10-fold crossvalidation and the average relative calculation time. This was done for the number of features being the nearest integer to 60% and 30% of the original number of features. Based on the test results, the SVM-based filter produced the best accuracy, but for the further experiments we chose the second filter in the accuracy ranking: Information Gain, because the SVM-based filter was over 100 times slower.

The information gain criterion IG is defined as the difference between the entropy before and after the optimal data split on feature f:

$$IG_f = -\sum_{i=1}^{C} p(c_i) \cdot log(p(c_i)) + \sum_{n=1}^{N} [\frac{v_n}{v} \sum_{i=1}^{C} p(c_{ni}) \cdot log(p(c_{ni}))] \quad (1)$$

where $p(c_i)$ is the probability that an instance belongs to class i and $p(c_{ni})$ is the probability that an instance within the bin n belongs to class i, v is the number of all instances and v_n is the number of instances in bin n, C is the number of classes and N is the number of bins. We did not use any methods that create new features (as PCA), because that makes the data interpretation and logical rule extraction in most cases totally impractical; the complexity of the obtained rules exceeds human capabilities of making any logical conclusions, what is important in most of our practical applications.

Table 1. Average values over the 10 datasets of classification accuracy of neural networks for the nearest integer of 60% and 30% of features (F60-acc, F30-acc) and execution time relative to Information Gain time with different feature filter methods using the RapidMiner implementation.

Method	F60-acc	F30-acc	time
no selection	92.74	92.74	0.0
Information Gain	92.12	91.02	1.0
Information Gain Ratio	92.37	89.80	1.0
Deviation	91.78	88.37	0.2
Chi Squared	91.82	90.48	0.8
Gini Index	92.07	89.52	1.1
Uncertainty	91.82	91.04	1.9
SVM	93.01	91.24	102
PCA	92.51	89.13	0.5
Correlation	89.35	87.40	0.1
Relief	93.02	88.27	245
Rule	92.15	88.44	16

3. Instance Selection Before Network Learning

Instance selection methods fall into two categories: noise filters and condensation methods. Noise filters remove noisy instances and thus improve the classification accuracy, but they compress the data very little. Condensation methods compress

the data and speed up the neural network (or any other classifier) training effectively but usually also decrease classification accuracy [8]. Frequently stronger compression causes stronger accuracy loss. If methods from both families are used then noise filters should be used first.

In the experiments, we first evaluated the performance of the instance selection algorithms implemented in the Information Selection extension for RapidMiner to decide, which one to use. Again our criteria were similar as with feature selection: the method should produce high compression, low accuracy loss and have short running time.

The description of many instance selection algorithms can be found in [9, 10]. The experimental comparison was also done in [11]. However, the authors used only small datasets, and thus, as the general tendency is preserved between their and our study, we obtained different results for our much larger datasets.

Also some methods of evolutionary-based instance selection were proposed in [12, 13, 14]. However, the computational time was several order of magnitude longer and for this reason we did not take them into account, although the authors reported their method more accurate. Another shortage of the evolutionary-based methods is that the selection is performed without letting us understand while particular instances are selected.

We evaluated the following instance selection algorithms using the RapidMiner Information Selection Extension: GE, RNG, CNN, IB2, IB3, DROP-3, RHMC, MC, ENN (which is the only noise filter on that list) and ENN. In the case of feature filters the results were very similar while averaged over the 10 datasets and thus selecting the proper feature filter was not so crucial. However, in the case of instance selection algorithms the differences are really very significant and here the proper choice is much more important.

Two of the best performing instance selection algorithms were ENN (Edited Nearest Neighbor) followed by IB3 [**?**] and DROP-3 [15]. However, they did not always provided the best accuracy (in some points ENN followed by ENN or even GE was better) and therefore we decided to use ENN with IB3 and ENN with a modified CNN (Condensed Nearest Neighbor).

The IB3 works in a similar ways as CNN. It takes the instances misclassified by k-NN and then it removes from the selected set the instances, which can be removed without the loss of classification accuracy.

DROP-3 [15] is a decremental algorithm, which first implements ENN and then examines which instances can be removed while their neighbors are still correctly classified without them. The examination is not in random order but staring from the instances situated furthest from other class instances (so called "enemies").

The compression of ENN with IB3 and DROP-3 was two times stronger than that obtained with ENN followed by CNN, but the accuracy was comparable. DROP-3 was situated almost on the same point on the compression-accuracy plot as ENN followed by IB3. Finally we decided to use first IB3 and then ENN with CNN. ENN removes noise by removing the instances which have a different class than predicted by the k-NN algorithm. Then CNN or IB3 removes the instances that can be eliminated without adversely affecting classification.

One of the basic problems with the condensation algorithms such as CNN, IB3 or the DROP family, is that there is usually no direct way to control how aggressively

they perform the instance elimination, unlike in feature filters, where we can select the desire numbers of remaining features. Thus we used two approaches to overcome the limitations: first bagging of instance selection methods [16] and in this work modification of the rejection criterion in the algorithm itself.

Table 2. Average values over the 10 datasets of classification accuracy of neural network, number of selected instances and execution time relative to CNN time of instance selection process for different instance methods using the RapidMiner implementation.

Method	accuracy	%Instances	time
no selection	92.74	100	-
GE	90.71	45	130
RNG	86.81	12	10
CNN	86.74	8.0	1.0
IB2	85.12	7.7	0.2
IB3	86.56	4.0	3.5
DROP-3	87.13	4.0	14**
MC	82-86*	3.5-20	8.0
RHMC	82-86*	3.5-20	8.1
ENN	93.17	90	1.0
ENN+CNN	87.44	7.1	2.0
ENN+IB3	87.15	3.9	4.5
ENN	93.17	90	1.0

* - MC and RHMC provide various results depending on adjustable parameters, but for the same number of instances they always provided lower accuracy then the four best algorithms.

** - DROP-3 was implemented as a Weka plugin, so the time comparison may not be adequate in this case.

In the case of bagging we use the same idea as bagging in classification, but instead of classifiers, instance selection algorithms constitute the ensemble. Then we establish a threshold of how many instance selection algorithms from the ensemble want to select a given instance [16]. Say, we had 10 algorithms. We decide that if $m = 5$ of them want to remove a given instance then the instance will get finally removed. But we can also use any arbitrary number for m between 1 and 10. The higher m the less aggressive is the selection - the compression will be weaker but the accuracy will be higher (which corresponds to a higher number of attributes kept in feature selection).

In our tests bagging worked very well, but its drawback was higher computational cost. Thus the other solution is to use variable m in the inner k-NN algorithms within CNN (and ENN - although it is not so crucial). For example, we can use the number of nearest neighbors $k = 9$. In the standard CNN an instance will get removed if it has the same class as more then $k/2$ of its neighbors, that is $m = 5$ in that case. If we increase m say to 8 then the instance will get removed if it has the same class as more then $m = 8$ of its neighbors, so the selection will be less aggressive and some of the instance situated close to class boundaries that with $m = 5$ would get removed will be

kept [17]. Again the compression will be weaker but the accuracy will be higher. This is shown in the pseudocode, where kNN(.) is the class of at least m of k neighbors in the k-NN algorithm and $C(\mathbf{x}_i)$ is the real class of instance $\mathbf{x}_i$. In particular k and m can be different for ENN and CNN.

Algorithm 1 The modified ENN+CNN algorithm

```
Input: original set T
Output: reduced set S (now empty)
for i = 1 ... numInstances do
  if C(x_i) ≠kNN(k, m, (T without x_i), x_i) then
    mark x_i for removal
  end if
end for
remove all marked instances from T
add randomly one instance from T to S
for i = 1 ... numInstances in T do
  if C(x_j) ≠kNN(min(k, sizeOf(S)), min(m, sizeOf(S)), S, x_i) then
    add x_i to S
  end if
end for
return S
```

4. Joined Feature and Instance Selection Before Network Learning

Our experiments with data selection as the preprocessing step showed that feature selection (FS) should be performed prior to instance selection (IS). We tested many different configurations, such as FS-IS, IS-FS, FS1-IS1-FS2-IS2, FS1-IS-FS2 and others. Also in the case of repetitive feature and instance selection we tried making the reduction stronger at each iteration. However, comparing that to the simplest strategy FS-IS, there was no significant difference on the compression-accuracy plot. Our explanation of that phenomenon is that in most data there is a higher percentage of irrelevant features than of noisy instances (most of the instances were removed because of their redundancy and not because of noise). Thus, we can perform efficiently feature selection using all instances, but less efficiently instance selection using all features. Moreover, several feature filters are based on some measure of correlation or some variants of information gain. Removing too many instances can make them work less efficiently. On the other hand most instance selection methods are based on the distance between the instances. If there are irrelevant features, we get not the optimal distance measures. A partial solution to that problem is multiplying the distance component in each feature direction by this feature importance obtained from some feature ranking.

Some papers proposed evolutionary optimization of feature and instance selection [18, 14], but as already mentioned in section 2., we did not consider this option

first because of the computational complexity and second because evolutionary optimization, similarly as feature construction methods (as PCA) does not enable us to understanding why particular results were obtained.

5. Feature Selection Embedded into Network Learning

Feature selection with neural networks can be done in several ways. The two basic approaches are by the analysis of weights, including pruning methods and by input data perturbances [19]. In perturbance analysis we replace the values of particular feature with random values in the test vectors and see how this influences the network accuracy. In weight analysis we assume that the less important features will generate smaller absolute values of weights and we can reject the features with the lower weighted sum of weights r_i (Eq. 2.). The weights can be also enforced to small values with a regularization term. The weight analysis was used in our experiments. A more complex method also consider the derivatives or the output neuron weights. Due to the non-linear transfer functions the results depend on the actual position on the transfer function and in classification task at the end of the training the position is predominantly in the saturated area, so there must be some more effort put into constructing an efficient solution. We used the following feature ranking measure:

$$r_i = \sum_{n=1}^{N} \frac{|w_{in}|}{\sum_{f=1}^{F} |w_{fn}|} \tag{2}$$

where r_i is the predictive power of the i-th feature, N is the number of hidden layer neurons, F is the numbers of features, w_{in} is the weight connecting the n-th hidden neuron with the i-th feature and w_{fn} $(= 0...F)$ is the f-th weight of the n-th hidden neuron.

The random perturbances of single feature values were not evaluated experimentally, because in the experiments we were removing features (thus setting them to zeros) before network training and that already partially corresponds to the perturbance analysis.

6. Instance Selection Embedded into Network Learning

We use the MLP neural network with hyperbolic tangent transfer function and with the number of output neurons equal to the number of classes. Most of the existing neural network training algorithms can be used. The error for a single vector is given by the following formula:

$$Error(x_v) = \sum_{c=1}^{C} (y_{ac} - y_{ec})^2 \tag{3}$$

where C is the number of classes, y_{ac} is the actual value of $c-th$ output neuron signal and y_{ec} is the expected value of $c-th$ output neuron signal (which is 1 if the

current instance class is represented by the $i-th$ output neuron and -1 otherwise). We assume that a vector is classified correctly if the neuron associated with its class gives a higher signal than any other output neuron.

If an instance is classified incorrectly, the error the network gives as a response to that instance is high. If an instance is classified correctly and is situated far from a class boundary, the error obtained for that instance is very low (due to the hyperbolic tangent transfer function shape). Thus we can remove the instances with the highest error (greater than $maxError$), as they are outliers and these with the lowest errors (lower that $minError$), as they do not help to determine the proper decision boundary [20]. By adjusting the two parameters $maxError$ and $minError$ we can regulate the compression level. That however cannot be done from the very beginning of the network training, because at that stage the network produces more or less random errors for each instance, as the learning starts with random weights. As the training progresses, we can gradually decrease $maxError$, starting from its maximal possible value ($4C$, assuming MSE error measure). $minError$ does not require gradual increasing but can be set at the end of network learning. $maxError$ improves mostly accuracy and compression very little [20]. We set $maxError = 1.6 + 0.16 \cdot numberOfClasses$. The influence of $minError$ values on the data compression and accuracy together with feature selection is experimentally evaluated in section 8.

7. Joined Feature and Instance Selection Embedded into Network Learning

To determine the optimal order of joined feature and instance selection with neural networks, we conducted experiments trying feature selection first, instance selection first and simultaneous feature and instance selection. As in the case of the selection prior to network learning, the results confirmed that the best option is to perform feature selection first and then instance selection. Thus the network training consists of three parts: 1. standard network training, 2. removal of irrelevant features, 3. removal of irrelevant instances. After the second step the training can either continue or it can be restarted from random weights. Better results were obtained with the restart.

We observed in the experiments that instance selection embedded in neural network worked well, but feature selection was in some cases as good as done with feature filters and in some cases less effective. For that reason we added another option for feature selection. First we build a simple neural network with three hidden units and train this network separately on each feature. We use early stopping, so that we can measure the classification accuracy on a training set, without the need of crossvalidation. Then we sort the features by the classification accuracy. Then we add the features to the reduced dataset starting from the most informative one. However, before we add the next one, we calculate its correlation with all the already added feature. If the correlation with at least one of them is higher than the threshold, we reject this feature. This appeared to be the most accurate option.

8. Experimental Evaluation

The purpose of the experiments was to evaluate particular methods in terms of classification accuracy and data compression and determine the Pareto-line for each method. The Pareto-line is a line in the compression-accuracy coordinates, shown in Fig. 1 that connects the points with the best compression for a given accuracy and best accuracy for a given compression, so that no other points exist that can improve both. The compression is defined as percentage of remaining features multiplied by percentage of remaining instances (lower value is better).

In the experiments we used RapidMiner for feature selection and some of the instance selection before the network training. All the other experiments, especially data selection embedded into neural network were performed in our own program. All the software we used can be downloaded from our web page at *www.kordos.com/tfml2017*.

We trained the networks with the R-prop algorithm. We used networks with one hidden layer. The numbers of neurons in the hidden layer was equal to the geometric mean of the number of inputs and number of classes. We performed the experiments on 10 classification datasets from the Keel Repository [21]: Ionosphere (351,33,2), Image Segmentation (210, 18, 7), Magic (19020, 20, 2), Thyroid (7200, 21, 3), Page-blocks (5472, 10, 5), Shuttle (57999, 9, 7), Sonar (208, 60, 2), Satellite Image (6435, 36, 6), Penbased (10992, 16, 10), Ring (7400, 20, 2). The numbers in the brackets are: number of instances, number of features, number of classes. All the experiments were performed in 10-fold crossvalidation and repeated 10 times. The purpose of repeating the experiments 10 times was to average over the initial random network weights and thus to ensure more stable and reliable results.

Table 3. Average values over the 10 datasets of classification accuracy of neural networks (F100-A, F60-A, F30-A) and number of selected instances (F100-I, F60-I, F30-I) for respectively 100%, 60% and 30% of features with three data selection methods (the real numbers of features were the nearest integers to these percentages).

Method	IS	F100-A	F100-I	F60-A	F60-I	F30-A	F30-I
FS: Inf. Gain	no selection	92.74	100	92.12	100	91.02	100
IS: ENN+CNN	m=8, k=9	93.01	35.22	92.30	29.11	91.03	27.81
with variable	m=7, k=9	92.65	18.91	91.58	16.15	88.76	15.11
m in k-NN	m=5, k=9	87.44	7.11	87.11	5.89	87.72	6.41
IS: ENN+IB3	ENN+IB3	87.15	3.85	86.88	3.98	87.01	4.95
FS in separate	no selection	92.74	100	92.90	100	91.45	100
network	minE=0.03	93.02	38.45	92.30	32.98	91.23	36.40
IS embedded	minE=0.1	92.64	19.98	91.91	22.12	88.76	26.78
into NN	minE=0.3	89.05	7.11	88.23	5.89	88.25	10.23
FS embedded	no selection	92.74	100	92.12	100	88.34	100
into NN	minE=0.03	93.02	8.45	92.15	33.15	88.94	38.14
IS embedded	minE=0.1	92.61	19.98	91.05	24.98	87.02	14.18
into NN	minE=0.3	89.05	7.11	87.91	9.15	86.15	10.78

The standard deviations were between 0.5 for the largest to 3.0 for the smallest datasets. The T-test confirmed that the differences in Table 3. are statistically

significant. (For example for the accuracies of 90.00 and 91.00 with 100 cases, the p-value of 0.05 is obtained with standard deviation of 3.585.)

