A modified and fast Perceptron learning rule and its use for Tag Recommendations in Social Bookmarking Systems

Anestis Gkanogiannis and Theodore Kalamboukis

Department of Informatics
Athens University of Economics and Business, Athens, Greece
utumno@aueb.gr tzk@aueb.gr

Abstract. A modified and fast to converge Perceptron learning rule algorithm is proposed as a general classification algorithm for linearly separable data. The strategy of the algorithm takes advantage of training errors to successively refine an initial Perceptron Classifier.
Original Perceptron learning rule uses training errors along with a parameter α (learning rate parameter that has to be determined) to define a better classifier. The proposed modification does not need such a parameter (in fact it is automatically determined during execution of the algorithm).
Experimental evaluation of the proposed algorithm on standard text classification collections, show that results compared favorably to those from state of the art algorithms such as SVMs. Experiments also show a significant improvement of the convergence rate of the proposed Perceptron algorithm compared to the original one.
Seeing the problem of this year's Discovery Challenge (Tag Recommendation), as an automatic text classification problem, where tags play the role of categories and posts play the role of text documents, we applied the proposed algorithm on the datasets for Task 2. In this paper we briefly present the proposed algorithm and its experimental results when applied on the Challenge's data.

1 Introduction

Text categorization is the process of making binary decisions about related or non-related documents to a given set of predefined thematic topics or categories. This task is an important component in many information management organizations. In our participation on the ECML/PKDD challenge 2009, we treat Task 2 as a standard text classification problem and try to solve it using a machine learning, supervised, automatic classification method.

The rest of the paper is organized as follows. Section 2 provides a description of the algorithm that we used. Section 3 briefly present the tasks of this year's Challenge. Section 4 presents the experimental setup, data processing and results and finally in section 5 we conclude on the results.

2 The Learning Algorithm

The algorithm that we used is an evolution of the algorithm that appeared in [1] as a text classification algorithm and then a revised version in [2] won last year's ECML PKDD Discovery Challenge on Spam Detection. The proposed algorithm is a binary linear classifier and it combines a centroid with a batch perceptron classifier and a modified perceptron learning rule that does not need any parameter estimation. Details on this modified algorithm, its experimental evaluation, theoretical investigation etc, have already submitted and are under review for publication at the time this paper was written. In the following paragraphs we will briefly describe this method that we used for solving the problem of ECML PKDD 2009 Discovery Challenge, Task 2.

2.1 Linear Classifiers

Linear Classifiers is a family of classifiers whose trained model is a linear combination of features. In another perspective linear classifiers train a model which is a hyperplane in a high dimensional feature space. In this space each instance, either of the train set or an unseen, is a point. The goal of a linear classifier is then to find such a hyperplane that splits the space into two subspaces, where one contains all the points of the positive class and the other contains all the points of the negative class.

Assuming that feature space is of n dimensions, each instance x_i will be represented by an n dimensions vector

$$\overrightarrow{x}_i = (w_{i1}, w_{i2}, \cdots, w_{in}) \tag{1}$$

where w_{ik} is a real value of the kth feature for instance x_i.

Apart of each vector representation $\overrightarrow{x}_i$, each instance x_i may bears information about being a member of a class or not. For example a document is known to be spam or an image is known that shows a benign tumor. This information can be coded using a variable y_i for each instance x_i which takes values as:

$$y_i = \begin{cases} 1 & \text{if } x_i \in C_+ \\ -1 & \text{if } x_i \in C_- \end{cases} \tag{2}$$

That is $y_i = 1$ when x_i is member of the positive class C_+ and $y_i = -1$ when it is member of the negative class C_-. So each instance x_i is represented by a tuple $(\overrightarrow{x}_i, y_i)$. A training set Tr would be

$$Tr = \{(\overrightarrow{x}_1, y_1), (\overrightarrow{x}_2, y_2), \cdots, (\overrightarrow{x}_m, y_m)\} \tag{3}$$

