Multi-topic Text Categorization Based on Ranking Approach

Valentina V. Glazkova, Mikhail I. Petrovskiy,

Computer Science Department of Lomonosov Moscow State University

***Abstract* — This paper is devoted to the multi-topic (multi-label) text classification problem. We propose two methods for reduction from ranking to the multi-label case. Unlike existing multi-label classification methods based on reduction from ranking problem, where the complex classification (threshold) function is being defined on the input feature space, in our approach we propose the construction of simple (linear) multi-label classification function using the output of the ranking methods (class relevance space) as the input. In our first method we propose to estimate the linear threshold function defined on the class relevance space. In the second method we directly find the linear operator mapping class ranks into the set of values of binary multi-label decision functions. Developed methods are less computationally expensive than existing methods and in the same time our methods demonstrate similar and in some cases significantly better accuracy. That has been demonstrated experimentally on well-known multi-label benchmark dataset Reuters-2000 (multi-topic text articles).**

***Index Terms* — Data Mining, machine learning, multi-label classification, ranking methods, text categorization.**

I. Introduction

In traditional multi-class classification problems the classes are mutually exclusive, that is required by definition. The multi-label classification problem is an extension of traditional classification problem, where each sample may belong to several classes simultaneously. This type of classification problems arises in many important applications, such as text categorization [1,2,3,4,5], image recognition [9], bioinformatics [6,7,8] and many others. In these applications any sample may have several different labels. It leads to the problem of essentially overlapped classes. Traditional multi-class learning algorithms cannot deal with it directly. Under this problem statement, the classifier, trained on multi-label data, must predict all relevant classes for a given sample. Existing multi-label classification methods can be divided into three basic groups: the methods based on decomposition of multi-label problem into multiple independent binary classification subproblems [1,9]; the methods based on direct posteriori class set probability estimation [3,8]; the methods based on reduction from ranking problem to multi-label case [1,2,6,7,10].

In the first group "one-against-the-rest" approach is used. For each class separate binary classification subproblem is formulated. In this subproblem all samples from the multi-label training set are divided into two disjoint subsets: "positive" samples, whose belong to the specified class, and "negative" samples that do not belong to this class. Then traditional binary learning algorithm is applied to each binary subproblem. As a result, a set of independent binary classifiers are trained. Each classifier is associated with specific class and predicts whether a given sample belongs to this particular class. Though, such methods are very easy to implement using of-the-shelf binary classification algorithms, they are strongly criticized for two major weaknesses. The first weakness is that such binary classifiers are trained independently, and, thus they do not take into account the correlation between classes [1,3,6]. The second weakness is the performance issue. It is obvious that if we have a multi-label problem of size m and q classes, we must solve q problems of size m, and, roughly speaking, it means to perform in q times more calculations.

The second group of multi-label classification methods are the ones based on a posteriori class set probability estimations [3,8]. Unlike methods from previous group, they are the "natural multi-label" methods and provide direct search and estimation of the most probable class set (combination of relevant classes). Though, these methods are the most precise and the most mathematically correct for the multi-label classification problem, they do have big performance issue. They require exploring of all possible combinations of classes. It means that their complexity grows exponentially with the number of classes that is unacceptable for the majority of practical applications.

The third group includes multi-label classification methods based on reduction from ranking to multi-label problem. Unlike in the multi-label classification problem statement the task of ranking method is not to separate relevant classes from irrelevant, but estimate relevance of each class and position

Manuscript received April 10, 2007. This work was supported in part by RFFI Grants No. 05-01-00744, 06-01-00691 and 06-07-08035-ofi, Leading Scientific Scool Support Grant No. 02.445.11.7427, Russian Federation President Grant MK-4264.2007.9.

V. V. Glazkova is post-graduate student with Computer Science Department of Lomonosov Moscow State University, Building 2, MSU, Vorobjovy Gory, Moscow, 119992, Russia Phone: +7 495 9391789, Fax: +7 495 9391988 (e-mail: glazv@mail.ru).