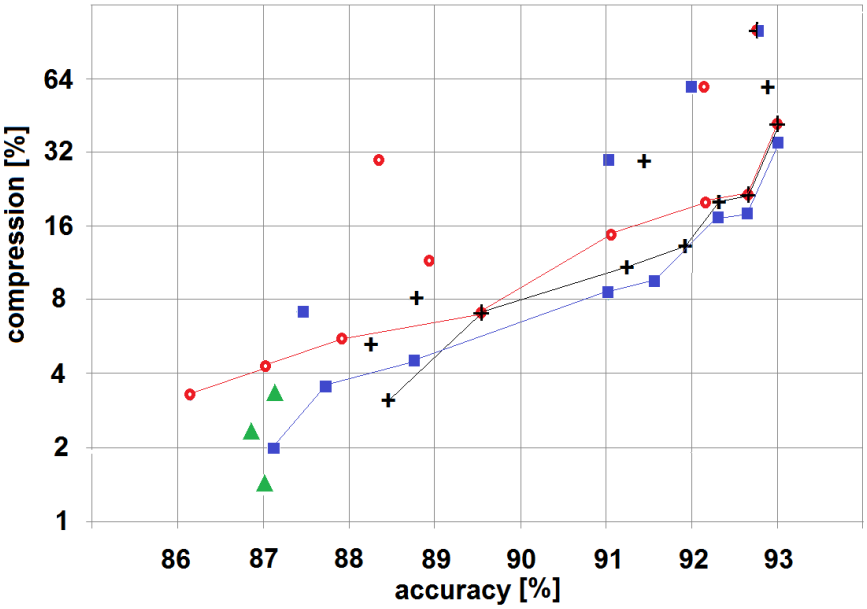


Figure 1. Compression (percentage of remaining features times percentage of remaining instances) vs classification accuracy. The compression axis is in logarithmic scale. A Pareto line is shown separately for each method (square = FS: Inf. Gain, IS: Mod. ENN+CNN, cross = FS in sep. network, IS embedded, circle = FS and IS embedded, triangle: FS: Inf. Gain, IS: ENN+IB3).

9. Conclusions

For the biggest data sets we were able to remove about 98-99% of instances without noticeably accuracy loss, but for smaller datasets the reduction was much weaker.

The most effective data selection is performed by feature selection followed by instance selection. This is true as well as for the selection prior to network training as embedded into the neural network. Currently the Pareto line for the selection with information gain and ENN+IB3 and then modified ENN+CNN is situated closest to the right lower corner, so this method looks the better. However, each of the methods have some strengths.

Feature ranking obtained by learning a simple neural network on a single features with removal of highly correlated features worked very well. The standard feature rankings, as information gain, were on the second place, while feature selection by neural network weight analysis on the third place. However, the last methods can be further enhanced by considering the data flow through the entire network, not only

the input to hidden weights and thus may produce better results, which will the topic of a further study.

Embedding noise reduction into the neural network learning process gives usually very good results. That can be attributed to the shape of decision boundaries, where the k-NN algorithm have the tendencies to smooth the edges.

Instance selection as noise removal works quite well in each case. There is however one problem with instance selection as data compression. Both DROP-3 and the instance selection based on the network error overcomes the shortage of CNN that it works in a random order, as they both preserve more of the instances situated close to class boundaries. However, both of the methods rely on the distance to the opposite class measured either directly (CNN, IB3 and the DROP family) or by the instance location reflected by error produced by the hyperbolic tangent function transformation. But both of the approached do not consider the fact, that the distance between opposite class instances maybe different in different areas of the input spaces and thus sometimes tend to remove rather the instances closest to the boundary, even if they are hidden between the "first row" of instances then the instances that are further, but in the first row and thus needed to preserve the boundary. That is considered by other instance selection methods, which examine the classes of neighbor instances of Voronoi cells, but in spite of that they do net perform better. Finding an effective solution to this problem is still open.

It is likely that the results can be further improved if the variable m in the k-NN algorithms is used also with IB3 and DROP-3 algorithms, which have better compression than CNN with comparable accuracy. That will be another topic of our future research.

10. References

[1] Kordos, M., Blachnik, M., Bialka, S., *Instance selection in logical rule extraction for regression problems.* Lecture Notes in Artificial Intelligence, 2013, **7895**, pp. 167–175.

[2] Blachnik, M., Kordos, M., *Simplifying SVM with weighted LVQ algorithm.* Lecture Notes in Computer Science, 2011, **6936**, pp. 212–219.

[3] Liu, H., *Computational Methods of Feature Selection.* Chapman and Hall, 2007.

[4] Stanczyk, U., Jain, L.C., *Feature Selection for Data and Pattern Recognition.* Springer, 2015.

[5] Uribe, C., Isaza, C., *Expert knowledge-guided feature selection for data-based industrial process monitoring.* Rev. Fac. Ing. Univ. Antioquia, 2012, **65**, pp. 112–125.

[6] Kordos, M., Cwiok, A., *A new approach to neural network based stock trading strategy.* Lecture Notes in Computer Science, 2011, **6936**, pp. 429–436.

[7] Hofmann, M., Klinkenberg, R., *RapidMiner: Data Mining Use Cases and Business Analytics Applications.* Chapman and Hall/CRC, 2016.

[8] Sun, X., Chan, P.K., *An analysis of instance selection for neural networks to improve training speed.* International Conference on Machine Learning and Applications, 2014, pp. 288–293.

[9] Garcia, S., Derrac, J., Cano, J.R., Herrera, F., *Prototype selection for nearest neighbor classification: Taxonomy and empirical study.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 2012, **34**, pp. 417–435.

[10] Olvera-Lapez, J.A., Carrasco-Ochoa, J.A., Martin, J.F., Kittler, J., *A review of instance selection methods.* Artificial Intelligence Review, 2010, **34**, pp. 133–143.

[11] Grochowski, M., Jankowski, N., *Comparison of instance selection algorithms.* Lecture Notes in Computer Science, 2004, **3070**, pp. 580–585.

[12] Antonelli, M., Ducange, P., Marcelloni, F., *Genetic training instance selection in multiobjective evolutionary fuzzy systems: A coevolutionary approach.* IEEE Transactions on Fuzzy Systems, 2012, **20**, pp. 276–290.

[13] Anwar, I.M., Salama, K.M., Abdelbar, A.F., *Instance selection with ant colony optimization.* Procedia Computer Science, 2015, **53**, pp. 248–256.

[14] Derrac, J., Cornelis, C., Garcia, S., Herrera, F., *Enhancing evolutionary instance selection algorithms by means of fuzzy rough set based feature selection.* Information Sciences, 2012, **186**(73–92).

[15] Wilson, D.R., Martinez, T.R., *Reduction techniques for instance-based learning algorithms.* Machine Learning, 2000, **38**, pp. 257–286.

[16] Blachnik, M., Kordos, M., *Bagging of instance selection algorithms.* Lecture Notes in Computer Science, 2014, **8468**, pp. 40–51.

[17] Kordos, M., *Instance selection optimization for neural network training.* Lecture Notes in Artificial Intelligence, 2016, **9692**, pp. 610–620.

[18] Tsaia, C.F., Eberleb, W., Chu, C.Y., *Genetic algorithms in feature and instance selection.* Knowledge-Based Systems, 2013, **39**, pp. 240–247.

[19] Leray, P., Gallinari, P., *Feature selection with neural networks.* Behaviormetrika, 1999, **26**, pp. 145–166.

[20] Rusiecki, A., Kordos, M., Kaminski, T., Gren, K., *Training neural networks on noisy data.* Lecture Notes in Artificial Intelligence, 2014, **8467**, pp. 131–142.

[21] Alcala-Fdez, J., et al., *Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework.* http://sci2s.ugr.es/keel/datasets.php, Journal of Multiple-Valued Logic and Soft Computing, 2011, **17**, pp. 255–287.

Schedae Informaticae Vol. 25 (2016): 165–176
doi: 10.4467/20838476SI.16.013.6194

tfml 2017
theoretical foundations
of machine learning, Kraków

Online Supervised Learning Approach for Machine Scheduling

Bartosz Sądel, Bartłomiej Śnieżyński
AGH University of Science and Technology
Faculty of Computer Science, Electronics and Telecommunication
Department of Computer Science
al. Mickiewicza 30, 30-059 Kraków
e-mail: *sadel@agh.edu.pl, bartlomiej.sniezynski@agh.edu.pl*

Abstract. Due to rapid growth of computational power and demand for faster and more optimal solution in today's manufacturing, machine learning has lately caught a lot of attention. Thanks to it's ability to adapt to changing conditions in dynamic environments it is perfect choice for processes where rules cannot be explicitly given. In this paper proposes on-line supervised learning approach for optimal scheduling in manufacturing. Although supervised learning is generally not recommended for dynamic problems we try to defeat this conviction and prove it's viable option for this class of problems. Implemented in multi-agent system algorithm is tested against multi-stage, multi-product flow-shop problem. More specifically we start from defining considered problem. Next we move to presentation of proposed solution. Later on we show results from conducted experiments and compare our approach to centralized reinforcement learning to measure algorithm performance.

Keywords: supervised learning, reinforcement learning, scheduling, multi-agent system

1. Introduction

Machine scheduling problem is almost as old as the first Ford's assembly lines, dating back to twenties of 19th century. Rapid development of manufactures sparked

Received: 11 December 2016 / Accepted: 30 December 2016

efforts to make this process more efficient. Early on heuristic methods have caught the attention. Those included Just in Time(JIT), Kanban and few others [1]. Those methodologies emerged from car manufactures in Japan and soon become popular in other industries. IT business had initially adapted those concepts to team management process, to become scheduling problem solutions later on. Simultaneously pull and push systems where introduced in [2].

Machine scheduling may be considered as an optimization problem. Unfortunately, due to being NP-hard, finding optimal solution is not an easy task. Therefore, many algorithms were developed to solve it with increasing effectivity. Along finding new methods, problem also evolved to become its on-line version in which the goal is to optimize scheduling as continuous process with orders/tasks being generated when system is already working. In the past few years, agent-based solutions with use of reinforcement learning have been proposed for such cases [3, 4].

In this paper we consider agent-base approach with another machine learning strategy, that is supervised learning. Results in other domains show that in complex environment this type of learning gives improvements faster than reinforcement learning [5, 6].

In this paper, we will begin with defining on-line machine scheduling problem. Next, we will propose our solution and provide necessary theory. This will be followed by conducted experiments and conclusions. Our aim is to check whether supervised learning can replace reinforcement in dynamic environments.

2. Problem Representation

In this paper dynamic version of *flow-shop* problem is considered. It should reflect real manufacturing process where we have company producing products of n different types. Each product requires processing on m different stations. Single station may contain any number of machines. It's enough for a product to be processed just by one machine on every station. To make similarity with traditional *job-shop*, we can name process of producing product a *job* and processing a product by a certain machine a *task*. Later on we will use these terms interchangeably. Each product needs to pass all the stations. Additionally path alongside stations have to be the same for all product types. Thanks to these restrictions we can number stations by the sequence in which jobs will be passing through those. Since in real life manufacturing situation usually there is a delay between finishing processing of a product on one machine and starting on the next one, we have to introduce buffers between stations, where we can store the waiting products. As orders usually consist of more than just one unit of product, we need additional buffer for finished jobs where products are stored until whole order is completed. Sample configuration is presented in figure 1. To simplify demonstration without loss of generosity we assumed different types of products to be marked as circles, pentagons and octagons. Every product in presented example must proceed sequentially by *Station1*, *Station2* and *Station3*. Finished products are stored in last buffer, called *Completed*.

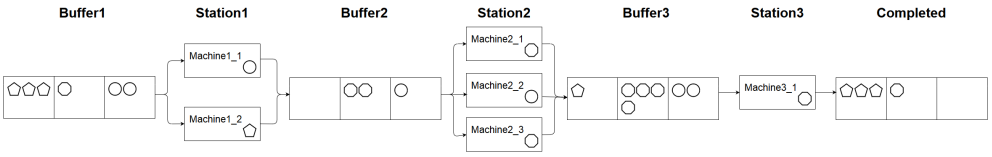


Figure 1. Example of model representing considered problem

Every order o_i in our model can be defined as:

$$o_i = (p_i, c_i, d_i, r_i, f_i, ty_i, q_i) \tag{1}$$

where p_i stands for priority, c_i for creation time, d_i for due time, r_i for reward received after order completion, f_i for penalty received if order is not completed on time, ty_i for the type of the product order demands and q_i for quantity of this product. λ_{ty} is Poisson parameter describing the frequency with which orders of type ty are arriving. Number of product units q_i is drawn from uniform distribution $U(a, b)$ where a and b are minimum and maximum values. After arriving, orders are stored in the queue sorted by their priority, so the most important ones are delivered first. When new order comes to the system, products of which it is composed are inserted into $Buffer_1$. Then machines from $Station_1$ takes products from it and work on them. After processing is finished products are stored in $Buffer_2$. Next the same happens for the next stations until products finally gets to $Completed$ buffer. When there will be enough products in buffer, we are delivering first order from our queue. Depending if we managed to deliver order before due time it is possible to get reward or penalty. Machine is defined as:

$$M_{i,j} = (H_{i,j}, V_{i,j}, t_{i,j}, ty_{i,j}) \tag{2}$$

where i is number of station, j is number of machine in station, $H_{i,j}$ is the health of machine, $V_{i,j}$ is the table containing velocity of processing of different product types on this machine, $t_{i,j}$ is the time is takes to reconfigure machine and $ty_{i,j}$ is type of product machine is configured to process. Machines are allowed to change the type of product they are processing only when they are idle. Each time, machine have ty different actions to choose, where each one corresponds to processing different product type. When action of changing processed type is chosen, machine needs to remain idle for a fixed period of time which simulates machine reconfiguration. In our model every machine can process just one product at a time. Thanks to this property of our model, sometimes it is more beneficial for a machine to wait for the product of the currently configured type, instead of changing. Additionally each machine can process each product type with different speed. This property makes certain machines better at processing one type of a product, where another one may work better with other type. Although we allow for situations where single machine or all machines in station needs no time for processing certain type of products, we still demand this product to pass through that station. In our environment every machine has chance to break down. When it happens currently processed product is pushed back to the previous buffer. Machine remains than idle for a $t_{i,j}$ of turns, simulating fixing process, after which it starts working again.

Important thing in every model is optimization criteria which is used to describe system effectiveness. In problem we are considering there are many different parameters we may want to optimize. The most common would be profit maximization or minimization of product amounts in buffers. First option is reflecting most business scenarios. This way we don't only look for most valuable orders to complete, but we also have to the handle the ones with the highest penalty. The second option chooses what to do basing on number of products waiting in all buffers excluding the *Completed* one. This way we are minimizing idle time of machines.

Table 1. Notation of parameters

T	Number of product types
ty	Product type
λ_{ty}	Poisson parameter describing frequency of orders with product type ty
$M_{i,j}$	Machine j in station i
$H^t_{i,j}$	Health of machine j in station i in turn t
$V_{i,j}$	Table with velocities of processing different product types on machine j in station i
$t_{i,j}$	Time it takes to reconfigure machine j in station i in turn t
o_i	I-th order
p_i	Order priority
r_i	Order reward
f_i	Order penalty
c_i	Order creation time
d_i	Order due time
q_i	Quantity of product units in order
$Buffer^t_{i,ty}$	Quantity of products ty in buffer of ı-th station in turn t
$Station_i$	I-th station
P_{limit}	Threshold used in supervised learning
u_{ty}	Unit price of product type ty
s_{ty}	Switch cost of product type ty

In this paper we chose to use the last option as it seems interesting for manufactures which don't want machines to be idle. Example of such machine may be foundry furnace which is shut down only once every twenty years and we would like to use it as much as we can since it can't be turned off. Used notation by us is gathered and presented in table 1.

3. Proposed Solution

Our solution takes usage of approach proposed by Śnieżyński in [6]. Namely, we will try to employ supervised learning algorithms to solve problem previously defined in section 2. It's done by introducing learning agents with special architecture allowing them for taking usage of machine learning algorithms, designed for static *problems*.

First we need to model our problem as a multi-agent system. To do so we will treat every machine like a separate intelligent agent. Each of them will have it's own knowledge base and own classifier. Decisions about changing processed product type by machine $M_{i,j}$ will depend only on the decision of the agent connected with this machine itself.