A linear classifier then is defined by a model $\left\langle \overrightarrow{W}, b \right\rangle$ where $\overrightarrow{W}$ is a vector in the same n-dimensional space and b is a scalar bias (threshold) value. This model defines a hyperplane h

$$h : \overrightarrow{W} \cdot x + b = 0 \tag{4}$$

This is the equation of a hyperplane h in the n-dimensional space. This hyperplane is of $n-1$ dimensions. $\overrightarrow{W}$ is a linear combination of n features (dimensions). Hyperplane h splits space into two subspaces, the one where for every vector $\overrightarrow{x}_i : \overrightarrow{W} \cdot \overrightarrow{x}_i + b > 0$ and the other where $\overrightarrow{W} \cdot \overrightarrow{x}_i + b < 0$. Every vector for which $\overrightarrow{W} \cdot \overrightarrow{x}_i + b = 0$ lies on hyperplane h. The objective of each linear classifier is to define such $h : \left\langle \overrightarrow{W}, b \right\rangle$. Different linear classifiers have different ways to define model vector $\overrightarrow{W}$ and bias b.

2.2 Perceptron

Perceptron is a flavor of Linear Classifiers. It starts with an initial model and iteratively refines this model using the classifications errors during training. It is the elementary particle of neural networks and it has been investigated and studied since the 1950s [3]. It has been shown that when trained on a linearly separable set of instances, it converges (it finds a separating hyperplane) in a finite number of steps [4] (which depends on the geometric characteristics of the instances on their feature space).

The Perceptron is a Linear Binary Classifier that maps its input $\overrightarrow{x}$ (a real-valued vector) to an output value $f(\overrightarrow{x})$ (a single binary value) as:

$$f(\overrightarrow{x}) = \begin{cases} 1 & \text{if } \overrightarrow{W} \cdot \overrightarrow{x} + b > 0 \\ -1 & \text{else} \end{cases} \tag{5}$$

where $\overrightarrow{W}$ is a vector of real-valued weights and $\overrightarrow{W} \cdot \overrightarrow{x}$ is the dot product (which computes a weighted sum). b is the bias, a constant term that does not depend on any input value. The value of $f(\overrightarrow{x})$ (1 or -1) is used to classify instance x as either a positive or a negative instance, in the case of a binary classification problem. The bias b can be thought of as offsetting the activation function, or giving the output neuron a "base" level of activity. If b is negative, then the weighted combination of inputs must produce a positive value greater than $-b$ in order to push the classifier neuron over the 0 threshold. Spatially, the bias alters the position (though not the orientation) of the decision boundary (separating hyperplane h).

We can always assume for convenience that the bias term b is zero. This is not a restriction since an extra dimension $n+1$ can be added to all the input vectors $\overrightarrow{x}_i$ with $\overrightarrow{x}_i(n+1) = 1$, in which case $\overrightarrow{W}(n+1)$ replaces the bias term.

Learning is modeled as the weight vector $\overrightarrow{W}$ being updated for multiple iterations over all training instances. Let

$$Tr = \{(\overrightarrow{x}_1, y_1), (\overrightarrow{x}_2, y_2), \cdots, (\overrightarrow{x}_m, y_m)\}$$

denote a training set of m training examples (instances). At each iteration k the weight vector is updated as follows. For each $(\overrightarrow{x}_i, y_i)$ pair in Tr

$$\overrightarrow{W}^{(k)} = \overrightarrow{W}^{(k-1)} + \frac{\alpha^{(k-1)}}{2} \left(y_i - f^{(k-1)}\left(\overrightarrow{x}_i\right) \right) \overrightarrow{x}_i \tag{6}$$

where α is a constant real value in the range $0 < \alpha \leq 1$ and is called the learning rate. Note that equation 6 means that a change in the weight vector $\overrightarrow{W}$ will only take place for a given training example $(\overrightarrow{x}_i, y_i)$ if its output $f(\overrightarrow{x}_i)$ is different from the desired output y_i. In other words the weight vector will change only in the case where the model has made an error. The initialization of $\overrightarrow{W}$ is usually performed simply by setting $\overrightarrow{W}^{(0)} = 0$.