M. I. Petrovskiy is associated professor with Computer Science Department of Lomonosov Moscow State University, Building 2, MSU, Vorobjovy Gory, Moscow, 119992, Russia Phone: +7 495 9391789, Fax: +7 495 9391988 (e-mail: michael@cs.msu.su).

classes on real or ordinal scale in the order of growing relevance. Though it sounds like similar approach as from the first group (set of binary classifiers) really it is significantly different. The key moment is that ranking algorithms estimate class ranks jointly, i.e. they solve the single common problem, not the set of independent subproblems. It means that ranking algorithms naturally take into account the correlation between classes. Another difference is that ranking algorithms minimize ranking loss [8] or some other "confusion" errors. It means that they try to estimate the perfect order of classes, but not the absolute relevance of the classes. Thus, for ranking and multi-label classification the aims are different, the definition of minimized errors and losses are different, the objective functions in optimization procedures are different. But the proposition is that ranking algorithms can be used for further solving multi-label classification problem. For these purposes it is necessary to construct a predictor or decision function that separates relevant classes from irrelevant using output of the ranking algorithm. As a rule, threshold functions are used. The problem is that this function depends on the sample. Usually we cannot use the same threshold for all samples. It means that in addition to training ranking predictor we need to estimate a threshold function. A. Elisseeff and J. Weston proposed to find threshold function, defined on the input feature space [6,7]. However, in many practical applications, for example, in text classification, this approach does not work well, because the input feature space dimensionality is too high (e.g. the size of dictionary might be hundreds of thousands). It means that time complexity of threshold function estimation and space required for storing function coefficients are too large. Besides, the precision of such functions is usually not very good.

To avoid these problems we propose two new methods of reduction from ranking to multi-label classification. These methods are united by the same key idea. It is constructing the decision function for separating relevant classes from irrelevant not on the input feature space, but on the space of class relevance, i.e. the space of vectors, whose coordinates are values of the class ranks, produced by the ranking algorithm. In addition to solving the problem of time and space complexity we assume that in the space of class relevance the structure of the threshold function or multi-label decision function might be simple enough (in contrast to the input feature vector space [6,7]). We propose to find these functions in a family of linear functions. The experimental performance evaluation has proved that such linear methods demonstrate very competitive accuracy in comparison to existing state-of-the-art more complicated methods and even outperform many of them.

This paper is organized as follows. Section II is devoted to related work and brief overview of existing ranking methods and methods for reduction from ranking to multi-label. Section III presents our new methods. Section IV reports experimental performance estimation results on well-known multi-label benchmark dataset Reuters-2000 [11]. Finally, conclusions are presented in section V.

II. Related Work

A. Ranking Problem Setting

Let us start with formal *ranking problem* statement in machine learning. Assume that the *input feature space* is *n*-dimensional real space $X = \Re^n$ and the set of all possible classes is denoted by Y, where the number of different classes is $q = |Y|$. Given a multi-label training set:

$$S = \{(\vec{x}_i, y_i) \mid 1 \le i \le m, \vec{x}_i \in \Re^n, y_i \subset Y\}, \qquad (1)$$

where each *sample* $\vec{x} \in \Re^n$ is associated with subset of Y, the *subset of relevant classes*. The task of ranking learning algorithm is to analyze the training set *S* and to produce a ranking function $r_S : X \times Y \to \Re$, such that for any given sample $\vec{x}$ it orders classes from Y according to their *relevance* to the sample. Class *l* is more relevant than class *k* if $r(\vec{x}, l) > r(\vec{x}, k)$. The majority of existing ranking methods lead to solving the optimization problems that are based of "confusion" (disorder) loss (or error) minimization, such as *Ranking Loss* [4,5,8]:

$$RankingLoss_S(r) = \sum_{(\vec{x}_i, y_i) \in S} \frac{1}{|y_i||\overline{y}_i|} \left| \left\{ (l, s) \in y_i \times \overline{y}_i : r(\vec{x}_i, l) \le r(\vec{x}_i, s) \right\} \right| \qquad (2)$$

The ideal ranking algorithm produces the ranking function that always predicts the *perfect order*, i.e. the order where all relevant classes are positioned higher than any irrelevant, but the order among relevant (and among irrelevant) is insufficient. Usually no specific constraints are set on the domain or scale of relevance values. In general case it is considered as $\Re$.

B. Existing Ranking Methods

In this paper it is impossible to overview all existing approaches and algorithms for ranking, because currently this is an area of very intensive research and the number of new algorithms grows constantly. But we have to mention the most popular ranking algorithms of the following types: neural network algorithms (Multiclass-Multilabel Perceptron and combination of binary Perceptrons) [4,5], instance-based learning methods (*k*-nearest neighbors for ranking [12]), support vectors machines (RankSVM [6,7]), adaptive boosting (AdaBoost.MH) [1,2] and boosted decision trees (ADTBoost.MH) [10].

1) Multiclass-Multilabel Perceptron (MMP) Algorithm

MMP algorithm [4,5] is an *online* algorithm: it gets training object, outputs ranking, then obtains the accurate set of relevant classes and calculates corresponding ranking error. If the error is zero, the algorithm gets the next training object; otherwise, the algorithm updates a hypothesis it maintains - that is a set of prototypes $\vec{w}_1, \vec{w}_2, ..., \vec{w}_q, \vec{w}_r \in \Re^n$, $r = \overline{1, q}$. It is repeated for all objects from training set (1).