Algorithm implemented by our system is presented on figure 3. In this diagram we can see flow of simulation in implemented solution. The first step in every turn is order generation. Whether order should be generated this turn and if so, how big should it be depends on λ_{ty} and parameters of experiment establishing lower and upper boundary of order sizes. Next, products form generated orders are added to the buffer of first station. After that for each machine in station we check if it has finished processing of any product in previous turn. If so, we collect it so later we can add it to the buffer in next station. Next step is to fire learning process on machine's classifier. In our experiments we do this every 5 turns. Later, every machine has a chance to change type of processed product. The change is impossible, if machine is already processing some product. In case of changing product type we save that information in a form of entry in our knowledge base. It's value is calculated using equation 3. In other case we continue to process currently processed product. In this places we are also checking if machine should break in this turn. Next, we go on to the next machine in station. After processing of all machines in a single station we move on to the next station but this time products added to the buffer are ones collected in previous station. When all stations are finished we start the next turn.

Architecture of single agent is presented in figure 2. This model introduces two main modules. First of them, named Processing module, is responsible for perceiving of the environment. It receives informations from environment and other agents. Later on, performs transformation of those observations into format appropriate for the classifier. Afterwards, learning module is asked if it knows what to do in a current situation basing on previously learned knowledge. Learning module communicates with classifier which role is to classify received data into one of the T classes where T is number of different product types considered in problem instance. If this classifier can choose proper action with probability higher than P_{limit} it simply responds with this action. If it's unsure what to do it can trigger learning process using gathered training data. After that learning module uses classifier again and returns the best choice without checking against p_{limit}. Finally, processing module puts chosen action into practice.

Although from outside this may look as reinforcement learning it's quite different. Where in reinforcement learning agents have the knowledge about previous states hidden in trained classifier, here we can collect history of environment states and taken actions (decisions made by agents) in form of training set. This way we can

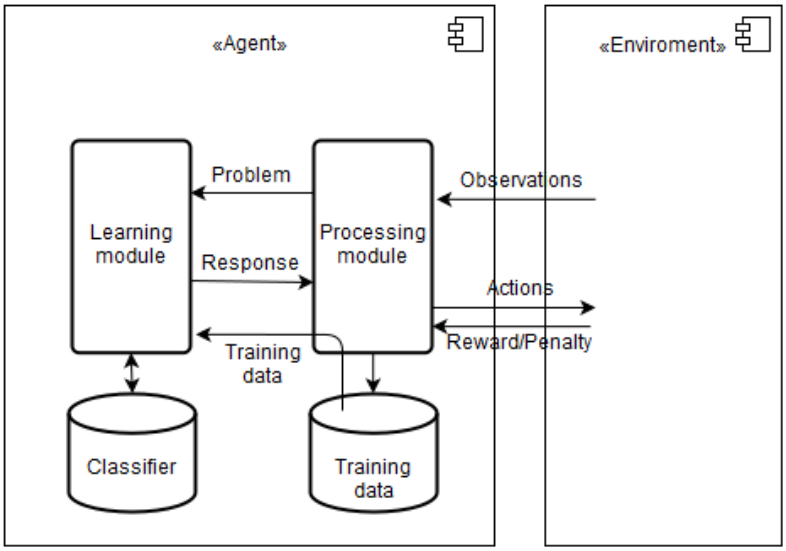


Figure 2. Used agent architecture

explicitly see experiment history and learn from it.

Each entry in the training set consists of two parts. The first of them is data and the second is a label, also called a category, which is description of the data part. There are two common ways to construct entries in problems like one considered here. In first of them data part of each entry consists of buffers from each station and health property of every machine. Product type which should be produced in this state is used as category in this approach. This method is often used as it's simple to implement and directly tells us, what should we do in given state. The second method combines data part and category from previous method to create the data part. As a label in this option we are using quality of the decision. Although harder to implement, this method offers us benefits, in a form of a way to express how good each decision is. Thanks to this, we can connect the same state with two different actions, where in the first method this would be impossible. In our work we are using second approach as we hope that ability to express value of moves, will improve proposed solution.

$$f(ty,i,j,t) = \begin{cases} u_{ty} * sgn(Buffer^t_{i,ty}), & \text{if } ty = ty_{i,j} \\ u_{ty} * sgn(Buffer^t_{i,ty}) - s_{ty}, & \text{otherwise} \end{cases} \tag{3}$$

Each entry is composed of amounts of products in buffers from 1 to m, health of all machines and a chosen action, which takes one of ty values. Value of our entry is calculated using equation 3. Entries are added to training set after every machine decision.

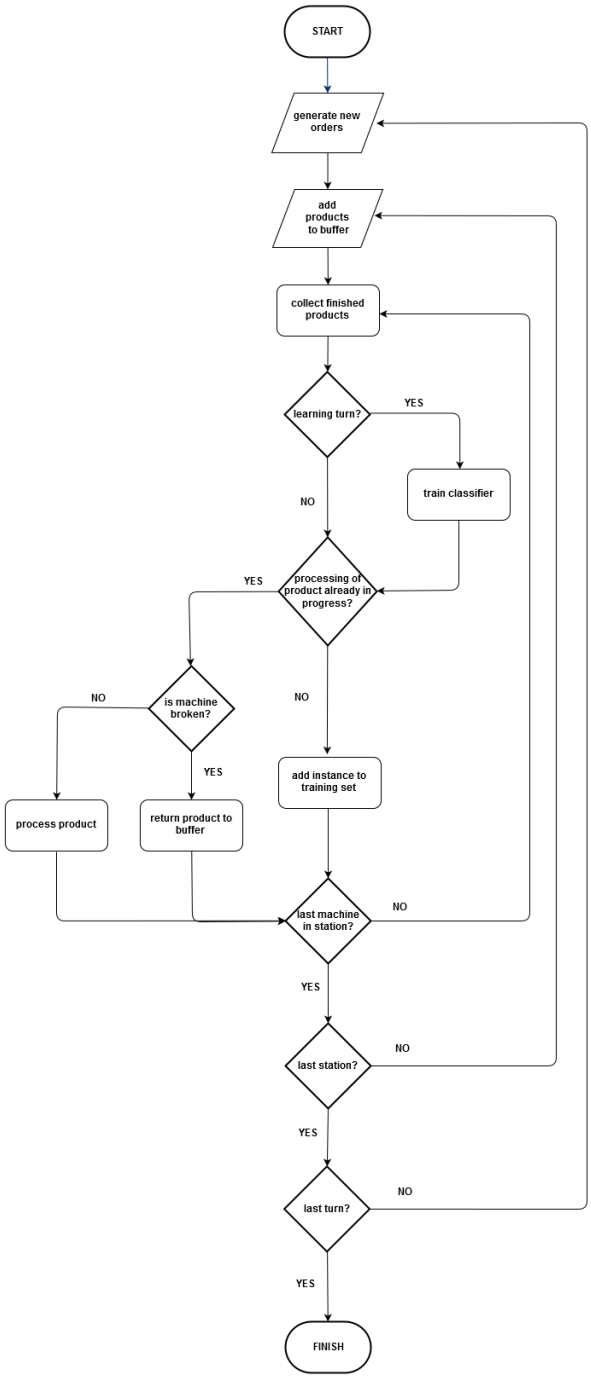


Figure 3. Flow diagram of implemented simulation

Learning process can be started every l turns or when quality of the best action is lower than a certain threshold. Depending on tools used it may mean constructing new classifier based on whole training set or just improve it using previously unseen examples. We have to take in to account that training classifier may be computationally expensive depending on the number of stations and machines in each of them, as they make the amount of collected entries significantly larger. This property makes supervised learning worse than reinforcement learning in problems where products are processed fast and there is no much time for decision-making. However in cases when processing takes hours or even days, learning may be run parallel to machine work thus be imperceptible.

4. Results

In order to test our approach various experiments were conducted. In all of them our method was tested against multi-agent reinforcement learning proposed in [4]. Parameters used in simulations are presented in table 2. Every test was conducted 10 times and the presented results are the mean of received ones. Although system is able to carry out much bigger simulations we chose to present smaller ones. This choice is motivated by similar configurations used in [4], which allows us for a better comparison between both solutions.

4.1. Comparison in a 2x2 System

First test were run in a simple configuration containing only 2 stations with 2 machines each. In this setup algorithms should learn quickly.

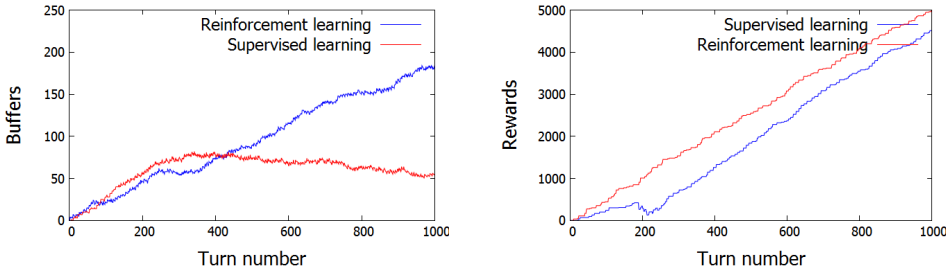


Figure 4. Results for 2x2 configuration

In figure 4a we can see results of both algorithms running in the same problem composition. We can notice that although supervised learning algorithm had slower

Table 2. Experiment parameters

Parameter	Value	
T	2	
ty	0	1
s_{ty}	40	45
u_{ty}	25	30
λ_{ty}	3	3
r_i	Sampled from uniform distribution $U(0, 60)$	
f_i	$d_t - t$	
Machines health	Sampled from uniform distribution $U(0, 1)$ with 0,05 probability to break	
d_i	Sampled from uniform distribution $U(15, 25)$	
Min order size	1	3
Max order size	1	3
Time span between learning	5 turns	
Supervised learning algorithm	J48 (C4.5)	
Reinforcement learning algorithm	Watkins	

start it took the lead before end of experiment. Although rough start it managed to stabilize and even decrease number of products waiting in buffers, where reinforcement learning struggled with that much harder. Slower start of supervised system version was probably caused by lack of the knowledge at the start of the experiment. Reinforcement learning performed better in that situation because it does not need as much previous experience to work with as supervised learning. Situation has changed when the supervised algorithm gathered enough entries in the training set and greatly improved it's performance.

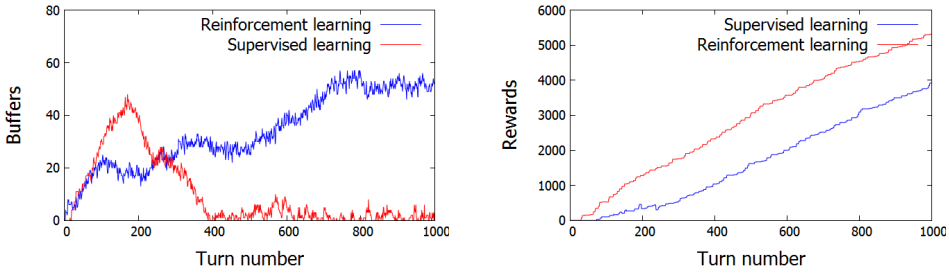
While reward optimization was not a goal of learning, we also made statistics of reward collected by both solutions. Results achieved for 2x2 configuration are presented on figure 4b. Since supervised learning gathers knowledge slower it's also starting to gain profits later. Despite rough start, our approach manages to leverage it's experience and start to achieve profits equal to those gathered by reinforcement learning.

4.2. Comparison in a 3x3x3 System

Second experiment was conducted in model with 3 layers where each of them contained 3 machines. This case allows to test algorithms capabilities in more complex environment, although we have to remember that even the simplest examples from

real manufacturing will probably be much more complicated than this one. Received results are presented in figure 5a. Both methods performed better than in previous example. Reinforcement learning managed to stabilize quickly and keep the number of products in buffers on almost the same level for the whole simulation. Supervised learning like before had problems on the beginning. However, around turn 200 it had break through and managed to decrease the number of products waiting for processing to just a few. With such performance it is maintaining an advantage over reinforcement learning to the end of the simulation.

In figure 5b we can see rewards achieved by learning algorithms during experiment. As we can see even without maximizing received rewards, both algorithms managed to work out profits. Reinforcement learning again achieved better results on this chart. It's due to the fact that it has been evenly working for the whole time. Supervised algorithms despite of having almost empty buffers in second half of the algorithm didn't manage to catch up. Although in later stages of experiment profit gain of both algorithms was equal, penalties for the first few orders put supervised learning too much behind.



(a) (b)

Figure 5. Results for 3x3x3 configuration

5. State of the art

Since machine scheduling problem is one of the most crucial ones for today's manufacturing it has gathered a lot of attention in the recent years. As a result of this, vast diversity of algorithms aiming to optimize the whole process has emerged. Solutions based on various approaches included among others mixed-integer programming presented in [7] to solve flowshop problem using two criterias, makespan and flow-time. Genetic algorithm approach was considered to solve n-jobs, m-machines setup. Results of this try compared with simulated annealing and neightbourhood search were presented in [8].

There are several works considering reinforcement learning with Q-learning in particular as a possible solution to machine scheduling problem. Amongst them usually multi-agent approach, where each machine serves as agent, is proposed as possible

solution. In [9] author employed multi-agent reinforcement learning for job shop problem in a real life enviroment. It brings good results but operates on small action space containig just 3 elements. Another work which employs reinforcement learning is [4] where centralized version of *reinforcement learning* approach is proposed for scheduling in online flow-shop problem. This solution despite taking longer time for single step evaluation, converges faster than multi-agent variation. There are less works considering supervised learning as solution. In [10] framework trying to learn rules corresponding with creation of optimal solution is introduced. In [11] support version machines (SVM) algorithm was used to solve resource constraint scheduling problem which is generalization of flow-shop problem considered by us.

Unfortunately, since labelling cost is high as it needs a lot of time from the experts and not always is even possible to be done, standard approach of *supervised learning* doesn't fit on-line version of scheduling problem too well. In reactive environments where there is no previously defined training set, most of algorithms are unable to learn. Luckily, proposed architecture allows agents to apply supervised learning autonomously and on-line, so they can tackle such problems too.

6. Conclusion

In this paper we have proposed a multi-agent solution for a *dynamic flow-shop* version. Agents apply autonomously supervised learning on line. Every agent collects experience which forms training data used to learn a classifier applied to chose what type of product should be processed on every machine. Later on experiments were conducted in which presented approach was tested against multi-agent reinforcement learning solution. The goal of the agents was to decrease number of products in buffers. As we can conclude after experiments, supervised learning outperformed reinforcement learning in terms of the assumed goal. Although it needed more time to take a grasp and converge with time it's getting better and better. It stabilizes earlier and can even decrease the number of product waiting in buffers where reinforcement learning was unable to do so in tested scenarios. What it means is that supervised learning is viable solution for dynamic problems if the right architecture is used.

Further research can focus on various directions. One direction is to try to employ presented approach using single-agent system instead of multi-agent one used in this thesis. Second way to go is to introduce some form of communication between agents, to enhance received result even further. The last direction is to use this approach in different problems with dynamic nature and see if it's gathering as good results as in this one.

Acknowledgements

The research presented in this paper was supported by the Polish Ministry of Science and Higher Education under AGH University of Science and Technology Grant 11.11.230.124.

7. References

[1] Sendil Kumar, C., Panneerselvam, R., *Literature review of jit-kanban system.* The International Journal of Advanced Manufacturing Technology, 2007, **32**(3), pp. 393–408.

[2] Olhager, J., Östlund, B., *An integrated push-pull manufacturing strategy.* European Journal of Operational Research, 1990, **45**(2), pp. 135–142.

[3] Ouelhadj, D., Petrovic, S., *A survey of dynamic scheduling in manufacturing systems.* Journal of Scheduling, 2008, **12**(4), pp. 417.

[4] Qu, S., Chu, T., Wang, J., Leckie, J.O., Jian, W., A centralized reinforcement learning approach for proactive scheduling in manufacturing. In: *ETFA*, IEEE, 2015, pp. 1–8.

[5] Sniezynski, B., *Agent strategy generation by rule induction.* Computing and Informatics, 2013, **32**(5).

[6] Śnieżyński, B., *A strategy learning model for autonomous agents based on classification.* International Journal of Applied Mathematics and Computer Science, 2015, **35**(3), pp. 471–482.