The training set Tr is said to be linearly separable if there exists a positive constant γ and a weight vector $\overrightarrow{W}$ such that

$$y_i \left(\overrightarrow{W} \cdot \overrightarrow{x}_i + b\right) > \gamma, \forall (\overrightarrow{x}_i, y_i) \in Tr \tag{7}$$

Novikoff [4] proved that the perceptron algorithm converges after a finite number of iterations k if the train data set is linearly separable. The number of mistakes (iterations) is bounded then by

$$k \leq \left(\frac{2R}{\gamma}\right)^2 \tag{8}$$

where $R = \max\{||\overrightarrow{x}_i||\}$ is the maximum norm of an input train vector.

2.3 Batch Perceptron

Equation 6 defines a single sample fixed increment perceptron learning rule. It is called fixed increment because parameter α is constant throughout training. In the case where this parameter changes at each iteration, we say that it is a variable increment perceptron. It is also called single sample because this rule applies at each instance x_i which was misclassified during iteration k. In other words, at iteration k each $(\overrightarrow{x}_i, y_i) \in Tr$ is presented to model $\overrightarrow{W}^{(k-1)}$ and if it is misclassified by it ($f^{(k-1)}(\overrightarrow{x}_i) \neq y_i$) then this single instance $\overrightarrow{x}_i$ is used (along with parameter $\alpha^{(k-1)}$) to alter $\overrightarrow{W}^{(k-1)}$ into $\overrightarrow{W}^{(k)}$.

A modification of this perceptron can be made defining a set of instances $Err \subset Tr$

$$Err = \{(\overrightarrow{x}_i, y_i)\}, f^{(k-1)}(\overrightarrow{x}_i) \neq y_i \tag{9}$$

that contains all the misclassified examples at iteration k and then modifying weight vector as:

$$\overrightarrow{W}^{(k)} = \overrightarrow{W}^{(k-1)} + \alpha^{(k-1)} \sum_{(\overrightarrow{x}_i, y_i) \in Err} y_i \overrightarrow{x}_i \tag{10}$$

In the case where bias value is not incorporated into example and weight vectors (via an additional n+1 dimension), then bias value is modified as:

$$b^{(k)} = b^{(k-1)} + \alpha^{(k-1)} \sum_{(\overrightarrow{x}_i, y_i) \in Err} y_i \tag{11}$$

Equations 10 and 11 are called a Batch Perceptron learning rule and as the single sample perceptron, parameter $\alpha^{(k-1)}$ can be constant (fixed increment) or varying at each iteration (variable increment).

2.4 Centroid Classifier

A Centroid classifier is a simple linear classifier, that will help us understand the notion behind our modification presented in the next Section. In the simple binary case there are two classes, the positive and the negative one. We define set C_+ and C_- containing instances from the positive and respectively the negative class. We call Centroid of the positive class and respectively the Centroid of the negative class as

$$\overrightarrow{C}_+ = \frac{1}{|C_+|} \sum_{\overrightarrow{x_i} \in C_+} \overrightarrow{x_i} \tag{12}$$

$$\overrightarrow{C}_- = \frac{1}{|C_-|} \sum_{\overrightarrow{x_i} \in C_-} \overrightarrow{x_i} \tag{13}$$

We then define a linear classifier as

$$h : \overrightarrow{W} \cdot \overrightarrow{x} + b = 0 \tag{14}$$

where

$$\overrightarrow{W} = \overrightarrow{C}_+ - \overrightarrow{C}_- \tag{15}$$

and bias value b is defined by some technique we discuss in the following paragraphs.

Figure 1 illustrates a simple case of a centroid classifier in a 2-dimensional space. Sets C_+ of the positive class and C_- of the negative class are shown along with their centroid vectors $\overrightarrow{C}_+$ and $\overrightarrow{C}_-$ respectively. We note that in this simple example, these two classes are linearly separable and therefore it is possible to find a value for bias b such that h is a perfect separating hyperplane.