The class ranking is induced by the set of prototypes

according to their similarity with the vector representation of the object. That is, given an object $\vec{x}$ the inner products $\vec{w}_1 \cdot \vec{x}, \vec{w}_2 \cdot \vec{x}, ..., \vec{w}_q \cdot \vec{x}$ induce an ordering the relevance level for each class. It is considered that the class *r* is *ranked higher* than class *s* if $\vec{w}_r \cdot \vec{x} > \vec{w}_s \cdot \vec{x}$.

Algorithm MMP uses error set notion that plays a major role in iterative updating of prototypes. *The error set* for $(\vec{x}_i, y_i)$ is defined as the set of all pairs (r, s) whose ordering according to predicted ranking disagrees with the set y_i of relevant classes for $\vec{x}_i$ (that is class *r* is ranked not-above class *s*, but class *r* is one of relevant classes while class *s* is not relevant):

$$E_i = \left\{(l, s) \in y_i \times (Y \ / \ y_i) : \vec{w}_l^i \cdot \vec{x}_i \le \vec{w}_s^i \cdot \vec{x}_i \right\},$$

Algorithm MMP moves each prototype representing a relevant class in E_i toward object $\vec{x}_i$ and, analogously, each prototype representing a non-relevant class in E_i away from $\vec{x}_i$. The purpose of prototype update is to increase the value of inner products between $\vec{x}_i$ and each of the prototypes in the subset of relevant classes in error set E_i and to decrease the inner product values between $\vec{x}_i$ and the non-relevant classes in E_i.

2) K-Nearest Neighbors Method

The k-nearest neighbors (k-NN) method [12] is an example of instance-based classification strategy. Unlike others, this method not requires training phase. The classifier simply "remembers" feature vectors of all objects in the training set. To rank categories of new object $\vec{x}_{test}$, this object is compared to each object $\vec{x}$ in the training set, that is, a distance between $\vec{x}_{test}$ and $\vec{x}$ is calculated. This distance is expressed by cosine of angle between these feature vectors:

$$\rho(\vec{x}_{test}, \vec{x}) = 1 - \cos(\vec{x}_{test}, \vec{x}).$$

Then the classifier selects *k* nearest (more similar with $\vec{x}_{test}$) objects based on the calculated distances $\vec{x}_{test}$ (*k* is adjustable parameter). For each class $l \in Y$, its *relevance* is calculated (the sum of similarity values of the selected nearest objects with label *l*):

$$r(\vec{x}_{test}, l) = \sum_{\vec{x} \in Near_k(\vec{x}_{test}) \wedge l \in y_{\vec{x}}} \cos(\vec{x}_{test}, \vec{x}),$$

where $Near_k(\vec{x}_{test})$ is the set of the *k* nearest neighbors in the training set for object $\vec{x}_{test}$, Y is the set of possible classes, $y_{\vec{x}} \subset Y$ is the relevant class set for object $\vec{x}$. K-nearest neighbours method drawbacks are the following: large classification time and large memory space that necessary for holding and handling all training set.

3) Kernel ranking method RankSVM

The idea of RankSVM method [6,7] is to find ranking function $r(\vec{x}, l) = \vec{w}_l \cdot \vec{x} + b_l$ that minimizes Ranking Loss:

$$Err_S(r) = \sum_{(\vec{x}_i, y_i) \in S} \frac{1}{|\ y_i\ ||\ \overline{y}_i\ |} \left| \left\{ (l, s) \in y_i \times \overline{y}_i : r(\vec{x}_i, l) \le r(\vec{x}_i, s) \right\} \right|,$$

while having a large margin for multi-label model. Maximizing the margin on the whole training set can be done via following problem:

$$\max_{\vec{w}_k, k=1,...,q} \ \min_{(\vec{x}, y) \in S} \ \min_{l \in y, s \in \overline{y}} \frac{1}{\left\| \vec{w}_l - \vec{w}_s \right\|^2},$$

subject to $(\vec{w}_l - \vec{w}_s) \cdot \vec{x} + b_l - b_s \ge 1, \ \ (l, s) \in y \times \overline{y}$.

One of the major drawbacks of RankSVM method is the space complexity. This method requires in worst case an amount of memory proportional to mq^2 (where *m* is number of training objects, *q* is number of classes). This amount of memory can become unaffordable when *q* is large, for example, for text classification problems.