[7] Selen, W.J., Hott, D.D., *A mixed-integer goal-programming formulation of the standard flow-shop scheduling problem.* Journal of the Operational Research Society, 1986, pp. 1121–1128.

[8] Reeves, C.R., *A genetic algorithm for flowshop sequencing.* Computers & operations research, 1995, **22**(1), pp. 5–13.

[9] Beke, T., Multi-agent reinforcement learning in a flexible job shop environment: the vcst case. Master's thesis, Gent Universiteit, Gent, Belgium 2013.

[10] Ingimundardottir, H., Runarsson, T.P., Supervised learning linear priority dispatch rules for job-shop scheduling. In: *Learning and Intelligent Optimization: 5th International Conference.* Springer 2011 pp. 263–277.

[11] Gersmann, K., Hammer, B., *Improving iterative repair strategies for scheduling with the {SVM}.* Neurocomputing, 2005, **63**, pp. 271 – 292 New Aspects in Neurocomputing: 11th European Symposium on Artificial Neural Networks.

Schedae Informaticae Vol. 25 (2016): 177–188
doi: 10.4467/20838476SI.16.014.6195

tfml 2017
theoretical foundations
of machine learning, Kraków

Portfolio Inputs Selection from Imprecise Training Data

Sarunas RAUDYS[1], Aistis RAUDYS[1], Zidrina PABARSKAITE[1],
Gene BIZIULEVICIENE[1,2]
[1]Department of Mathematics and Informatics, Vilnius University
Didlaukio 47, 08303, Vilnius, Lithuania
e-mail: *sarunas.raudys@mif.vu.lt*
[2]State Research Institute Centre for Innovative Medicine
Zygimantu 9, 01102, Vilnius, Lithuania

Abstract. This paper explores very acute problem of portfolio secondary overfitting. We examined the financial portfolio inputs random selection optimization model and derived the equation to calculate the mean Sharpe ratio in dependence of the number of portfolio inputs, the sample size L used to estimate Sharpe ratios of each particular subset of inputs and the number of times the portfolio inputs were generated randomly. It was demonstrated that with the increase in portfolio complexity, and complexity of optimization procedure we can observe the over-fitting phenomena. Theoretically based conclusions were confirmed by experiments with artificial and real world 60,000-dimensional 12 years financial data.

Keywords: Complexity, financial portfolio, overfitting, sample size, variable selection

1. Introduction.

Data mining methods are slowly making their way into finance, where trading is mainly dominated by econometric and statistical models. The same is with financial portfolio construction. Markowitz mean variance portfolio optimization was proposed

Received: 11 December 2016 / Accepted: 30 December 2016

many years ago and many practitioners still use it in original form to create (learn from data) optimal portfolios [1, 2]. Mean variance portfolio optimization (MVO) is typically used to construct portfolios from various investments. For example, how much stocks bonds and real estate one needs to have in its portfolio to achieve best risk reward ratio? The quality of the portfolio is typically measured by the *Sharpe ratio* (Sh) [2]. This ratio is a mean of the profits divided by the variance (standard deviation). It constitutes how much you earned and with what risk. Very few investigate nonlinear methods offered by machine learning community.

The MVO works with any time series of profit and loses (PNL). So people use it with artificial investments such as generated by automated trading systems. Automated trading is known as algorithmic trading, systematic trading and other names. It is the process where human puts his investments knowledge into the computer program and allows the program to make buy and sell decisions automatically. It varies by types, trading frequency and strategies that are used. The trading firms can employ many potential trading systems. Each trading strategy (TS) can be run in simulated mode and out of this simulation is the series of PNLs. These series correspond to the success of the trading systems to generate profits. The question is what algorithms to trade together to maximize profitability and minimize the risk. Such time series can be used by the MVO engine to calculate the best portfolio best set of trading strategies. Numerous factors influence this process.

Complex portfolio design rules having too large number of inputs for relative short learning sequences often lead to overfitting. It is very easy to get good results in simulations with training data, but notoriously difficult on the unseen data. Main factors that are affecting the overfitting are: the training set size, a number of portfolio inputs, inexact estimation of means values and correlations of the returns [3]. In view of that, it is also very important to verify strategies in out of sample manner and select such methods that will work well on the unseen data.

In present-day tasks we face extremely large number (say, $N = 60{,}000$) of trading strategies and need to construct an investment portfolio for trading during short future time interval. Obviously, we cannot include all N systems into the portfolio. Therefore, we are obliged to choose much smaller subset of the best systems and use simpler portfolio design methods [3]. *The simplest portfolio* is an equal weighted rule where one weights all selected investments equally. This non-trainable portfolio is called, $1/N$, or Naïve portfolio. Often it outperforms more sophisticated methods [4, 5].

Machine learning has numerous methods that allow to deal with imprecise data and to perform feature selection or extraction. Many methods select the "best" subset of N_b ($N_b \ll N$) trading strategies (TS) are suggested in the literature [6, 7].

The simplest way to generate N_b - dimensional subset is to sort N trading strategies according to *sample estimates* of the Sharpe ratio, $\widehat{Sh}$. In this approach, one selects N_b of them having the highest $\widehat{Sh}$ values (method **A**). Sadly, this method ignores correlations. Sophisticated way of the best subset selection is forward selection Comgen procedure [8] that takes into account the correlations (method **B**). It makes series of locally optimal solutions and hopes that it will lead to a near global optimum solution. The benefit of this system is in its simplicity and granularity, as it virtually creates integer portfolio weights $(0, 1, 2, \dots)$ that can be traded straight away.

An important alternative is *a random selection* (method **C**) where one generates

m independent random subsets composed of N_b TSs. One estimates Sharpe ratios of m subsets, and selects the best subset having the highest $\widehat{Sh}$ value. An advantage of this method is *a possibility to analyze the accuracy of the best selection procedure theoretically*. Moreover, this way allows taking into account the correlations and frequently leads to selection of good subset.

Analysis of diverse portfolio design schemas suggested by multiple authors showed that majority of them does not hold out-of-sample scheme [9]. To obtain reliable results, in selection of the best input subset or the portfolio design strategy one needs to use independent validation data set. Due to finite size, the validation data is also imprecise. Hence, *selection of the TSs subset is inexact*. In such circumstances, use of more complex selection algorithm or an increase in the size of the TSs subset, N_b, often does not lead towards the desirable result. It is worth noting that typical MVO assumes that correlation, variance and profitability of the time series will remain constant. Notoriously it changes and changes a lot. Therefore, in the portfolio input and design scheme selection we face notable adaptation to validation set (secondary overfitting).

To our knowledge the *secondary overfitting effect* was never considered in the portfolio design literature. An objective of the present paper is analytic, numerical and experimental clarification of reasons of this important for the practitioner phenomenon and choosing for research directions allowing to overcome this difficulty.

2. Theoretical analysis of accuracy.

An objective of the present section is to obtain an analytical formula to calculate a mean value of true Sharpe ratio when random selection procedure $\boldsymbol{C}$ is used to learn (find) the "best subset" of TSs. To examine the accuracy, one needs to define a distribution density function of true Sharpe ratio values $f_t(Sh)$, and conditional density of estimates, $f_c\left(\widehat{Sh} \mid Sh\right)$. To simplify theory and numerical analysis, we assume true values, Sh, and *estimates*, $\widehat{Sh}$, can take only discrete values, $Sh_1, Sh_2, \ldots, Sh_A$, and $\widehat{Sh}_1, \widehat{Sh}_2, \ldots, \widehat{Sh}_B$. If numbers A and B are sufficiently large, this simplification is not restrictive. Let the elements of both vectors are ranked in an increasing way. In the discrete model, instead of probability densities $f_t\,(Sh)$, and $f_c\left(\widehat{Sh} \mid Sh\right)$, we deal with probabilities of discrete values

$$P_{true}\,(Sh = Sh_i) = P_{true\,i}, \quad (i = 1, 2, \ldots, A)\,, \tag{1}$$

$$P_{cond}\left(\widehat{Sh} = \widehat{Sh}^{\,j} \mid Sh = Sh_i\right) = P_i^{cj}, \quad (i = 1, 2, \ldots, A; j = 1, 2, \ldots, B)\,, \tag{2}$$

where P stands for the probability.

To investigate relations between the sample size and accuracy analytically, we need to choose models $P_{true}(Sh = Sh_i)$ and $P_{cond}\left(\widehat{Sh} = \widehat{Sh}^j \mid Sh = Sh_i\right)$. To calculate the mean Sharpe ratio, $E\,(Sh)$, we need to derive two expressions:

1. conditional probabilities $P_{cond}\left(\widehat{Sh} = \widehat{Sh}^j \mid Sh = Sh_i\right)$ and
2. probabilities of the maximal values of m estimates $\widehat{Sh}_1, \widehat{Sh}_2, \ldots, \widehat{Sh}_m$ (here subscript indicates a serial number the of N_b - dimensional subset of TSs'). Note, each estimate, $\widehat{Sh}_l$, can get any of B values defined in Eq. 2.

Without loosing generality, we normalize values of Sh and $\widehat{Sh}$ to have them varying in interval $[0, 1]$. According to the theory of probabilities, a joint probability of two dimensional vector (Sh, $\widehat{Sh}$)

$$P_{joint}\left(\widehat{Sh} = \widehat{Sh}^j,\, Sh = Sh_i\right) = P_i^{cj} \times P_{true\,i} = P_{ij}^{joint}. \tag{3}$$

Then *unconditional* probability

$$\begin{aligned} P_{ucond}(\widehat{Sh} = \widehat{Sh}^j) = \sum_{i=1}^{A} (P_i^{cj}\left(\widehat{Sh} = \widehat{Sh}^j,\, Sh = Sh_i\right) \times P_{true}\,(Sh = Sh_i) = \\ \sum_{i=1}^{A}\left(P_{ij}^{joint} \times P_{true\,i}\right) = P_{uc}^j. \end{aligned} \tag{4}$$

Subsequently, *conditional* probabilities can be expressed as

$$P_{cond}\left(Sh_l = Sh_i \mid \widehat{Sh} = \widehat{Sh}_l^j\right) = \frac{P_{joint}\left(\widehat{Sh}_l = \widehat{Sh}^j,\, Sh_l = Sh_i\right)}{P_{ucond}(\widehat{Sh}_l = \widehat{Sh}^j)} = P_{ci}^j, \tag{5}$$

where the subscript index l means "any of $1, 2, \ldots, m$".
According to definition of the maximal values their probabilities can be expressed as

$$\begin{aligned} P_{cond}\left(\widehat{Sh}_{maximal} = \widehat{Sh}^j\right) = & P\left(\widehat{Sh}_1 < \widehat{Sh}^{j+1},\, \widehat{Sh}_2 < \widehat{Sh}^{j+1}, \ldots, \widehat{Sh}_m < \widehat{Sh}^{j+1}\right) - \\ & P\left(\widehat{Sh}_1 < \widehat{Sh}^j,\, \widehat{Sh}_2 < \widehat{Sh}^j, \ldots, \widehat{Sh}_m < \widehat{Sh}^j\right). \end{aligned}$$

In the random search selection, the probabilities $\widehat{Sh}_1, \widehat{Sh}_2, \ldots, \widehat{Sh}_m$ are independent. Thus,

$$P\left(\widehat{Sh}_{maximal} = \widehat{Sh}^j\right) = P\left(\widehat{Sh}_l < \widehat{Sh}^{j+1}\right)^m - P\left(\widehat{Sh}_l < \widehat{Sh}^j\right)^m \tag{6}$$

where $P\left(\widehat{Sh}_l < \widehat{Sh}^j\right) = \sum_{l=1}^{j-1} P_{uc}^l$.

As a result

$$P\left(\widehat{Sh}_{maximal} = \widehat{Sh}^{j}\right) = \left(\sum_{l=1}^{j} P_{uc}^{l}\right)^{m} - \left(\sum_{l=1}^{j-1} P_{uc}^{l}\right)^{m} \tag{7}$$

Then the mean value of the expected Sharpe ratio after the random selection procedure, $E(Sh)$, can be calculated from Equations (5) and (7)

$$E(Sh) = \sum_{i=1}^{A} Sh_i \times \sum_{j=1}^{B} P_{cond}\left(Sh_l = Sh_i | \widehat{Sh}_l = \widehat{Sh}^{j}\right) \times P\left(\widehat{Sh}_{max} = \widehat{Sh}^{j}\right) = \sum_{i=1}^{A} Sh_i \times \sum_{j=1}^{B} P_{ci}^{j} \times \left(\left(\sum_{l=1}^{j} P_{uc}^{j}\right)^{m} - \left(\sum_{l=1}^{j-1} P_{uc}^{j}\right)^{m}\right) \tag{8}$$

3. Numerical analysis of the two-dimensional Beta distribution model

Eq. (8) does not allow seeing a relationship between the decreasing of the Sharpe ratio due inexact selection of the best subset of trading strategies in an explicit way. It can be done numerically. To see the relationship of $E(Sh)$ and *validation set size, L, the portfolio inputs selection algorithms complexity parameters*, m, N, N_b, and sets $P_{true\,i}, P_i^{cj} (i = 1, 2, \ldots, A; j = 1, 2, \ldots, B)$ one needs to define them. A simple way to fulfill this requirement is to assume values $P_{true\,i}, (i = 1, 2, \ldots, A)$ to be calculated from Beta density

$$P_{true\,i}, = \gamma Sh^{\alpha}(1 - Sh)^{\beta} \tag{9}$$

where α and β are shape parameters and coefficient γ is chosen from requirement $\sum_{i=1}^{A} Sh_i = 1$. By simple scaling of two extra parameters the Sharpe ratio value can be made to vary in an arbitrary interval. Then we would have a generalized Beta distribution.

We assume conditional probabilities $P_{cond}\left(\widehat{Sh} = \widehat{Sh}^{j} \mid Sh = Sh_i\right)$ are defined by Beta distribution with parameters γ_c, α_c and β_c. In numerical analysis we define values of α_c and β_c according to $mean = Sh_i$ and $variance = V_0/L$ of the Beta distribution density (9)

$$\alpha_c = Sh_i(Sh_i(1 - Sh_i)L/V_0 - 1), \beta_c = \alpha_c(Sh_i^{-1} - 1) \tag{10}$$

where parameter L symbolizes the validation set size used to obtain estimates $\widehat{Sh}^{l}$ and parameter V_0 symbolizes the variance when $L = 1$;

Below we will examine an example with $\alpha = 18.218$, $\beta = 160.968$, $V_0 = 0.006$, $A = B = 1,000$ and $L = 42$ (number 42 symbolizes two months validation days used to estimate Sharpe ratio in automated financial trading). These values were chosen

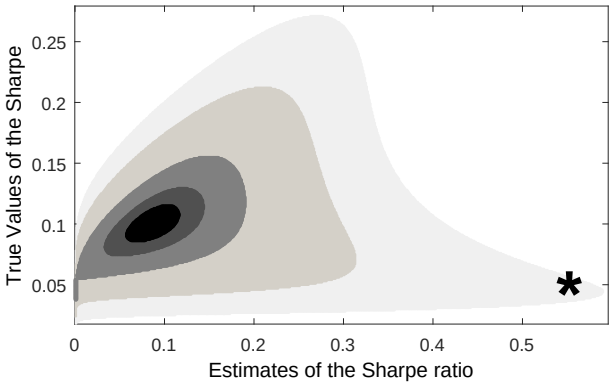


Figure 1. Schema of Two-D Beta distribution density.

while analyzing real world financial data with 50,356 trading strategies. In Figure 1 we present 2-D visualization of distribution of probability $P_{joint}(\widehat{Sh} = \widehat{Sh}^j, Sh = Sh_i)$ in variables $\widehat{Sh}$ (x axis) and Sh (y axis) space. We see the smallest Sharpe values (painted in black, here we have the highest P_{ij}^{joint} values) are much more correlated as the largest $\widehat{Sh}$, Sh values (painted in bright gray, here we have the smallest P_{ij}^{joint} values). Inspection of Figure 1 shows that an increase in the number of m random subsets TSs (entering the rights part of the gray area marked by "*") allows finding subsets characterized by high validation set based Sharpe values. The test estimates (Sh), however, are low in this area.