A method for finding such a value is Scut [5], where we iteratively choose values for bias b and then keep the one that lead to the best classifier (as measured by some evaluation measurement). Bias takes values as

$$b_i = \overrightarrow{W} \cdot \overrightarrow{x}_i, \forall \overrightarrow{x}_i \in Tr \tag{16}$$

and then an evaluation measure (for example the F_1 measure) is computed for classifier $h : \overrightarrow{W} \cdot \overrightarrow{x} + b_i = 0$. Finally as bias value is chosen the one that gave the maximum evaluation measure. It is clear that the instance x_i that corresponds to the chosen b_i lies on hyperplane h. In the shown 2-dimensional example of Figure 1 this instance is marked by point $\overrightarrow{x}_{Scut}$.

This simple algorithm has previously investigated and
methods have been proposed for altering initial centroids or weights in order to achieve a better classifier [6–8].

In the next subsection we present how ideas from Centroid Classifier and Perceptron are combined to our modified version of Perceptron.

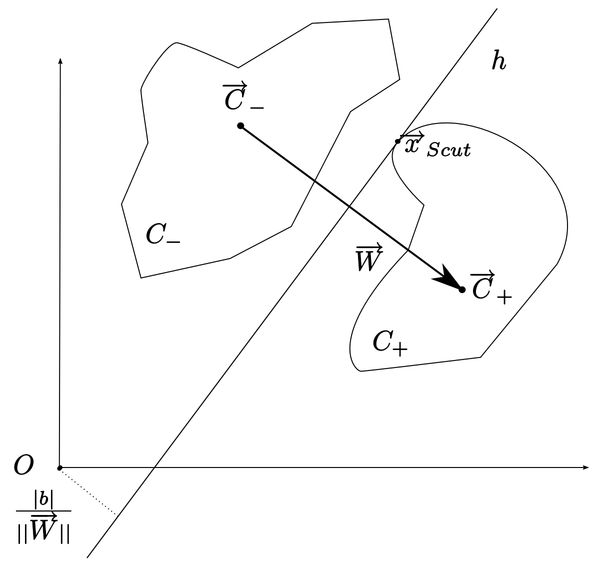

Fig. 1. A simple Centroid Classifier in 2 dimensions. The positive class C_+ is linearly separable from the negative one C_-.

2.5 The proposed modification to Perceptron

Centroid Classifier of the previous subsection can be seen as a perceptron with initial weight vector $\overrightarrow{W}^{(0)} = \overrightarrow{C}_+ - \overrightarrow{C}_-$, bias value b as defined by an Scut method and no other training adjustments at all. The case shown in Figure 1 is an ideal case for a Centroid Classifier, meaning that it is possible to find a value for b resulting to a perfect separating hyperplane $h : \overrightarrow{W} \cdot \overrightarrow{x} + b = 0$.

This is not however true in all cases. Figure 2 shows such a case where finding a perfect separating hyperplane is not possible for a simple Centroid Classifier. Dark regions contains misclassified instances that cannot correctly classified. A Simple Sample or a Batch Perceptron would use these errors to modify the weight vector $\overrightarrow{W}$.

If we define sets FP and FN as:

$$FP = \{(\overrightarrow{x}_i, y_i)\} \forall x_i \in C_-, f(\overrightarrow{x}_i) \neq y_i \quad (17)$$
$$FN = \{(\overrightarrow{x}_i, y_i)\} \forall x_i \in C_+, f(\overrightarrow{x}_i) \neq y_i \quad (18)$$

in other words set FP contains negative instances that were misclassified as positive (False Positive), whereas set FN contains positive instances that were misclassified as negative (False Negative). A Batch Perceptron then using mis-

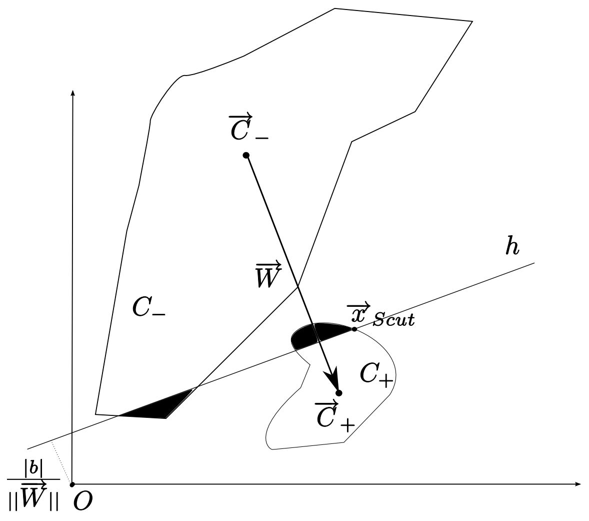

Fig. 2. No perfect separating hyperplane exists for this Centroid Classifier. Dark regions are misclassified.