4) Ranking methods based on boosting algorithms

The purpose of boosting algorithms is to find a highly accurate classification rule by combining many weak hypotheses, each of which may be only moderately accurate. Boosting algorithm combines hypotheses generated from the same learning algorithm on different subsets of training examples. Therefore in each round *t*, where a weak hypothesis $h_t(\vec{x}, l)$ is generated, an appropriate subset of training examples has to be chosen and finally all calculated weak hypotheses have to be combined into a single final ranking hypothesis:

$$r(\vec{x}, l) = \sum_{t=1}^{T} h_t(\vec{x}, l),$$

where *T* is the number of boosting rounds.

In the training phase, AdaBoost.MH [1,2] algorithm maintains a set of weights over both training examples and their classes, where training examples and their corresponding classes that are hard (easy) to predict correctly get incrementally higher (lower) weights. These weights are used by the weak learning algorithm whose goal is to find a weak hypothesis with moderately low error with respect to these weights. However this method requires the large number of steps in the training phase for good accuracy, and therefore, requires long training time (especially for text classification problems).

ADTBoost.MH ranking method [10] is one more method based on boosting-algorithms. Motivation of this method is readability of classification models with using of Alternating Decision Trees (ADTrees, [13]). Alternating decision trees contain splitter nodes and prediction nodes. A splitter node is associated with a test (comparisons attribute values with specific constants); a prediction node is associated with a real value. Each object defines a set of paths in an ADTree. The

sum of the predictions along the paths defines relevance of predicted classes. ADTBoost.MH method [10] is an extention of ADTBoost [13] for multi-label classification problem. A prediction node is now assosiated with a set of real values, one for each class. In ADTBoost.MH method, the number of the classification rules in a multi-label ADTree is related to the boosting rounds.

5) Ranking methods based on decomposition of multi-label learning problem into multiple independent binary classification problems

Let us consider how decision function of binary classification can be applied to multi-label ranking problem. This approach is based on the reduction of multi-label problem into multiple binary problems ("one-against-rest" approach): one binary problem is created for each of q classes. In a binary problem for class l all training objects associated with this class are regarded as positive examples, and all other training objects are regarded as negative. Then binary learning algorithm (for example, binary SVM [9] or Perceptron algorithm [4,5]) is applied to each of q obtained binary problems. As the result, for each class, decision function $f_l(\vec{x}),\ l = \overline{1,q}$ of binary classification will be obtained. This decision function can be used as ranking function for class l: $r_l(\vec{x}) = f_l(\vec{x})$. For example, for Perceptron algorithm, ranking function can be written in the form $\vec{r}(\vec{x}) = (r_1(\vec{x}),...,r_q(\vec{x})) = (\vec{w}_1 \cdot \vec{x},...,\vec{w}_q \cdot \vec{x})$, where each of prototypes $\vec{w}_l,\ l = \overline{1,q}$ is an output of corresponding Perceptron algorithm.

C. Reduction from ranking to multi-label classification

Let us now briefly introduce the existing approach for reduction from ranking to multi-label classification. The following methods have been proposed for it: *cutting off by constant threshold* (usually by zero) [1,2,10] and *Threshold prediction function defined on the Input Feature Space* (TIFS) [6,7]. For the case of cutting off by constant threshold the solution of multi-label classification function $F: X \to 2^q$ can be presented in the form: $F(\vec{x}) = \{l \in Y \mid r(\vec{x}, l) > 0\}$. This approach was applied in AdaBoost.MH [1,2] and ADTBoost.MH [10] methods. However, it has demonstrated low accuracy in other methods.

A. Elisseeff and J. Weston have proposed to find a threshold function $t_S: X \to \Re$ in the input feature vector space [6,7]. Having ranking function $\vec{r}(\vec{x})$, it is possible to calculate values of ranks $\vec{r}(\vec{x}_i) = (r_1(\vec{x}_i),...,r_q(\vec{x}_i))$ for each training sample $\vec{x}_i$. Then they proposed an approach based on estimation of the threshold functions in some predefined functional family Ψ. The solution can be found as a function that minimize the following error for all samples from the training set [6,7]:

$$t(\vec{x}_i) = \arg\min_{t(\vec{x}_i)\in\Psi}\left[\left|\{l \in y_i : r_l(\vec{x}_i) \le t(\vec{x}_i)\}\right| + \left|\{l \in \bar{y}_i : r_l(\vec{x}_i) > t(\vec{x}_i)\}\right|\right]$$

However, the complexity of learning procedure and application of such threshold function depend on the dimensionality of the input feature space, and as a result the threshold estimation complexity is of the same order of complexity as the ranking algorithm itself. Thus, we should do double work: predict ranks and then estimate threshold on the input feature space that requires nearly the same amount of calculations.