Calculations according to Eq. 8 confirm the conclusion made from visual analysis of Figure 1. Graphs for $L = 21$, 42 and 63 presented in Figure 2 indicate: 1) with lager value of validation set size, L, we obtain higher true Sharpe values, Sh, 2) the means of Sh are increasing with m at the very beginning, then saturate, and later start decreasing. Thus, training performed by random selection procedure confirms overtraining (peaking) effect known in the data mining and machine learning research [10]. The peaking effect appears earlier when validation set size, L, is small (inspect a curve marked by 21 in Figure 1). When L is extremely small, then $\widehat{Sh}$ and Sh become almost uncorrelated. Then peaking starts almost immediately. Contrary, for larger L values the peaking occurs later and is less expressed (see curve for $L = 64$ in Figure 2).

Note, *parameter m characterizes a complexity of the model selection procedure.* Thus, speaking in general, the theory and graphs in Figure 2 explain peaking relationship between accuracy and complexity of learning portfolio inputs selection algorithm.

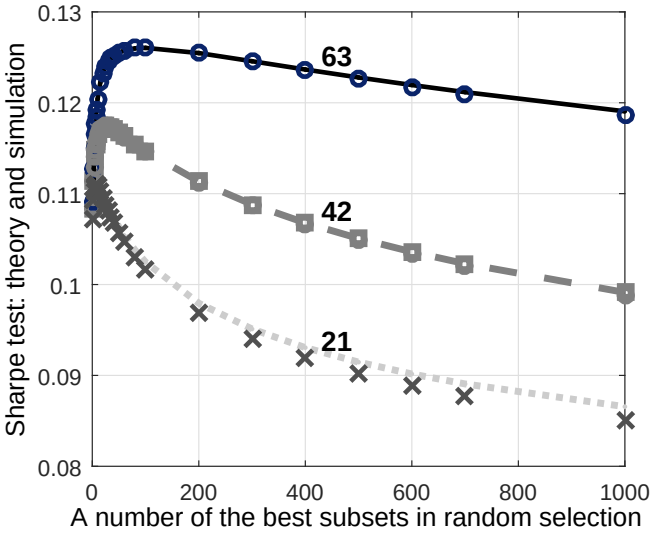


Figure 2. Mean of true Sharpe ratio after the best subset selection.

4. Experiments with 60,314-dimensional financial data

Our analysis is aimed to understand the influence of complexity on the $1/N$ portfolio design in real world financial trading when portfolio dimensionality is extremely high and the size of the data to be used for training (the best TSs subset selection) is relatively small. Therefore, for verification of conclusions presented above we performed experiments with the real world automated trading data.

Trading algorithms comes in many varieties. In our particular case, from prop trading firm we received series (60,314 – dimensions, years 2004 – 2016 data) of the PNLs generated by mean reversion type of strategies. Mean reversion strategies (MRS) are also known as contra trend strategies as they go against the trend. If market is moving upwards at some point MRS can decide that it moved too much and there will be a market correction. So algorithm will short sell and wait for correction. The same is for opposite direction. If the market falls down too much and/or too fast, then the MRS will buy with anticipation of some correction - at least short market movement upwards. If market moves upwards the trading strategy typically will sell and close the position with the profit. Sometimes, especially during market tumor and panic such strategies can generate sharp losses as market moves in one direction for extended period of time. Therefore, it is extremely important such strategies to trade in the portfolio. The risk in portfolio is divided among many strategies and quick loss in one of them makes only small loss in overall portfolio level.

Due to the presence of numerous economy and finance environments changes, in our investigations we used two months data for validation (estimation $\widehat{Sh}^l$ and selection of m best TSs). Later two months data were used for testing the trading

algorithm, i.e. estimation of Sh^l).

In real world trading with sudden environmental changes, we have *two sources of errors* that are affecting difference between evaluation of $\widehat{Sh}$ and true Sh. The first source is finite sizes of learning and validation segments of historical data. This problem was considered in previous section. The second source is the data variation. Investigation of contemporary financial time series showed that Sharpe ratios estimated for two earlier and later two months length data segments are very weakly correlated. Often correlations are even negative. In Figure 3 we present a "successful example" with absence of visible correlation (2+2 months of 2016 spring data). Like in the previous section we are interested in dependence of the mean Sh value on m. Due to random generation the subsets of TSs, single $\widehat{Sh}$ and Sh graphs are notably scattered.

To reduce fluctuations we generated $M = 5{,}0000{,}000$ subsets composed of N_b TSs for each of them. Then for each subset we calculated the estimates $\widehat{Sh}$ (two training months were used) and the true values, Sh (again two extra months were used). As a result, for further analysis we obtained $2 \times 5{,}000{,}000$ dimensional array of the Sharpe ratio values. For each particular m value chosen from *a priori* fixed vector $mm = [50, 100, 200, \ldots, 100{,}000]$ we used binomial coefficients to calculate true mean Sharpe ratio, $\widehat{Sh}(m)$, for all possible, $M!/m!/(M-m)!$, combinations. In Figure 4 we present graphs of the mean values calculated for three TSs subset sizes, $N_b = 20, 40$ and 90.

For each of the TSs subset size, N_b, we observe peaking effect. Curves in Figure 4 remind the curves, presented in Figure 2 calculated for the 2-D Beta model. The most obvious decrease in the performance of the TSs selected is observed for small sizes of the subsets. Small training strategies subsets result smaller portfolio accuracy and smaller correlations between the $\widehat{Sh}$ and, Sh values. Therefore, the peaking effect appears very early (curve for N_b=20 in Figure 4). With reserved increase in N_b, the portfolio performance increases (curve for N_b=40 in Figure 4). Notable increase in size of the TSs, m, increases the complexity of the optimization algorithm. In such a case, we obtain overtraining (overfitting) effect once more: for N_b=90 the true Sharpe ratio graph is notably below as that for N_b=40. This result confirms: too great increase in complexities of the TSs subset size, N_b, and in the algorithm used to select the best subset, m, causes a negative effect. Figure 4 illustrates that for successful employment of the random search procedure (N_b=40, $m \approx 400{,}000$) an average of the true (validation set estimates) the Sharpe ratio values obtained for these optimization procedures parameters, Sh_{best}=3.72, (see Figure 4). It is notably higher value as average of Sh_r ($r = 1, 2, \ldots, 5{,}000{,}000$). For practical application one needs to know optimal values of the optimization procedure parameters (N_b, m, L). Studies in this direction have already begun.

The peaking effect had been noticed in statistical, pattern recognition, data visualization, design of prediction rules, neural networks [10, 11, 12, 13]. We expect that this conclusion can be applicable also for other research and development disciplines where data mining and machine learning are used. Therefore, in dynamic Portfolio design, we need to pay attention both into development of faster optimization methods and ways to determine the optimal complexity of the optimization algorithms.

In the introduction we mentioned two heuristically based algorithms, the independent training strategy selection (**A**) and the feed forward selection algorithm Comgen

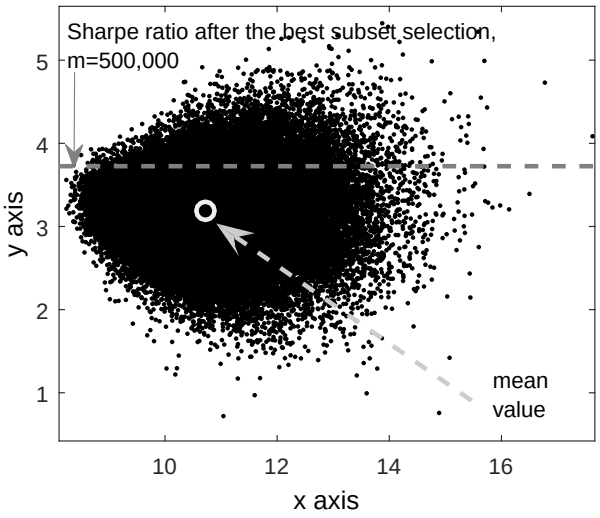


Figure 3. 2-Dscatter of Sharpe training and test Sharpe ratios calculated for M=100,000 subsets.

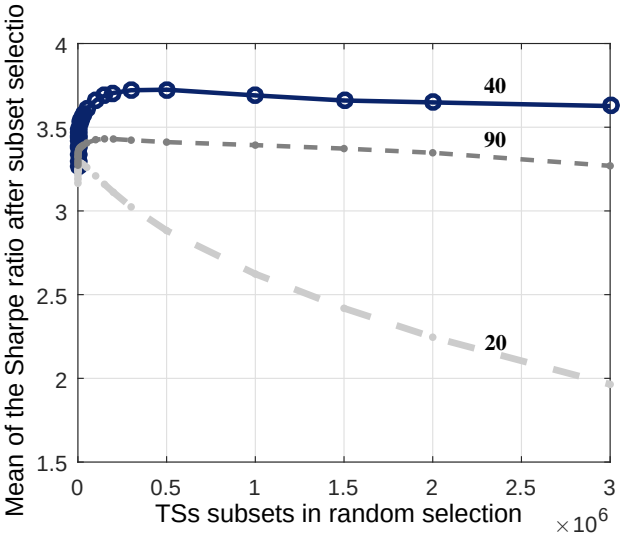


Figure 4. The Sharpe ratio after the random best subset selection.

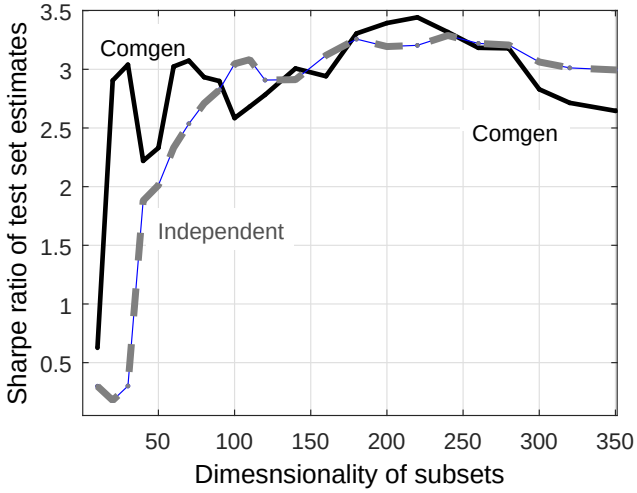


Figure 5. Sharpe ratio of the best TSs where trading systems were ranked individually or selected by the Comgen algorithm.

(**B**). In comparison with random search they work much faster. Their complexity can be characterized by the number of TSs selected, N_b. In Figure 5 we present variation of the test set Sharpe ratio in dependence of dimensionality of the TSs selected. The dependency curves are much more scattered as that in Figure 4, since only short, two month length data sequences were used to calculate $\widehat{Sh}$ and Sh.

Both graphs in Figure 5 exhibit the peaking behavior. For both algorithms the optimal complexity of the Portfolio inputs is found somewhere in the range $200 - 250$ trading strategies. The true Sharpe ratios detected are a bit lower as that obtained by relatively slow random search procedure. In the practical automated daily investment work the random search (algorithm **C**) can be easily applied since modern laptop computers calculate the family of curves similar to that in Figure 4 in 10 - 20 minutes.

5. Concluding remarks

This paper explores the secondary over-fitting effect that is very acute of $1/N$ portfolio design. It is an adaptation to validation data set used to select the best subset of inputs and/or the best algorithm to calculate the portfolio weights. Ignoring this up to now unexplored effect constitutes a big headache for portfolio managers as constructed portfolios do not behave in the way they were supposed to. In theoretical analysis we examined the random portfolio inputs optimization procedure and derived the equation to calculate the mean Sharpe ratio in dependence of (on) the number of portfolio inputs N_b, the validation sample size L used to estimate Sharpe ratios

of each particular subset of inputs and the number, m, of randomly generated N_b-dimensional portfolio inputs from their N-dimensional set. This equation was adapted for practical calculations of the mean Sharpe ratio when both, the probabilities of the true and estimate Sharpe ratios, are calculated from the 2-D Beta distribution model. It was demonstrated that with an increase of portfolio complexity, N_b, and complexity of optimization procedure, m, we can observe the over-fitting phenomenon.

Theoretically based conclusions were confirmed by experiments with high dimensional real world financial data and suggest several recommendations for future research and practical work. Due to the presence of numerous economy and finance environmental changes the 2-D scatters of Sharpe ration evaluated on training and validation data in diverse time periods often show zero or even negative correlations. Consequently, the practitioner should be careful: sometimes even a negligible optimization of the portfolio inputs subset can worsen the result. For that reason the practitioner should examine numerous subsequent in time 2-D Sharpe ratio scatters and be cautious with for the portfolio inputs optimization. Therefore a prudent analysis of changes in preceding historical data is very important. Preliminary experiments demonstrated that paying an attention to validation data size and knowledge about character of possible data changes could lead to novel useful ways of the portfolio management and determination of the portfolio size and parameters of optimization rules.

Acknowledgements

This work was supported by the Research Council of Lithuania (grant MIP-100/2015)

6. References

[1] Markowitz, H.M., *Foundations of portfolio theory.* The journal of finance, 1991, **46**(2), pp. 469–477.

[2] Reilly, F.K., Brown, K.C., *Investment analysis and portfolio management.* Cengage Learning, 2011.

[3] Raudys, S., *Portfolio of automated trading systems: Complexity and learning set size issues.* IEEE transactions on neural networks and learning systems, 2013, **24**(3), pp. 448–459.

[4] DeMiguel, V., Garlappi, L., Uppal, R., *Optimal versus naive diversification: How inefficient is the 1/n portfolio strategy?* Review of Financial Studies, 2009, **22**(5), pp. 1915–1953.

[5] Haley, M.R., *Shortfall minimization and the naive (1/n) portfolio: an out-of-sample comparison.* Applied Economics Letters, 2015, pp. 1–4.

[6] Guyon, I., Elisseeff, A., *An introduction to variable and feature selection.* Journal of machine learning research, 2003, **3**(Mar), pp. 1157–1182.

[7] John G H, Kohavi R, P.K., *Irrelevant features and the subset selection problem.* The journal of finance, 1994, pp. 121–129.

[8] Raudys, A., Pabarškaitė, Ž., Discrete portfolio optimisation for large scale systematic trading applications. In: *Biomedical Engineering and Informatics (BMEI), 2012 5th International Conference on*, IEEE, 2012, pp. 1566–1570.

[9] Bailey, D.H., Borwein, J.M., de Prado, M.L., Zhu, Q.J., *Pseudomathematics and financial charlatanism: The effects of backtest over fitting on out-of-sample performance.* Notices of the AMS, 2014, **61**(5), pp. 458–471.

[10] Bradley, P.S., Fayyad, U.M., Mangasarian, O.L., *Mathematical programming for data mining: Formulations and challenges.* INFORMS Journal on Computing, 1999, **11**(3), pp. 217–238.

[11] Jackowski, K., Wozniak, M., *Algorithm of designing compound recognition system on the basis of combining classifiers with simultaneous splitting feature space into competence areas.* Pattern Analysis and Applications, 2009, **12**(4), pp. 415–425.

[12] Tetko, I.V., Livingstone, D.J., Luik, A.I., *Neural network studies. 1. comparison of overfitting and overtraining.* Journal of chemical information and computer sciences, 1995, **35**(5), pp. 826–833.

[13] Raudys, S., *Experts' boasting in trainable fusion rules.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 2003, **25**(9), pp. 1178–1182.

Schedae Informaticae Vol. 25 (2016): 189–207
doi: 10.4467/20838476SI.16.015.6196

tfml 2017
theoretical foundations
of machine learning, Kraków

Search for Resolution Invariant Wavelet Features of Melanoma Learned by a Limited ANN Classifier

Grzegorz Surówka
Faculty of Physics, Astronomy and Applied Computer Science,
Jagiellonian University, 30-151 Kraków, Poland
Institute of Technology, State Higher Vocational School, 33-300 Nowy Sącz, Poland
e-mail: *grzegorz.surowka@gmail.com*

Abstract. This article addresses the Computer Aided Diagnosis (CAD) of melanoma pigmented skin cancer. We present back-propagated Artificial Neural Network (ANN) classifiers discriminating dermoscopic skin lesion images into two classes: malignant melanoma and dysplastic nevus. Features used for our classification experiments are derived from wavelet decomposition coefficients of the image. Our research objective is i) to select the most efficient topology of the hidden layers and the network learning algorithm for full-size and downgraded image resolutions and, ii) to search for resolution-invariant topologies and learning methods. The analyzed classifiers should be fit to work on ARM-based hand-held devices, hence we take into account only limited learning setups.