classified instances modifies weight vector as Equation 10 or equivalently as:

$$\overrightarrow{W}^{(k+1)} = \overrightarrow{W}^{(k)} + \alpha^{(k)} \left(\sum_{\overrightarrow{x}_i \in FN^{(k)}} \overrightarrow{x}_i - \sum_{\overrightarrow{x}_i \in FP^{(k)}} \overrightarrow{x}_i \right) \tag{19}$$

However there is a parameter α, either constant or variable that needs to be estimated. This learning rate parameter is strongly related to the field on which perceptron learning is applied and train data itself. A way to estimate it is using a validation set of instances and selecting a value for α that leads to maximum performance. But this operation must be repeated whenever field of operation or data is switched and costs very much in terms of time.

Another approach is to use a fixed value for the learning rate like $\alpha = 1$ or $\alpha = 0.5$ for example, without attempting to find a optimal value. However this could result to very unwanted effects because learning rate is too small or too large for the specific field of operation and training instances.

The key idea of our approach is illustrated in Figure 3 where we concentrate on the misclassified regions. Positive class and a portion of negative class are shown. Initial weight vector $\overrightarrow{W}^{(0)}$ and hyperplane $h^{(0)}$ are defined by a simple Centroid Classifier. The idea is, at the next iteration 1, to modify weight vector and bias into $\overrightarrow{W}^{(1)}$ and $b^{(1)}$ such that the resulting hyperplane $h^{(1)}$ passes through the points defined by centroid vectors of the misclassified regions FP and FN.

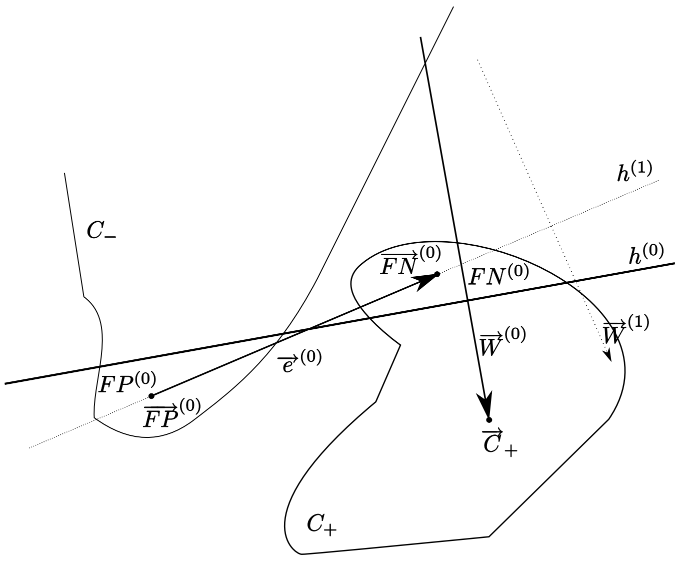
C_-
$h^{(1)}$
$h^{(0)}$
$\overrightarrow{FN}^{(0)}$
$FN^{(0)}$
$\overrightarrow{W}^{(1)}$
$\overrightarrow{W}^{(0)}$
$\overrightarrow{e}^{(0)}$
$FP^{(0)}$
$\overrightarrow{FP}^{(0)}$
$\overrightarrow{C}_+$
C_+

Fig. 3. The proposed modification to batch perceptron.