III. Our Approach to Reduction from Ranking to Multi-label

At first let us consider the formal multi-label classification problem statement: given a multi-label training set (1), the goal of *multi-label classification* learning algorithm is to construct the decision function $F: X \to 2^q$ that for any given sample $\vec{x}$ determines its relevant classes. The prediction quality is measured by accuracy of this class set determination, and the order of all classes is insufficient (in opposite to ranking problem). In fact, most multi-label methods are based on Hamming Loss minimization:

$$HL(F(x), y) = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{|Y|}\left|\{F(\vec{x})\} \nabla \{y_i\}\right|, \qquad (3)$$

It is a standard precision measure for multi-label classification, where $a \nabla b = (a \cup b) \setminus (a \cap b)$, is a symmetric set difference; $F(\vec{x})$ is multi-label classification function; y_i is the set of relevant classes for sample $\vec{x}_i$.

The process of solving multi-label classification problem with ranking method can be decomposed into two stages. The first stage is construction of the predictor that ranks classes from *Y* according to their relevance for any given sample. The second stage is construction of the multi-label classification function to separate on the ranked class scale the relevant classes from irrelevant. In this paper we propose two new methods for reduction from ranking to multi-label classification problem. They are denoted as TCRS (Threshold function on Class Relevance Space) and LORM (Linear Operator from class Ranks to Multi-label decision function). The key idea of these methods is in *constructing multi-label classification functions not on the input feature space, but on class relevance vector space*, i.e. the space of vectors, which coordinates are values of the class ranks. In other words, we first estimate ranks for a given sample and than consider these ranks as a new features of the sample to construct multi-label decision function. The motivation of this approach is based on two postulates:

1. Usually, *the dimensionality of class relevance space is several orders smaller than dimensionality of the input feature space*, because the dimensionality of class relevance space is equal to the number of classes.
2. Since for any given sample the predicted ranks are usually already the result of some nonlinear transformation from

the input space to ranks, we assume *that dependency between class ranks and multi-label decision function must be very simple*. In this paper we consider the family of *linear functions*.

These formulated assumptions allow us to greatly reduce the computational efforts to construct the threshold or decision function for multi-label classification using outputs of ranking algorithm. In that way, the aim of the presented research is to check the hypothesis that the presented postulates are true. So, we formulate two different linear methods based on postulates 1 and 2, propose solving algorithms for them and experimentally estimate the accuracy of the presented methods in comparison to existing state-of-the-art methods on well-known multi-label benchmark dataset.

The first our method is a modification of the threshold method [6,7]. It constructs the threshold function which estimates the threshold value depending on the class relevance vectors. Thus, it maps class ranks into value of the threshold on the same scale (scale of ranks). It minimizes the error, which is the number of incorrectly classified relevant and irrelevant classes (similar to Hamming Loss [8]). The second our method is based on construction of several binary decision functions that map class relevance vectors into {-1,1} set for each class. "-1" means that this class is irrelevant, "1" means that the class is relevant. This method is based on minimization of mean-square error. It is important to note that unlike binary methods, defined in the input feature space (see Introduction), our second method does take into account the class correlations, since it uses as input the class ranks, which are correlated by definition of ranking problem solution. Both our methods find decision functions in a family of linear functions and use least squares optimization method.

A. Threshold Method in the Class Relevance Space

This method is a an improvement of the threshold method [6,7]. In our version of this method we propose to find threshold function defined not on *n*-dimensional input feature space, but on *q*-dimensional class relevance vector space ($q<<n$). In this method we assume that ranking algorithm has produced the perfect (or nearly perfect) order of class ranks. It means that class l is relevant for $\vec{x}$ iff its rank $r_l(\vec{x})$ is among the top $s(\vec{x})$ elements of $(r_1(\vec{x}), r_2(\vec{x}), ..., r_q(\vec{x}))$. In that way, the task is to estimate the size $s(\vec{x})$ of top ranked classes, which we assume to be relevant. For each training sample $\vec{x}_i$, we find threshold value t_i, which depends on ranks $\vec{r}(\vec{x}_i) = (r_1(\vec{x}_i),...,r_q(\vec{x}_i)) \in \Re^q$, predicted by some ranking algorithm. The threshold function is being found in a family of linear functions, defined on class relevance vector space:

$$t(\vec{r}(x)) = \sum_{j=1}^{q} a_j r_j(x) + a_0 , \qquad (4)$$

As in [6,7] the threshold function is estimated as whose that minimizes the number of incorrectly classified relevant and irrelevant classes for each training sample $\vec{x}_i$:

$$t_i = \arg\min_t \left(\left| \{ l \in y_i \mid r_l(\vec{x}_i) \le t \} \right| + \left| \{ l \in \bar{y}_i \mid r_l(\vec{x}_i) > t \} \right| \right). \qquad (5)$$