Keywords: melanoma, CAD, wavelets, ANN

1. Introduction

Computer aided classification of the benign form (displastic nevus) and the malignant form (melanoma) of the pigmented skin lesions plays a crucial role in the prevention of human melanoma. Melanoma, as contrasted with the other forms of skin cancer, is extremely dangerous due to early metastases. Early diagnosis of this tumor is a life-saving factor. Lack of specialists, too late detections and an increased melanoma

Received: 11 December 2016 / Accepted: 30 December 2016

morbidity rate have become a medical problem for some time and, on that account, a challange for computer assisted diagnosis (CAD) [1, 2, 3]. The standard therapy, biopsy, is not feasible for all the patients due to treatment costs and some health reasons. Even if affordable, excision must be done at an early stage, when standard clinical diagnosis based on the ABCD criteria may fail [4]. For that reason, analysis of skin lesions images have become a useful diagnostic tool. The most popular and cheapest form of lesion screening is dermoscopy (ELM-Epiluminescence Microscopy) [5]. This technique uses optics and white light illumination of the lesion and stores the skin image usually on a computer. This is extremely useful for comparing past and current advances of the lesion. Some advanced instruments can illuminate the lesion at different angles or with a set of different wavelengths. This helps penetrate deeper layers of the skin to reveal its spatial structure [6, 7, 8]. Their coverage is however limited and the most common are the cheapest, handy dermatoscopes.

Stages of pigment cells atypia are recognized by medical doctors with help of the semi-quantitative descriptive measures: ABCD(E), the 7-Point Checklist, Menzies and other less common [9]. Clinical diagnosis of Melanoma depends on the appearance of classic dermoscopic features but unfortunately the so called early melanomas are mainly featureless [10].

The computer aided melanoma diagnosis is roughly divided into two classes of methods:

- Segmentation-based methods: they assume that melanoma can be visually examined based on shape, color and structure content. There are two main reasons why those methods may fail:
  - different illumination and optical magnification plus noise and artefacts (hair, blistery areas) can make such observations/analyzes not sensitive enough [11],
  - search for featureless melanomas is infeasible.
- Texture-based methods (wavelet- or curvelet- based): they attempt to search for some frequency and scale information of the skin texture [12] to distinguish between benign and malignant skin progression [13, 14]. The related works referenced in this article (Sec.2) prove that especially wavelet-based methods are robust and show high performance in the melanoma classification.

There exists a big market for (para)medical smartphone applications. Both medical doctors, who are interested in medical decision-supporting tools, and plain consumers, who want to quickly 'self-diagnose' and be aware of personal health issues, show interest. The self-examination/self-diagnosing optical extensions and grip panels for mobile phones are very popular. There is also a growing market for handy dermatoscope-like devices with optics and ARM-based processors for a 'mobile' CAD and melanoma diagnosis [15, 16].

Since the developed machine learning and image recognition and interpretation algorithms may demonstrate high complexity and small handheld devices have limited processing power and memory, it is of great importance to search for methods that analyze optimally both full-size dermoscopic images and preserve high efficiency also in downgraded image resolutions.

In Sec. 2 research on wavelet based features and methods is presented. One conclusion of the latest advances [17] is that different wavelet families perform not equally in terms of the melanoma classification efficiency. The reverse bi-orthogonal (RBio) and bi-orthogonal (Bior) wavelets prove to be the most efficient, robust, and resolution invariant wavelet families for the machine learning of dermoscopic images. In [17] ensemble learning of different model types and optimized for various quality measures is performed for the wavelet features. In this work we want to check efficiency and classification performance of a single, homogeneous model taking into account the most successful wavelet families concluded in [17]. We pick the RBio3.1 wavelet base and study back-propagated artificial neural network (ANN) classifiers of the melanoma (RBio3.1)-based features with limited topologies for the hidden layers to take into account execution platforms with limited processing resources.

Within these bounds our objectives are:

- to select the best NN topology and learning method in terms of absolute melanoma classification performance upon condition of efficient cost/time of ANN back-propagation learning for three different resolutions of the dermoscopic images,
- to search for a resolution-invariant model of ANN for the original clinical data and the descendant downgraded image resolutions.

Below we review previous research on both wavelet-based features of melanoma and NN-based melanoma classifiers. Then we show methodology of our machine learning experiments and present and discus the results.

2. Related work

The first contribution to wavelet based decomposition of melanoma dermoscopy images belongs to Patwardhan et al. [18, 19]. This group successfully studied binary classification models for benign nevus and melanoma by decomposing different frequency scales of the skin texture (wavelet packets). This approach also known as selective wavelet trees in each iteration decomposes all the sub-bands of the pigmented skin texture rather than the low-frequency sub-band only (recursive algorithm). Since that time the wavelet-based features have also been successfully studied by other groups [20, 21, 22, 23, 24, 25, 26, 27, 28]. The binary classifiers from Patwardhan [18, 19], and early contributions from [20, 21, 22], were using only one wavelet base (Daubechies 3) to build classification models. Recent research on efficiency of different wavelet bases in classification of melanoma and benign dermoscopy images shows that bi- and reverse bi-orthogonal wavelets outperform other wavelet families. In this article we use the reverse bi-orthogonal wavelet 3.1 (RBio3.1) as the transform base.

Neural learning of wavelet-based features at various stages of melanoma development was first studied by Walvick et al. [13]. Their feature set was then optimized by removing redundancies with help of principal component analysis. A feed forward neural network trained with the back propagation algorithm was then used in the classification process to obtain better classification results. In [29] a two level wavelet transform was performed on the segmented image in order to take into account only the lesion region and not the surrounding tissue. In order to maintain the

original image size, a stationary wavelet transformation was used which suppresses down-sampling, producing a wavelet matrix of the same size as the input image. For the approximations and details components mean and variance of wavelet coefficients were calculated and normalized (0,1). Three wavelets from four different families (Daubechies, Coiflet, BiorSplines, Symlet) were tested to see the impact on the classification. The feed forward back propagation neural network consisted of 8 input nodes (for 8 features), 2-5 hidden nodes and one output node and was trained/tested by the LOO technique. The overall classification accuracy was about 83%-86% for all the wavelet bases for the best performing network with 3 hidden nodes.

Also [24] employed the variance and mean of wavelet decomposition coefficients to extract useful features of dermoscopy images. Those coefficients were used as the inputs of neural network. Results showed 90% ability in distinction between benign and malignant lesions.

Work [30] segmented suspicious lesions from the normal skin and extracted features that distinguish malignant melanomas from benign moles with 2D wavelet transform in Matlab. Some selected features formed the input to an ANN classifier. Authors claim accuracy to be 84% but no details about the procedures are available.

An interesting research was done in [31]. This paper proposes a melanoma classification system based on coefficients created by wavelet and curvelet decompositions. The curvelet transform returned the set of curvelet coefficients indexed by scale, orientation and location parameters. Fast Discrete Curvelet Transform via wedge wrapping was used to find the features. Recognition accuracy of the three layers (40-25-10) back-propagation neural network classifier with wavelets was 51.1%, which is a surpisingly low value, and with curvelets 75.6%, not high neither.

Good indication for lesion malignancy is border irregularity. Work [32] presented a contours harmonic wavelet coefficients forming a sequence of multi-scale roughness descriptors to characterise the distribution of energy across contour's line. The descriptors for regular and irregular borders were compared with ground truth from experts. Differentiating between benign and malignant lesions according to those descriptors led to maximum classification accuracy of 93.3% with sensitivity of 80%.

Similar work was reported in [14] where wavelet decomposition was used to extract energy distributions among different wavelet sub-bands. Statistical and geometric irregularity descriptors based on the wavelet coefficients were used to model structural components from the contour. The effectiveness of the descriptors was measured using the Hausdorff distance between sets of data from melanoma and mole contours. The best descriptor outputs were input to a back propagation neural network to construct a combined classifier system. Experimental results showed that the selected thirteen multi-scale features with small sample set produced the best discrimination performance with AUC=0.89 and best specificity of 90%, and sensitivity of 83%.

Artificial neural networks (ANN) in general are widely used in various pattern recognition problems, also in CAD. There are different types of neural systems well suited to classify visual features from (medical) images. In this short review we list only standard back-propagated neural networks (not e.g. CNN). In the melanoma classification problem ANNs are used as one of the following:

- the main classification engine (melanoma-benign lesion) for all/selected features,
- dedicated expert system for a single segmentation task (e.g. edge detection), or

- auxiliary ML system for initial/post- analysis and verification.

First ANN classification of melanoma and benign lesions comes from [33] where about 80% correct classifications were reported. The network topology (14-X-1, X-hidden neurons) accepted two asymmetry- and twelve color-based features.

In [34] authors analyzed basic, shape and color features with different normalization conditions and concluded that on both dichotomous and trichotomous tasks, the ANNs performed (sensitivity: 91%, specificity: 94% for 1619 lesion images) similarly as logistic regression and SVMs, and better than k-nearest neighbors and decision trees.

In [35] authors analyzed 48 parameters belonging to four categories: geometry, color, texture, and color clusters inside the lesion and performed step-wise feature selection to identify an optimal subset of 10 variables (starting from the most significant: red multicomponent, decile of red, border homogeneity, mean value of red, grey-blue areas, contrast, interruptions of the border, mean skin-lesion gradient, background regions imbalance, variance of the border gradient). The clinical/dermoscopic equivalents of those variables are: multicomponent pattern/homogeneity, lesion darkness, border cleanliness, mean color of the lesion, grey-blue areas, network analysis, variation in the border cleanliness, grading of the border, color asymmetry, intensity in the border interruptions. Distinguishing melanoma from benign lesions with these optimal features gave the maximum sensitivity of 93% and specificity 92.75%.

Project DANAOS [36] aimed at analyzing robustness of the NN-based machine learning system with respect to multi-population lesion samples. Its authors concluded that the performance of their NN expert system is comparable with that of clinicians, with the average AUC of 84,4%.

Rajab et al. [37] investigated the neural network edge detection (NNED) in the iterative thresholding segmentation. They drew conclusions on the method performance over a range of different border irregularity properties and signal-to-noise ratio.

Preprocessing of dermoscopic images through the Fourier and log-polar transforms was used to build an unsupervised image segmentation and image registration system where neural networks and discriminant analysis were used to find the best classification rules for the extracted border- and color-based features [38].

[39] described a multi-layer perceptron classifier for melanoma recognition with accuracy of 77.7%. The number of features for classification was optimised to only five which speeds up the CPU time.

In [40] a service on the Internet was introduced to upload dermoscopy images for on-line extraction of the tumor area and calculation of 428 global features (color, symmetry, border, and texture ABCD) for the characterization of the lesion. The extracted features classified the lesion as melanoma or nevus using a neural network classifier achieving a sensitivity of 85.9% and a specificity of 86.0% on a set of 1258 dermoscopy images using cross-validation.

A co-operative neural network-based edge detection on enhanced colors and contrast of the image is reported in [41]. As supervisors (ground truths) three expert dermatologists were used.

An automatic neural skin cancer classification system was developed in [42] for dermoscopy imagess and optimized for different types of neural network topologies and different preprocessing modes. The authors reported best recognition accuracy

of the 3-layers back-propagation neural network classifier as 89.9% and of the auto-associative neural network as 80.8%. In the system some features were extracted through 2-D wavelet packet decomposition under performance tests of seven different wavelet bases (the best wavelet base was Bior5.5, and the most stable experimentally Db1 or Db10). Some opimization was done to the number and structure of the hidden layers. The best topology was reported for the three hidden layers with 40-25-10 neurons.

In [43] a back-propagation neural network (BPN) is used for segmentation. The results are compared with the ground truth images which shows that BPN has slightly worse segmentation accuracy and slower training period than Extreme Learning Machine (ELM).

[44] claims rather low accuracy of the melanoma discrimination with back propagation neural network which yields 60%-75%. This outcome is outperformed by the SVM models.

Group [45] segmented dermoscopic images using Maximum Entropy Threshold and extracted features using Gray Level Co-occurrence Matrix (GLCM) for classification into cancerous or non-cancerous cases using back-propagated neural networks (BPN). The reported accuracy is 88%.

Paper [46] proposed the flow: feature extraction, dimensionality reduction and classification to discriminate skin lesions into 'normal' and 'abnormal' skin cancer classes. In the first stage authors used discrete wavelet transforms, in the second stage PCA. In the classification stage a feed forward back-propagated artificial neural network and a k-nearest neighbor paradigmes were applied. Accuracy of those experiments was: 95% (ANN) and 97.5% (kNN).

Discrimination between the six Menzies color classes in the calibrated RGB dermoscopy images were studied in [47]. The JeffriesMatusita and transformed divergence separability distances were used to evaluate the color class separability. A nonlinear cluster transformation allowed almost the total separation of each color class in the feature space. Several neural networks in competition were used as classifiers. Classification achieved 93% of sensitivity, 62% of specificity and 74% of accuracy (average). Authors claim that it might be possible to evaluate a lesion based on the presence of Menzies colors in the dermoscopic image, mimicking the human diagnosis.

In [48] statistical features and dermoscopic features (ABCD: Asymmetry, Border, Color and Diameter) for detection and diagnosis of melanoma were used. Segmentation-based thresholding plus statistical feature extraction using GLCM were used to calculate the Total Dermoscopy Score (TDS). The combined result of the TDS parameter and a neural network classification yielded accuracy of 88% which was claimed to be efficient for the skin cancer detection and diagnosis.

[49] presents a melanoma detection system working in two phases: the first phase detects whether the skin lesion is of pigment type, the second phase distinguishes between malignant melanoma and benign skin lesions. The reported classification results from the neural network are about 98% (phase 1) and 93% (phase 2). Although algorithms and methods are presented step-by-step some details cannot be derived. The high results are controversial.

[50] analyzes the recognition performance of three different classifiers: support vector machine (SVM), artificial neural network (one hidden layer, sigmoid transfer function) and k-nearest neighbor. From experiments run on a database of more than

5000 dermoscopy images they concluded that the SVM approach outperforms the other methods reaching an average recognition rate of 82.5% comparable with those obtained by skilled clinicians. So this is an upper limit for average recognition rate by neural network.

In [51] several methods of melanoma classification were proposed: a multilayered perceptron, a Bayesian classifier and the K nearest neighbors algorithm. These methods worked independently and also in combination making a collaborative decision support system. The performance factors obtained for seven neurons in a single hidden layer were: classification rate: 86.73%, sensitivity: 78.43%, specificity: 95.74%, slightly less than in the collaborative method: classification rate: 87.76%, sensitivity: 78.43%, specificity: 97.87%.

In the review [52] authors surveyed computer-based systems according to acquisition, feature definition, extraction and skin lesion classification. Authors concluded that some widely used lesion parameters like lesion size, shape, color, and texture do not correspond to known biological phenomena and the structural patterns that are considered essential for manual lesion categorization are absent in the analysis due to their complexity. From the analyzed machine learning methods (discriminant analysis, neural networks, support vector machines) SVM performed the best. Neural Networks using two principal components as input produced 85% correct matches (sensitivity 79%, specificity 90%) for the vertical growth phase of melanoma development and 94% correct matches (sensitivity 86%, specificity 90%) for the radial growth phase.

Using meta-analytical methods Authors of [53] compared the diagnostic accuracy of the different dermoscopic algorithms with each other and with the artificial intelligence methods for the detection of melanoma. They concluded that pooled sensitivity for artificial intelligence was slightly higher than for dermoscopy (91% vs. 88%) and pooled specificity for dermoscopy was significantly better than artificial intelligence (86% vs. 79%). There was no significant difference in the diagnostic performance of various dermoscopy algorithms (ABCD, 3-point checklist, 7-point checklist, Menzies score). A useful 1994-2006 summary for characteristics of the included studies is included.

Authors of article [1] classified and researched literature on the CAD systems for melanoma identification. They tabularized lots of references according to the generic steps of CAD: image preprocessing, border detection, features/descriptors extraction and classification. Low opinion was expressed on the lack of common standards and absence of benchmark datasets for standardized algorithm evaluation.