We define these misclassified centroids at each iteration as

$$\overrightarrow{FP}^{(k)} = \frac{1}{|FP^{(k)}|} \sum_{\overrightarrow{x_i} \in FP^{(k)}} \overrightarrow{x_i} \tag{20}$$

$$\overrightarrow{FN}^{(k)} = \frac{1}{|FN^{(k)}|} \sum_{\overrightarrow{x_i} \in FN^{(k)}} \overrightarrow{x_i} \tag{21}$$

where sets FP and FN are defined in Equations 17 and 18. We then define the error vector at each iteration as

$$\overrightarrow{e}^{(k)} = \overrightarrow{FN}^{(k)} - \overrightarrow{FP}^{(k)} \tag{22}$$

Batch Perceptron learning rule of Equation 19 is then modified to:

$$\overrightarrow{W}^{(k+1)} = \overrightarrow{W}^{(k)} + \alpha'^{(k)} \overrightarrow{e}^{(k)} \tag{23}$$

We can easily compute the value of this modified learning rate $\alpha'^{(k)}$ if we note that misclassified centroids $\overrightarrow{FN}^{(k)}$ and $\overrightarrow{FP}^{(k)}$ lie by construction on the new hyperplane $h^{(k+1)}$. As a result error vector $\overrightarrow{e}^{(k)}$ is vertical to the new normal vector $\overrightarrow{W}^{(k+1)}$. So

$$\overrightarrow{W}^{(k+1)} \cdot \overrightarrow{e}^{(k)} = 0$$

$$\left(\overrightarrow{W}^{(k)} + \alpha'^{(k)}\overrightarrow{e}^{(k)}\right) \cdot \overrightarrow{e}^{(k)} = 0$$
$$\overrightarrow{W}^{(k)} \cdot \overrightarrow{e}^{(k)} + \alpha'^{(k)}||\overrightarrow{e}^{(k)}||^2 = 0$$
$$\alpha'^{(k)} = -\frac{\overrightarrow{W}^{(k)} \cdot \overrightarrow{e}^{(k)}}{||\overrightarrow{e}^{(k)}||^2}$$

And then the modified learning rule of Equation 23 is

$$\overrightarrow{W}^{(k+1)} = \overrightarrow{W}^{(k)} - \frac{\overrightarrow{W}^{(k)} \cdot \overrightarrow{e}^{(k)}}{||\overrightarrow{e}^{(k)}||^2}\overrightarrow{e}^{(k)} \tag{24}$$

This is the normal vector defining the direction of the next hyperplane $h^{(k+1)}$. The actual position of it is determined by the new bias value which is easily computed (bringing in mind that misclassified centroids lie on the new hyperplane):

$$b^{(k+1)} = -\overrightarrow{W}^{(k+1)} \cdot \overrightarrow{FP}^{(k)} = -\overrightarrow{W}^{(k+1)} \cdot \overrightarrow{FN}^{(k)} \tag{25}$$

Equations 24 and 25 define the new hyperplane

$$h^{(k+1)} : \overrightarrow{W}^{(k+1)} \cdot \overrightarrow{x} + b^{(k+1)} = 0$$

3 Task Description

As last year's, this year's ECML PKDD Discovery Challenge deals with the well known social bookmarking system called Bibsonomy [1]. In such systems, users can share with everyone links to web pages or scientific publications. The former are called bookmark posts, where the later are called bibtex posts. Apart from posting the link to the page or the publication, users can assign tags (labels) to their posts. Users are free to choose their own tags or the system can assist them by suggesting them the appropriate tags.

This year's Discovery Challenge problem is about generating methods that would assist users of social bookmarking systems by recommending them tags for their posts. There are two distinct task for this problem. Task 1 is about recommending tags to posts over an unknown set of tags. That means that the methods developed for Task 1 must be able to suggest tags that are unknown (in other words suggest new tags). Task 2, on the other hand, is about recommending tags that have been already known to be ones. [2]

3.1 Data Description

Data provided for these tasks was extracted from Bibsonomy databases. Two datasets where provided, one for training participant's methods, and the other

[1] http://www.bibsonomy.org

[2] More details about tasks can be found on Challenge's site at http://www.kde.cs.uni-kassel.de/ws/dc09/#tasks

for evaluating their performance. Both of them where provided as a set of 3 files (tas, bookmark, bibtex). Files bookmark and bibtex contain textual data of the corresponding posts. File tas contains which user assigned which tags to which bookmark or bibtex resource. Each triplet (user,tags,resource) defines a post. Train and test files where of the same tabular format, except test tas file which of course did not contain tag information, as this was Challenge's goal. [3]

More details about preprocessing of the datasets will be given on the following section 4.