Taking sum of (5) over all training samples and substituting (4) in (5) we obtain the following optimization problem:

$$\min_{\vec{a}} \sum_{i=1}^{m} \left[t_i - \left(\sum_{j=1}^{q} a_j r_j(\vec{x}_i) + a_0 \right) \right]^2 . \qquad (6)$$

To solve (6) we apply least squares method. To predict the set of relevant classes for new given sample $\vec{x}_{test}$ using (4) we need to do the following. At first we calculate ranks $\overrightarrow{r_{test}} = \vec{r}(\vec{x}_{test}) = (r_1(\vec{x}_{test}),...,r_q(\vec{x}_{test}))$; then they are substituted in threshold function (4): $t_{test} = a_0 + a_1 r_1(\vec{x}_{test}) + ... + a_q r_q(\vec{x}_{test})$. And finally, the size of the relevant class set is estimated to the number of classes, for which the value of ranking function is above the threshold $s(\vec{x}_{test}) = \left| \{ l \mid r_l(\vec{x}_{test}) > t_{test}, l \in Y \} \right|$.

B. Linear Operator in the Class Relevance Space

In this section we present the approach different from threshold estimation, presented in previous section. It is based on the notation of *multi-label decision functions*, defined for each class separately. We denote the vector function $\vec{f}(\vec{x}) = \left(f_1(\vec{x}),..., f_q(\vec{x}) \right)$ as a *decision function*, where $f_l(\vec{x}),\ l = \overline{1,q}$ are binary decision functions for each class l. The sign of $f_l(\vec{x})$ gives a prediction whether a class l is relevant for a given sample $\vec{x}$. We propose the method for finding decision functions, based on ranking function in assumption of existing of linear operator mapping from $\vec{r}(\vec{x})$ to $\vec{f}(\vec{x})$. Thus, this approach is similar to the previous one in the sense that it estimates the multi-label decision functions in a family of linear functions defined on class relevance vector space. But the decision functions are different from threshold approach. The nice advantage is that unlike thresholds, this type of multi-label decision functions do not assume that the ranking class order (produced by ranking method) is perfect or really close to perfect. As in previous method we assume that class ranking problem for a given sample *x* has been already solved by some ranking algorithm and we have obtained the class ranks: $\vec{r}(\vec{x}) = (r_1(\vec{x}),...,r_q(\vec{x}))$. In this method we assume that decision functions linearly depends on ranking function:

$$\vec{f}(\vec{x}) = \tilde{A}\vec{r}(\vec{x}) + \tilde{b} . \qquad (7)$$

Adding to a ranking vector a fictitious component $r_0(\vec{x}) \equiv 1$ we can reformulate the problem (7) in homogeneous form:

$$\vec{f}(\vec{x}) = A\vec{r}(\vec{x}) , \qquad (8)$$

where $\vec{r}(\vec{x}) \equiv (r_0(\vec{x}), r_1(\vec{x}),..., r_q(\vec{x}))$; A is unknown linear operator (represented by $q \times (q+1)$ matrix). Let us define $f_l(\vec{x}_i)$ to be equal to 1 if the class l is relevant for sample $\vec{x}_i$

(i.e. $l \in y_i$) and –1, otherwise. Using ranking function $\vec{r}(\vec{x})$ we can find values $\vec{r}(\vec{x}_i) = (r_0(\vec{x}_i), r_1(\vec{x}_i), ..., r_q(\vec{x}_i))$ for each $\vec{x}_i$ in training set *S*. Then for each training sample $\vec{x}_i$, $i = \overline{1,m}$ the system (8) can be represented as $\vec{f}(\vec{x}_i) = A\vec{r}(x_i)$. Thus, we obtain *m* (the size of the training set) linear systems with the same $q \times (q+1)$ unknown matrix *A*. On the other hand, for each of training sample $\vec{x}_i$, $i = \overline{1,m}$, $r_l(\vec{x}_i)$ and $f_l(\vec{x}_i)$ are known and for each class *l* it is possible to estimate *l-th* row $\vec{a}_l$ of *A* matrix minimizing the total mean-square error:

$$\min_{a_{lj},\, j=0,q} \sum_{i=1}^{m} \left(f_l(\vec{x}_i) - \sum_{j=0}^{q} a_{lj} r_j(\vec{x}_i) \right)^2 . \quad (9)$$

Solving (9) for all q classes using the least squares optimization procedure, we obtain the matrix of unknown linear operator *A*, which defines the mapping between class ranks and our multi-label decision functions.