In the review [2] the state of the art of melanoma CAD was examined: in-vivo imaging techniques, image acquisition, pre-processing, segmentation methods, feature extraction and selection, and classification of dermoscopic images. Of high value are the indications of various conditions that affect the technique's performance. The case of artificial neural networks for the melanoma classification is broadly represented in the following references.

Article [54] reviews to a lesser extent the 2011-2014 works in the skin cancer detection. Classifier performance results from other existing melanoma CAD systems can be found in [55].

The last review [3] gives a vast look at different feature types and classification methods. It reports and classifies the 2007-2015 studies paying attention at refering

them to the global and local patterns.

3. Data and Methods

3.1. Data

The database collected for this study included anonymous images of the moles from 185 patients of one private clinic in Poland (the formal agreement forbits publications of the location details to keep the patients data unidentified). The examinations were performed with plain digital camera with an extra dermoscopy extension and immersion liquid to remove light reflections. The primordial JPEG pixel resolution was 2272x1704 and the RGB color depth 24-bit. The resection and hist-pat examination of the moles allowed to assign labels to 102 malignant melanoma and 83 dysplastic nevus cases. The 83 non-melanoma images were picked up randomly from the predominant majority of about 2000 displastic lesions. In our analysis there were no 'unknown' or 'don't care' labels.

Melanoma incidence rate may fluctuate over countries, but clinical statistics show an average of about 5% melanoma images as a fraction of all the dermoscopic images of the pigmented nevi. This means that the melanoma class is under-represented compared to the benign class. Learning classifiers from such cases would require special rules to properly treat the imbalanced class i.e. to draw equal attention to the minority class [56, 57, 58]. In this experiment we took the whole statistics for the minority class (melanoma, 102 cases) and then randomly selected the 83 non-melanoma images from the larger pool (about 340) covering the pigmented benign lesions (dysplastic junctional nevus, displastic compound nevus). In clinical practice only the above mentioned benign pigment cases are confused with melanoma. The ground truth was in each case the hist-pat examination.

In this work we do not analyze in detail the impact of data balancing techniques for the classification of malignant melanoma (see e.g. [58]) but we estimated if such impact affects our classifiers. Three different under-sampling procedures of the majority class in the data space were performed to balance the data vectors learned by the classifier before cross validating the classifier models. No major change in classification performance ($< 2\%$) was observed. This allows us to treat the data imbalance problem in our experiment under control. This of course does not mean that there is no impact of the data imbalance problem at all but that the bias comes preferably from the 'clinical' source and not the data statistics or procedures.

In the analysis there were three sets of images: the original set 2272x1704 (A) and the two downscaled sets (by averaging neighbor values in 2x2 elements) of 1136x852 (B) and 568x426 (C) pixels respectively.

3.2. Features

There were no apparent artefacts on the analyzed dermoscopy images (black borders, hairs, droplets of immersion fluid, etc.) or there were few negligible distortions so no preprocessing tasks to the images took place.

To support wavelet transformations the dermoscopy images of all three sets (A, B, C) were transformed to indexed images with linear, monotonic color maps of double precision numbers. Each iteration of the wavelet decomposition downscales the input image by 2 both in rows and columns and three such iterations were done.

Wavelet decomposition of signals is well established in theory after works of Mallat [59], Daubechies [60] and the others (Gabor, Morlet). It is widely applied especially to discrete signals in the form of DDWT-Discrete Dyadic Wavelet Transform. This transform is widely used to analyze the signal structure, signal de-noising and compression capabilities.

Images are two-dimensional signals and the wavelet transform to the images are done according to the Mallat algorithm [59]. One iteration of this algorithm produces 4 downscaled sub-images which can be considered as LL, LH, HL and HH filters (L-low-pass, H-high-pass filter) after one-dimensional wavelet transform on the rows and then on the columns.

DDWT can be applied recursively to the low-frequency sub-band only, but in our analysis we used the wavelet packets so each of the four filters was subject to further wavelet decompositions (not only LL).

Altogether in the three iterations 1+4+16=21 different transformation branches were produced. In one branch the following 12 simple features were calculated: (e_i, i=1,2,3,4) - energies of the sub-images, (e_i/e_{max}, i=1,2,3,4) - maximum energy ratios and ($e_i/\Sigma e_k, k \neq i$, i=1,2,3,4) - fractional energy ratios, after [18, 19, 20]. Energy was defined as a sum of absolute values of the pixels. This procedure was repeated for the three sets A, B, C of different image resolutions yielding 21x12=252 attributes in each single set.

For reasons presented in the Introduction for the skin texture analysis we took RBio3.1 wavelet base. Reverse bi-orthogonal wavelets (wavelet pairs) have the property of perfect reconstruction i.e. if X-image, A-reconstructed image of approximations and D-reconstructed image of details, then X=A+D. This property is possible due to two separate filter sets, one for decomposition and another one for image reconstruction. This wavelet is symmetric function and is not orthogonal ($X^2 \neq A^2 + D^2$).

Search for the best subset of features [61] can be used to i) reduce bias (overtraining), ii) reduce computational burden and iii) enhance classification performance. This is usually done because the simplest approach, an exhaustive or random search to evaluate the best feature set, is infeasible or even computationally prohibitive. In this work we do not take advantage of any feature selection or extraction algorithms. This follows the results presented in [62] where widely known feature selection mechanisms (CFS, PCA, GSFS) were applied to melanoma classification problem. It was concluded that although application of feature selection algorithms may reduce the complexity of the classification, the performance is highly dependent upon the classifier. Therefore, it was opted to use all the features and preserve them for some late-selections. It is also the objective of this work - to search for efficient ANN clas-

sifiers in terms of their topologies and/or error minimization approaches and not by the feature selection of the data base.

3.3. Algorithm

Artificial neural network is a black-box approach to the knowledge acquisition and can model complex relationships between inputs and outputs. As a learning method it is a 'standard' and well recognized approach [63, 64, 65]. Our objective was to collate this 'simple' learner with ensemble models discussed in [17] for the same (high-performing) wavelet families (Bio/RBio). Since both the topology (hidden layers) and the learning details (the choice for the learning function) can bias the classification performance of melanoma, we examined and compared different setups. Since neural computations can be burden on CPU/memory and can be time-consuming, we arbitrarily focused on one static feedforward back-propagated artificial neural network (ANN) without any recursive or meta (deep) learning extensions. Parallel processing (both CPU- and GPU-based) was implemented.

In our study we used a static feedforward back-propagation artificial neural network (ANN) to classify the dermoscopy images based on the calculated 252 wavelet features. As a preprocessing phase normalization of the the input was done and the labels were fixed to '1' (Malanoma) and '0' (Dysplastic Nevus).

The ANN structure was:

- 252 input nodes that represent the wavelet features,
- a number of hidden nodes grouped into one or two hidden layers ($\mathcal{T}$opol) subject to change ($\mathcal{T}\text{opol} \in \{10, 20, [10-10], [10-20], [20-20]\}$),
- 2 output neurons, each one activated on the vectors belonging to one class only (so in the binary classification in mutually exclusive way).

Two setups were analyzed within the scope of the activation functions:

- NN1: hyperbolic tangent sigmoid transfer function (a1) for the hidden layers and linear activation function (a2) for the output layer [65]. This is a generic ANN to model any kind of input to output mapping.
- NN2: hyperbolic tangent sigmoid transfer function (a1) for the hidden layers and also hyperbolic tangent sigmoid transfer function (a1) for the output layer. This ANN should perform more efficiently while classifying inputs according to target classes.

As a performance function ($\mathcal{P}$erf) for NN1 and NN2 we used the 'standard' mean square error (mse) and the cross-entropy (ent). The latter case was an attempt to check how 'information-gain'-based learning objective affect both the classification and computational performance. The binary cross-entropy (ent) is calculated as $-p * log(p) - (1-p) * log(1-p)$ (p-a priori probability of one class) and it heavily penalizes outputs that are extremely inaccurate, with very little penalty for fairly correct classifications.

For the sake of cross-validation (CV) we randomly divided our data into training (70/100), validation (15/100) and testing (15/100) set. Every epoch all the training samples were presented simultaneously to the network to train it. The validation data was used to evaluate the prediction errors hence to optimize and update the weights in the backpropagation phase. The testing set was used to calculate all the performance coefficients. The algorithm was as follows:

1. Separate data into three sets: training, validation, testing
2. Build a network (NNx, $\mathcal{P}$erf, $\mathcal{T}$opol, $\mathcal{L}$earn)
3. Determine the parameters (stoping conditions: maximum number of epochs, maximum training time; learning conditions: learning rates, etc. Loop (4-6) (average over initial conditions, #(CV partitions)=6)
4. Initialize the weights and biases randomly (but with same seed to compare different setups) Loop (5-6) over epochs until *stop*1
5. Train the network with the train data
6. Compute the network performance on validation data and back propagate the error to update the weights
7. Use the network (compute the network performance on the test data)

As *stop*1 standard conditions were applied: the maximum number of epochs reached, maximum time exceeded, performance gradient fallen below e^{-6}.

Several training algorithms and refinements for ANNs have been proposed in the literature to enhance the convergence speed and reduce the generalization error of the network [65, 64, 66]. In this work we do not discuss mathematical properties of those algorithms, rather focus on the classification interest when they are run with 'standard' base parameters. The analyzed backpropagation training algorithms ($\mathcal{L}$earn) were: (shown with initial parameters where applicable)

L1 Levenberg-Marquardt ($\mu = 0.001$),

L2 Bayesian Regularization ($\mu = 0.005$),

L3 Broyden-Fletcher-Goldfarb-Shanno,

L4 Conjugate Gradient with Powell-Beale restarts,

L5 Fletcher-Powell Conjugate Gradient,

L6 Polak-Ribiére Conjugate Gradient,

L7 Gradient Descent ($lrate = 0.01$),

L8 Gradient Descent with Adaptive Learning ($lrate = 0.01$),

L9 Gradient Descent with Momentum ($lrate = 0.01, momentum = 0.9$),

LA Variable Learning Rate Gradient Descent ($lrate = 0.01, momentum = 0.9$),

LB One Step Secant,

LC Resilient backpropagation ($lrate = 0.01, \Delta = 0.07$),

LD Scaled Conjugate Gradient.

Although in our ANN learning we meet the CV paradigm and validate during training, to promote better generalization for the $\mathcal{P}$erf function 'mse' (mean squared error) we applied also the performance regularization ratio (0.01) which takes into account not only minimizing the error but also the weights and biases (for L2 set to 0).

Due to performance aspects the Matlab library was used for calls to the ANN training algorithms. The code was run on the CUDA-based NVidia GTX 1070 GPU.

In our brief review of the related literature we went through the advances of both the ANN and mobile devices for the melanoma CAD. Unfortunately, there are no methodical studies how the ANN structure (hidden layers) affect the melanoma classification performance. Arbitrary values for both the number of the hidden layers and the number of neurons in the hidden layers are published. Usually 2- and 3-hidden layers are presented. Taking into account the computational burden reported and own attempts on both CPU- and GPU-based parallel computing platforms and, last but not least, performance analyzes and benchmarks for the ARM-based mobile devices, we limited ourselves to up to 2 hidden layers with pretty small (up to 20) neurons on each layer.

4. Results and Discussion

We analyzed two networks: NN1 and NN2 (see Section 3.3.) each tought according to two performance measures: (mse) and (ent). The best CV performance for the discrimination of Melanoma from Dysplastic nevi is reached with NN2(mse) which is full sigmoid-like network. Performance of NN1(mse/ent) and NN2(ent) is slightly worse (by about 8-3%) and more sparse in terms of ($\mathcal{L}$earn, $\mathcal{T}$opol) coverage. Below we present the results for the best performing NN2(mse).

Our objective was to find the best performing ANN for the classification of melanoma dermoscopy images under the assumption that the algorithm for using the (trained) network and, as a next step, even the network training process take place on an ARM-based mobile device. For that reason only limited topologies were taken into account (starting with 10 hidden neuron on one hidden layer up to 20x20 hidden neurons on two layers). Although we take into account only about 10, 20, 100, 200, and 400 weights this is not a small selection compared to the literature. Close by performance our experiments show coarsely how complex in terms of epochs and time the back-propagation algorithms can perform. Statistics for the 5x13=65 different setups shows the following grouping of results in terms of pairs ($number\_of epochs$, $setups\_finished$): $(10, 17)$, $(15, 31)$, $(20, 38)$, $(30, 44)$, $(50, 45)$, $(100, 51)$. As a necessary condition for further analysis we took the threshold of 20 epochs (median).

In Table 1 we show numerical results of AUC for resolutions A, B and C for the five setups of the hidden layers and for the thirteen different back-propagation algorithms assumed that the number of epochs is below 20. Absence of L7-LA proves that methods based on (variations of) gradient decent converges very slowly (maximum number of epochs even above 1000) and are out of range of mobile hand-held devices and are not feasible as CAD applications.

Table 1. Numerical results for AUC for (from top to bottom) resolutions A, B, C as a function of $\mathcal{T}$opol and $\mathcal{L}$earn. Filter: #Epoch<20

$\mathcal{T}$opol	AUC $[10^{-2}]$												
	L1	*L2*	*L3*	*L4*	*L5*	*L6*	*L7*	*L8*	*L9*	*LA*	*LB*	*LC*	*LD*
10	92	94	92	94	94	95	-	-	-	-	95	-	93
20	94	6	94	95	-	93	-	-	-	-	-	-	97
10-10	97	6	93	96	96	96	-	-	-	-	97	-	-
10-20	95	93	95	93	92	94	-	-	-	-	95	92	92
20-20	96	61	92	94	91	95	-	-	-	-	-	89	96
10	99	86	97	98	96	98	-	-	-	-	98	98	-
20	100	72	98	98	98	98	-	-	-	-	97	99	97
10-10	98	86	98	99	97	98	-	-	-	-	-	92	-
10-20	97	86	99	98	96	99	-	-	-	-	99	97	-
20-20	95	14	99	98	98	97	-	-	-	-	99	-	-
10	95	-	95	94	94	93	-	-	-	-	94	92	92
20	98	-	93	92	92	93	-	-	-	-	93	93	94
10-10	92	92	91	91	93	93	-	-	-	-	92	89	-
10-20	94	93	91	90	91	92	-	-	-	-	92	94	94
20-20	94	7	92	92	93	93	-	-	-	-	93	93	-

AUC is a good overall measure for how a classifier performs. When the classifier is used however one should pick up the threshold thereby the tradeoff between the sensitivity and specificity. This is called the ROC operating point and its 'manual' optimal selection does depend on the subject of interest. In this study we determined the optimal operating point from probabilistic considerations [67] by the following geometrical procedure. A straight line calculated with $slope = \frac{N}{P}\frac{(p(P|N)-p(N|N))}{(p(N|P)-p(P|P))}$ crossing the point $ROC(0,1)$ we shifted down and to the right, until it intersected the ROC curve. P and N are the total instance counts in the positive (melanoma) and negative (dysplatic) class and expressions for p denote different probabilities for misclassifications, e.g. $p(N|P)$ denotes the probability of misclassifying a positive class as a negative class etc. In Figure 1 we present those optimal points from trainings with different $\mathcal{L}$earn and $\mathcal{T}$opol setups.

First we remark that the above numerical procedure fail for few points (middle field and right upper corner). All the other points are optimal operating points. The strings of 'Lx' stand for the best-performing back-propagation algorithms and the colors represent the structure of the hidden layers. There are no single winners and we should mind that all the points come from the best-converging networks. It

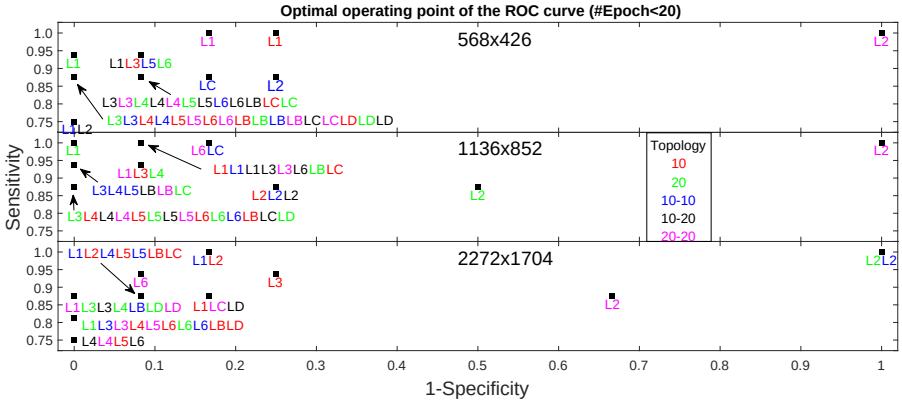


Figure 1. Optimal points (not components of one ROC) from trainings with different $\mathcal{L}$earn and $\mathcal{T}$opol setups. Resolutions A, B, C are presented.

seems that different combinations of learning algorithms and topologies reach upper-left ROC region. Taking into account possible system limitations of the developed applications (e.g. image resolution limits, the number of the weights, memory limits of the training cycle) one can optimally select a pair ($\mathcal{L}earn$,$\mathcal{T}opol$).