4 Experimental Setup and Results

Challenge's organizers had suggest that graph method would fit better to task 2, whereas content based method would fit to task 1. In our work we concentrated on task 2, and from this point on whenever we mention a task, we mean task 2. Although organizers suggested graph method for the task, we choose to use our modified perceptron rule for solving this problem. We made this decision because we wanted to test the performance and robustness of the proposed algorithm on a domain with a large category set. As we are going to present in our under review paper, we have evaluated the proposed algorithm on standard text classification datasets as well as on artificially generated (and linearly separable) datasets. Although feature spaces of these datasets are of tens or hundreds of thousands features, their categories sets are of few to at most a thousand categories. We wanted to investigate how this method is going to perform when both feature and category spaces are large.

So, Task 2 can be seen as a standard text classification problem, and the proposed algorithm as a machine learning, supervised, automatic classification method that applies on it. In this problem, tags (labels) that assigned on posts can be seen as categories. On the other hand, posts can be seen as text documents, where category labels (tags) are assigned on them.

4.1 Data Preprocessing

Viewing task 2 as a supervised text classification problem, implies that datasets must transformed to a vector space, where the proposed linear classifier can be used. For every post (user,tags,resource), we construct a text document and then transform it to the vector space.

We choose to discard user information from the posts, so the only textual information for each post came from the assigned tags and the resource. Furthermore for each bookmark post we kept *url*, *description* and *extended_description* fields. For each bibtex post we kept *journal*, *booktitle*, *url*, *description*, *bibtexAbstract*, *title* and *author* fields.

Fore every post, and using those field, we construct a text document. We then transform document dataset to a vector space. First tokenization of the

[3] More details about datasets can be found at http://www.kde.cs.uni-kassel.de/ws/dc09/dataset

text, then stop word removal, then stemming (using Porter's stemmer [9], then term and feature extraction and finally feature weighting using *tf*idf* statistics.

The following table 1 presents some statistics about categories (tags) and documents (posts) in the train and the test dataset.

	Train dataset	**Test dataset**
Number of Documents	64,120	778
Number of Categories	13,276	-

Table 1. Statistics for categories and documents in datasets

The following diagram 4 presents the distribution of the sizes of categories in the train dataset. Axis ***x*** denotes the number of categories that are of a certain size. Axis ***y*** denotes the number of documents that a certain sized category contains.

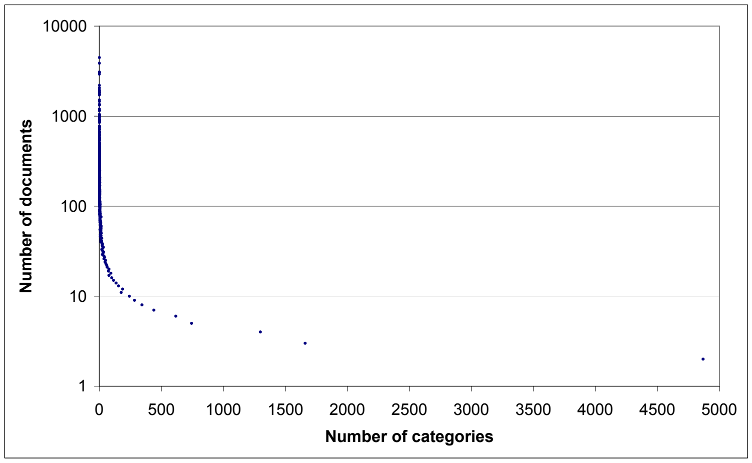

Fig. 4. Distribution of the sizes of categories in the train dataset

We note that categories sizes are small in general. In fact 10,500 out of 13,276 categories have at most 10 documents. The average size of categories is 1.97 (average posts per tag).

4.2 Experimental Setup and Train phase

After converting documents (posts) into vectors in a high-dimensional space, we can apply the proposed text classification method for solving the multilabel classification problem. Since the method trains a binary linear classifier, the problem must be transformed into binary classification. This is done by cracking the problem into multiple binary classification problems. So, at the end we have to solve $13,276$ binary classification problems.