To classify new sample $\vec{x}_{test}$, we need to predict class ranks $\vec{r}(\vec{x}_{test}) = (1, r_1(\vec{x}_{test}), ..., r_q(\vec{x}_{test}))$ and calculate the values of q decision functions (8) according to the formula:

$$f_l(\vec{x}_{test}) = \vec{a}_l \vec{r}(\vec{x}_{test}) = a_{l0} + a_{l1} r_1(\vec{x}_{test}) + ... + a_{lq} r_q(\vec{x}_{test}) . \quad (10)$$

If $f_l(\vec{x}_{test}) \geq 0$, we consider that the class *l* is relevant for $\vec{x}_{test}$; and irreleveant otherwise. Thus, the decision about relevance of each class is taken separately, though, the correlation between classes is considered, since the decision functions are functions from ranks, which are correlated by definition.

IV. Experiments

We evaluate the performance of all developed methods on Reuters-2000 (multi-topic text articles) dataset that is the most popular benchmark dataset for multi-label classification [11]. The characteristics of this dataset are shown in table 1. For testing methods two subsets of Reuters-2000 dataset [11] were used (case 1: 3000 training documents and 3000 test documents; case 2: 15000 and 15000).

TABLE I
Characteristics of benchmark dataset

Data set		Number of classes	Number of features	Avg. classes per instance	Training set size	Testing set size
Reuters-2000 [12]	Case 1	101	47236	3.0	3000	3000
	Case 2				15000	15000

A. Experimental Setup

Our methods TCRS (*t=ar*) and LORM (*f=Ar*) for reduction from ranking to multi-label have been evaluated in combination with the following popular ranking methods: Multiclass-Multilabel Perceptron (MMP) [4]; binary Perceptron (bPerc) (the ranking method based on decomposition of multi-label problem into multiple binary classification problems, in which Perceptron algorithm was applied as binary classification method) [5]; simple ranking *k*-Nearest Neighbors (kNN) [12]. For each selected ranking method we run series of experiments with different reduction method: ">0" constant zero threshold (it is not applicable to kNN ranking [12]); "TIFS" Elisseeff and Weston [6,7] threshold function in the input space; "TCRS" our threshold function on the class relevance space (Section 3.1); LORM our linear operator based method (Section 3.2.). Each group of experiments corresponds to the same ranking method, but different reduction methods. Additionally, for demonstration purposes we have included two multi-label methods, based on other approaches (not on reduction from ranking). They are "binary decomposition". AdaBoost.MH method [1,7] and ML-KNN [8] method for posteriori class set probability estimation using kNN model. The ML-KNN method is one of the most precise, but very computationally expensive. For each ranking method the parameters (K for nearest neighbors and number of iterations T) are estimated using cross-validation.

TABLE II
Hamming Loss and Ranking Loss for multi-label classification methods on Reuters-2000 dataset

Method	Reuters-2000 (3000/3000)		Reuters-2000 (15000/15000)	
	Rank. Loss	**Hamming Loss**	Rank. Loss	**Hamming Loss**
AdaBoost.MH [7]	0.0268	0.0160	0.0169	0.0146
ML-kNN [8]	0.0194	**0.0144**	0.0167	0.0135
MMP+(>0)	0.0241	0.2942	0.0137	0.3031
MMP+TIFS		0.0157		**0.0126**
MMP+TCRS (t=ar)		0.0155		**0.0126**
MMP+LORM (f=Ar)		**0.0149**		0.0129
bPerc+(>0)	0.0210	0.0705	0.0099	0.0337
bPerc+TIFS		0.0162		0.0128
bPerc+TCRS (t=ar)		0.0158		**0.0127**
bPers+LORM (f=Ar)		**0.0146**		0.0130
kNN+TIFS	0.0202	0.0146	0.0167	0.0131
kNN+TCRS (t=ar)		**0.0141**		**0.0130**
kNN+LORM (f=Ar)		0.0144		0.0137

B. Experimental Results

Results of ranking and multi-label classification methods are presented in the table 2. Our methods names are printed in bold. The best results in each group are marked with bold underlined font. Each group includes different reduction methods applied after the same ranking method. The evaluation measures presented in resulting tables are Ranking Loss and Hamming Loss. Ranking Loss shows the quality of ranking and does not depend on reduction method. Thus, only

Hamming Loss measure shows the real quality of multi-label prediction. Ranking Loss is included in the resulting table just to demonstrate how good is the class order, produced by ranking algorithm (before reduction). Experimental results show that our linear methods for reduction from ranking to multi-label classification outperform their main competitors in each group. Thus, on these benchmark dataset they are more precise than constant threshold and threshold function on the input space methods. Besides, they demonstrate the accuracy comparable to one of the most precise probabilistic multi-label method ML-KNN and completely outperform popular AdaBoost.ML method, based on binary decomposition approach. Another interesting thing is that our linear operator based method ("f=Ar") can significantly improve the accuracy of multi-label solution in the case when the produced ranking order is not perfect. These are very promising results.