L1 (Levenberg-Marquardt) shows extreme robustness for almost all topologies for resolutions A, B and C.

L6 (Polak-Ribiére Conjugate Gradient) seems to be the second best for A,B,C - it is less represented but reaches high AUC values in the upper-left corner.

Fig.1 shows that the magnitude of the the operating points is preserved and stable for B and C and, unexpectedly, outperforms the initial A conditions. An extra attempt was done to degrade the resolution C by a factor of two, but it failed (performance decrease by 30%). A stable overall performance for A, B and C proves resolution invariance of the wavelet features for the original clinical data and the two descendant downgraded image resolutions. This behavior confirms results from [17] where the ensemble learning paradigm was implemented. In our setup a single 'standard' learner (ANN) performs in the same way. It seems that this performance is even better than in [17] where the downgraded resolutions showed still high but slightly worse results. In our experiments with ANN, which is a homogeneous (not complex) model, better sensitivity and specificity is reached for dermoscopy images with smaller resolutions. The maximum reached at B shows the best production resolution of 1136x852 pixels.

Whether this behavior is due to the wavelet features and not the (selected) learning methods it should be further studied both in some future experiments with other learning paradigms and by theoretical considerations about the wavelet families and their properties.

5. References

[1] Korotkov, K., Garcia, R., *Computerized analysis of pigmented skin lesions: A review.* Artificial Intelligence in Medicine, 2012, **56**(2).

[2] Masood, A., Ali Al-Jumaily, A., *Computer aided diagnostic support system for skin cancer: A review of techniques and algorithms.* International Journal of Biomedical Imaging, 2013, **2013**(7), pp. 323268.

[3] Oliveira, R.B., Papa, J.P., Pereira, A.S., Tavares, J.M.R., *Computational methods for pigmented skin lesion classification in images: Review and future trends.* Neural Computing and Applications, 2016.

[4] Skvara, H., Teban, L., Fiebiger, M., Binder, M., Kittler, H., *Limitations of dermoscopy in the recognition of melanoma.* Arch. Dermatol., 2005, **141**, pp. 155–160.

[5] Stolz, W., Semmelmayer, U., Johow, K., Burgdorf, W.H., *Principles of dermatoscopy of pigmented skin lesions.* Seminars in Cutaneous Medicine and Surgery, 2003, **22**(1), pp. 9–20.

[6] Wang, S.Q., Hashemi, P., *Noninvasive imaging technologies in the diagnosis of melanoma.* Seminars in Cutaneous Medicine and Surgery, 2010, **29**(3), pp. 174–184.

[7] Talbot, H., Bischof, L., *An overview of the polartechnics solarscan melanoma diagnosis algorithms*, 2003, pp. 33–38.

[8] Boone, M., Suppa, M., Dhaenens, F., Miyamoto, M., Marneffe, A., Jemec, G., Del Marmol, V., Nebosis, R., *In vivo assessment of optical properties of melanocytic skin lesions and differentiation of melanoma from non-malignant lesions by high-definition optical coherence tomography.* Arch. Dermatol. Res., 2016, **308**(1), pp. 7–20.

[9] Johr, R.H., *Dermatoscopy: Alternative melanocytic algorithms - the abcd rule of dermatoscopy, menzies scoring method, and 7-point checklist.* Clinics in Dermatology, 2002, **20**, pp. 240–247.

[10] Kittler, H., Pehamberger, H., Wolff, K., Binder, M., *Follow-up of melanocytic skin lesions with digital epiluminescence microscopy: Patterns of modifications observed in early melanoma, atypical nevi, and common nevi.* J. Am. Acad. Dermatol., 2000, **43**(3), pp. 467–476.

[11] Goodson, A.G., Grossman, D., *Strategies for early melanoma detection: Approaches to the patient with nevi.* J. Am. Acad. Dermatol., 2009, **60**(5), pp. 719–735.

[12] Chang, T., Kuo, C.C., *Texture analysis and classification with tree-structured wavelet transform.* IEEE Transactions on Image Processing, 1993, **2**(4), pp. 429–44.

[13] Walvick, R.P., Patel, K., Patwardhan, S.V., Dhawan, A.P., Classification of melanoma using wavelet-transform-based optimal feature set. In: *Medical Imaging 2004*, International Society for Optics and Photonics, 2004, pp. 944–951.

[14] Ma, L., Staunton, R.C., *Analysis of the contour structural irregularity of skin lesions using wavelet decomposition.* Pattern Recognition, 2013, **46**(1), pp. 98–106.

[15] Massone, C., Hofmann-Wellenhof, R., Ahlgrimm-Siess, V., Gabler, G., Ebner, C., Soyer, H.P., *Melanoma screening with cellular phones.* PLoS ONE, 2007, **2**(5), pp. e483.

[16] MacKinnon, N., Vasefi, F., Booth, N., Farkas, D.L., *Melanoma detection using smartphone and multimode hyperspectral imaging.* SPIE BiOS, 2016, **9711**, pp. 971117–1.

[17] Surówka, G., Ogorzałek, M., *On optimal wavelet bases for classification of melanoma images through ensemble learning.* Artificial Intelligence and Soft Computing, Lecture Notes in Computer Science,, 2016.

[18] Patwardhan, S.V., Dhawan, A.P., Relue, P.A., *Classification of melanoma using tree structured wavelet transforms.* Computer Methods and Programs in Biomedicine, 2003, **72**, pp. 223–239.

[19] Patwardhan, S.V., Dai, S., Dhawan, A.P., *Multi-spectral image analysis and classification of melanoma using fuzzy membership based partitions.* Computerized Medical Imaging and Graphics, 2005, **29**, pp. 287–296.

[20] Surówka, G., Merkwirth, C., Żabińska-Płazak, E., Graca, A., *Wavelet based classification of skin lesion images.* Bio Alg. Med Syst., 2006, **2**(4).

[21] Surówka, G., Grzesiak-Kopec, K., *Different learning paradigms for the classification of melanoid skin lesions using wavelets.* Proc. EMBC07 Lyon, 2007.

[22] Surówka, G., *Supervised learning of melanocytic skin lesion images.* Proc. HSI Kraków, 2008.

[23] Indira, D., Jyotsna Suprya, P., *Detection & analysis of skin cancer using wavelet techniques.* International Journal of Computer Science and Information Technologies, 2011, **2**(5), pp. 1927–1932.

[24] Fassihi, N., Shanbehzadeh, J., Sarrafzadeh, H., Ghasemi, E., *Melanoma diagnosis by the use of wavelet analysis based on morphological operators.* Proc. Int. Multiconf. Eng. Comp. Sci. I Hong-Kong, 2011.

[25] Castillejos, H., Ponomaryov, V., Nino-de Rivera, L., Golikov, V., *Wavelet transform fuzzy algorithms for dermoscopic image segmentation.* Computational and Mathematical Methods in Medicine, 2012, **578721**.

[26] Ramteke, N.S., Jain, S.V., *Analysis of skin cancer using fuzzy and wavelet technique - review & proposed new algorithm.* International Journal of Engineering Trends and Technology, 2013, **4**(6).

[27] Sugin, S., Jegadeesh, A., *Segmentation of skin images using fixed grid wavelet networks.* International Journal of Engineering Research & Technology, 2014, **3**(4).

[28] Rajarathinam, A., Arivazhagan, A., *Timely efficient automated system by segmentation using wavelet transform.* International Journal of Science, Engineering and Technology Research, 2015, **4**(8).

[29] Sikorski, J., *Identification of malignant melanoma by wavelet analysis.* Proceedings of Student/Faculty Research Day, CSIS, Pace University, 2004.

[30] Aswin, R., Jaleel, J.A., Salim, S., *Implementation of ann classifier using matlab for skin cancer detection.* ICMiC13, 2013, pp. 87–94.

[31] Mahmoud, M.K.A., Al-Jumaily, A., Takruri, M., *The automatic identification od melanoma by wavelet and curvelet analysis: Study based on neural network classification.* 11th International Conference on Hybrid Intelligent Systems, 2011, pp. 680–685.

[32] Clawson, K.M., Morrow, P., Scotney, B., McKenna, J., Dolan, O., *Analysis of pigmented skin lesion border irregularity using the harmonic wavelet transform.* Machine Vision and Image Processing Conf., 2009.

[33] Ercal, F., Chawla, A., Stoecker, W.V., Lee, H.C., Moss, R.H., *Neural network diagnosis of malignant melanoma from color images.* IEEE Trans. Biomed. Eng., 1994, **41**(9).

[34] Dreiseitl, S., Ohno-Machado, L., Kittler, H., Vinterbo, S., Billhardt, H., Binder, M., *A comparison of machine learning methods for the diagnosis of pigmented skin lesions.* Journal of Biomedical Informatics, 2001, **34**, pp. 28–36.

[35] Rubegni, P., Burroni, M., Perotti, R., Fimiani, M., Andreassi, L., Cevenini, G., Dell'Eva, G., Barbini, P., *Digital dermoscopy analysis and artificial neural network for the differentiation of clinically atypical pigmented skin lesions: A retrospective study.* J. Invest. Dermatol., 2002, **119**, pp. 471–474.

[36] Hoffmann, K., Gambichler, T., Rick, A., Kreutz, M., Anschuetz, M., Grünendick, T., Orlikov, A., Gehlen, S., Perotti, R., Andreassi, L., et al., *Diagnostic and neural analysis of skin cancer (danaos). a multicentre study for collection and computer-aided analysis of data from pigmented skin lesions using digital dermoscopy.* British Journal of Dermatology, 2003, **149**, pp. 801–809.

[37] Rajab, M., Woolfson, M., Morgan, S., *Application of region-based segmentation and neural network edge detection to skin lesions.* Computerized Medical Imaging and Graphics, 2004, **28**, pp. 61–68.

[38] Maglogiannis, I., Pavlopoulos, S., Koutsouris, D., *An integrated computer supported acquisition, handling, and characterization system for pigmented skin lesions in dermatological images.* IEEE Trans. Inf. Techn. Biomed., 2005, **9**(1).

[39] Zagrouba, E., Barhoumi, W., *An accelerated system for melanoma diagnosis based on subset feature selection.* Journal of Computing and Information Technology, 2005, **13**(1), pp. 69–82.

[40] Iyatomi, H., Oka, H., Celebi, M.E., Hashimoto, M., Hagiwara, M., Tanaka, M., Ogawa, K., *An improved internet-based melanoma screening system with dermatologist-like tumor area extraction algorithm.* Computerized Medical Imaging and Graphics, 2008, **32**(7), pp. 566–579.

[41] Schaefer, G., Rajab, M.I., Celebi, M.E., Iyatomi, H., *Skin lesion segmentation using cooperative neural network edge detection and colour normalization.* Inf. Techn. and Applic. Biomed., 2009.

[42] Lau, H.T., Al-Jumaily, A., *Automatically early detection of skin cancer: Study based on neural network classification.* IEEE International Conference of Soft Computing and Pattern Recognition, 2009, pp. 375–380.

[43] Vennila, G.S., Suresh, L.P., Shunmuganathan, K., *Dermoscopic image segmentation and classification using machine learning algorithms.* American Journal of Applied Sciences, 2012, **8**(11).

[44] Mhaske, H., Phalke, D., *Melanoma skin cancer detection and classification based on supervised and unsupervised learning.* International conference on Circuits Controls and Communications, 2013, pp. 1–5.

[45] Jaleel, J.A., Salim, S., Aswin, R., *Computer aided detection of skin cancer.* International Conference on Circuits, Power and Computing Technologies, 2013.

[46] Elgamal, M., *Automatic skin cancer images classification.* International Journal of Advanced Computer Science and Applications, 2013, **4**(3).

[47] Silva, C.S., Marcal, A.R., *Colour-based dermoscopy classification of cutaneous lesions: An alternative approach.* DOI: 10.1080/21681163.2013.803683, 2013.

[48] Achakanalli, S., Sadashivappa, G., *Skin cancer detection and diagnosis using image processing and implementation using neural networks and abcd parameters*, 2014.

[49] Alasadi, A.H., ALsafy, B.M., *Early detection and classification of melanoma skin cancer.* Int. J. Information Technology and Computer Science, 2015, **12**, pp. 67–74.

[50] Torre, E.L., Caputo, B., Tommasi, T., *Learning methods for melanoma recognition.* International Journal of Imaging Systems and Technology, 2010, **20**(4), pp. 316–322.

[51] Ruiz, D., Berenguer, V., Soriano, A., SáNchez, B., *A decision support system for the diagnosis of melanoma: A comparative approach.* Expert Systems with Applications, 2011, **38**, pp. 15217–15223.

[52] Maglogiannis, I., Kosmopoulos, D.I., *Computational vision systems for the detection of malignant melanoma.* Oncology Reports, 2006, **15**(Spec no. 4), pp. 1027–32.

[53] Rajpara, S., Botello, A., Townend, J., Ormerod, A., *Systematic review of dermoscopy and digital dermoscopy/artificial intelligence for the diagnosis of melanoma.* British Journal of Dermatology, 2009, **161**(3), pp. 591–604.

[54] Sathiya, S.B., Kumar, S., Prabin, A., *A survey on recent computer-aided diagnosis of melanoma.* International Conference on Control Instrumentation Communication and Computational Technologies, 2014, pp. 1387–1392.

[55] Abedini, M., Chen, Q., Codella, N.C., Garnavi, R., Sun, X., *Accurate and scalable system for automatic detection of malignant melanoma.* In book: Dermoscopy Image Analysis, 2015, pp. 293–343.

[56] Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P., *Smote: Synthetic minority over-sampling technique.* Journal of Artificial Intelligence Research, 2002, **16**, pp. 321357.

[57] Stefanowski, J., Wilk, S., *Selective pre-processing of imbalanced data for improving classification performance.* Data Warehousing and Knowledge Discovery, 2008, pp. 283–292.

[58] Rastgoo, M., Lemaitre, G., Massich, J., Morel, O., Marzani, F., Garcia, R., Meriaudeau, F., *Tackling the problem of data imbalancing for melanoma classification.* BIOSTEC - 3rd International Conference on BIOIMAGING, 2016.

[59] Mallat, S.G., *A theory for multiresolution signal decomposition: The wavelet representation.* IEEE Transactions on pattern analysis and machine intelligence, 1989, **11**(7).

[60] Daubechies, I., *Ten lectures on wavelets.* CBMS SIAM, 1994, **61**.

[61] Tang, J., Alelyani, S., Liu, H., *Feature selection for classification: A review.* CRC Press, 2014, **37**.

[62] Maglogiannis, I., Doukas, C.N., *Overview of advanced computer vision systems for skin lesions characterization.* IEEE Trans. Inf. Techn. Biomed., 2009, **13**(5), pp. 721–733.

[63] Michie, D., Spiegelhalter, D.J., Taylor, C.C., *Machine Learning, Neural and Statistical Classification.* Prentice Hall, 1994.

[64] Haykin, S., *Neural Networks: A Comprehensive Foundation.* 2 edn. Prentice Hall, 2004 ISBN 0-13-273350-1.

[65] Demuth, H.B., Beale, M.H., De Jess, O., Hagan, M.T., *Neural Network Design.* 2 edn., 2004 ISBN-10: 0-9717321-1-6, ISBN-13: 978-0-9717321-1-7.

[66] Battiti, R., *First- and second-order methods for learning: Between steepest descent and newton's method.* Neural Computation, 1992, **4**(2).

[67] Hajian-Tilaki, K., *Receiver operating characteristic (roc) curve analysis for medical diagnostic test evaluation.* Caspian J. Intern. Med., 2013, **4**(2), pp. 627–635.