The number of problems is quite large and therefore the used method must be as much fast as possible. After the train phase (which finishes after the reasonable time of 2 hours in a mainstream laptop), the final classification system consists of $13,276$ binary classifiers.

4.3 Test phase and Results

Test phase consists of presenting each document of the test dataset (778 in total) to every binary classifier resulted from training phase ($13,276$ in total). Each classifier decides whether the presented document (post) belongs or not to the corresponding category (tag). Time needed fore presenting all document to all classifiers on a mainstream laptop was about 10 minutes (that is about 0.8 seconds for a document to pass through all classifiers).

We produced 2 types of results. The ones that come from binary classification and the ones that come from ranking. During binary classification a document could be assigned or not into a category. Therefore a document, after been presented to every binary classifier, could be assigned to zero, one, or more categories (max is $13,276$ of course).

On the ranking mode, a classifier gives a score to each presented document (higher score mean higher confidence of the classifier that this document belongs to the corresponding category). Therefore at this mode, a document can be assigned to any number z of categories we select (simply by selecting the z categories which gave the higher scores).

We chose our submission to the Challenge, to contain results of the ranking mode (by selecting the 5 higher scored categories for each document).

After releasing the original tag assignments of the test dataset, our results of the ranking mode achieved a performance of $F_1 = 0.1008$. The results of the first mode (binary mode), that where never submitted, achieved a performance of $F_1 = 0.1622$. Of course, those results could not have been known prior releasing original test tas file, but we had a belief that the ranking mode (suggesting 5 tags for every post, instead of less or even zero) would had better results. Unfortunately this belief was false.

5 Concluding Remarks

In this paper we described the application of a modified version of the Perceptron learning rule on Task 2 of ECML PKDD Discovery Challenge 2009. This algorithm acts as a supervised machine learning, automatic text classification algorithm on the data of the task. Task 2 is transformed to a supervised text classification problem by treating users' posts ass text documents and assigned tags as thematic categories.

This algorithm has been prior tested on various text classification datasets and artificially generated linearly separable datasets, and it has shown a robust performance and efficiency. Compared with the original Batch Perceptron learning algorithm, it shows a significant improvement on the convergence rate.

Its fast training phase made it feasible to be used on Task 2 dataset, which consists of a large categories set (more than $13,000$ categories) and a linear classifier had to be trained for each category.

Although its results on Task 2 test dataset where not so well, we think that its fast training phase and fast evaluation (since it is just a dot product for each category-document tuple) allow for further investigation.

Acknowledgments

This paper is part of the 03ED316/8.3.1. research project, implemented within the framework of the "Reinforcement Programme of Human Research Manpower" (PENED) and co-financed by National and Community Funds (20% from the Greek Ministry of Development-General Secretariat of Research and Technology and 80% from E.U.-European Social Fund).

References

1. Gkanogiannis, A., Kalampoukis, T.: An algorithm for text categorization. In: 31st ACM International Conference on Research and Development in Information Retrieval SIGIR-2008. (2008) 869–870
2. Gkanogiannis, A., Kalamboukis, T.: A novel supervised learning algorithm and its use for spam detection in social bookmarking systems. In: ECML PKDD Discovery Challenge '08. (2008)
3. Rosenblatt, F.: The perceptron: a probabilistic model for information storage and organization in the brain. Psychological Review **65**(6) (November 1958) 386–408
4. Novikoff, A.B.: On convergence proofs for perceptrons. In: Proceedings of the Symposium on the Mathematical Theory of Automata. Volume 12. (1963) 615–622
5. Yang, Y.: A study on thresholding strategies for text categorization (2001)
6. Karypis, G., Shankar, S.: Weight adjustment schemes for a centroid based classifier (2000)
7. Harman, D.: Relevance feedback and other query modification techniques. (1992) 241–263
8. Buckley, C., Salton, G.: Optimization of relevance feedback weights. In: SIGIR '95: Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval, New York, NY, USA, ACM (1995) 351–357
9. Porter, M.F.: An algorithm for suffix stripping. Program **14**(3) (1980) 130–137