V. CONCLUSION

One of the approaches to solve multi-label classification problem is based on reduction from ranking to multi-label classification. In this approach we first solve ranking problem applying some of-the-shelf ranking algorithm to given multi-label training set, then obtain class ranks, and finally, separate the relevant classes from irrelevant, assuming that the class order is perfect. Traditionally the threshold method is used for these purposes. In this method the threshold function is defined on the input feature space. It leads to computationally expensive constructing of the predictor, which is usually imprecise. In this paper we propose the improvement of this approach. The key idea consists in training multi-label decision functions on class relevance vector space, i.e. on the output of ranking algorithms, not on the input feature space data. It allows to greatly reduce the amount of calculations, because the dimensionality of the class relevance space is usually in orders smaller than the dimensionality of the input space. Besides, since the class ranks are usually already produced by some nonlinear ranking method we can assume that dependence between estimated class ranks and multi-label decision functions may be simple enough. In this paper we consider linear these decision functions.

Following this idea in our first method we have extended standard threshold method and proposed to define a threshold function on a class relevancies vector space. It leads to the optimization problem of minimization the error similar to Hamming Loss. The second our method is based on construction of the linear operator, which maps ranking results directly to the values of multi-label vector decision function. According to this method, the decision about the relevance of each class is taken separately, but, unlike standard binary decomposition, our method takes into account correlations between classes, because it is applied to class ranks, which are already correlated. This method leads to the problem of mean squares error minimization. For both our methods we find the decision functions in the family of linear functions and use least squares optimization procedure. Experiments on multi-label benchmark dataset (text documents Reuters-2000) have demonstrated high accuracy of developed methods compared to existing ones. It confirms our hypothesis that to solve multi-label classification problem it is possible to define simple (linear) multi-label decision functions on class relevance vector space. This approach is precise enough and computationally inexpensive.

REFERENCES

[1] Schapire R. E., Singer. Y.: Improved boosting algorithms using confidence-rated predictions. Machine Learning, 37(3), 1999, pp. 297-336.

[2] Schapire R. E., Singer. Y.: BoosTexter: A boosting-based system for text categorization. Machine Learning, 39(2-3), 2000, pp. 135-168.

[3] McCallum A.: Multi-label text classification with a mixture model trained by EM. Working Notes of the AAAI'99 Workshop on Text Learning, Orlando, FL, 1999.

[4] Crammer C., Singer Y.: A new family of online algorithms for category ranking. Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval. Tampere, Finland, 2002, pp. 151 – 158.

[5] Crammer C., Singer Y.: A family of additive online algorithms for category ranking. Journal of Machine Learning Research, 3, 2003, pp. 1025–1058.

[6] Elisseeff A., Weston J.: Kernel methods for multi-labelled classification and categorical regression problems. Technical report, BIOwulf Technologies, 2001.

[7] Elisseeff A., Weston J.: A kernel method for multi-labelled classification. Proceedings of the 14th Neural Information Processing Systems (NIPS) Conference, Cambridge, 2002.

[8] Zhang M.-L., Zhou Z.-H.: A k-nearest neighbor based algorithm for multi-label classification. Proceedings of the 1st IEEE International Conference on Granular Computing (GrC'05), Beijing, China, 2005, pp. 718-721.

[9] Boutell M. R., Luo J., Shen X., Brown C.M.: Learning multi-label scene classification. Pattern Recognition, 37, 2004, pp. 1757-1771.

[10] Comite F. D., Gilleron R., Tommasi M.: Learning multi-label alternating decision tree from texts and data. Machine Learning and Data Mining in Pattern Recognition, MLDM 2003 Proceedings, Lecture Notes in Computer Science 2734, Berlin, 2003, pp. 35–49.

[11] Chang, C.-C., Lin, C.-J. LIBSVM: a library for support vector machines, 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[12] Minh Duc Cao, Xiaoying Gao.: Combining Content and Citation for Scientific Document Classification. AI2005, LNAI 3809, 2005, pp. 143-152.

[13] Freund Y., Mason L.: Alternating decision tree learning algorithm. In Proc. 16th International Conf. On Maching Learning., San Francisco, USA, 1999, pp. 124-133.