A Perceptron-Like Linear Supervised Algorithm for Text Classification

Conference Paper · November 2010

DOI: 10.1007/978-3-642-17316-5_8 · Source: DBLP

2 authors:

Anestis Gkanogiannis
Athens University of Economics and Business
9 PUBLICATIONS **64** CITATIONS
SEE PROFILE

Theodore Kalamboukis
Athens University of Economics and Business
34 PUBLICATIONS **370** CITATIONS
SEE PROFILE

A Perceptron-Like Linear Supervised Algorithm for Text Classification

Anestis Gkanogiannis and Theodore Kalamboukis

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

Abstract. A fast and accurate linear supervised algorithm is presented which compares favorably to other state of the art algorithms over several real data collections on the problem of text categorization. Although it has been already presented in [6], no proof of its convergence is given. From the geometric intuition of the algorithm it is evident that it is not a Perceptron or a gradient descent algorithm thus an algebraic proof of its convergence is provided in the case of linearly separable classes. Additionally we present experimental results on many standard text classification datasets and artificially generated linearly separable datasets. The proposed algorithm is very simple to use and easy to implement and it can be used in any domain without any modification on the data or parameter estimation.

Keywords: Perceprton, Machine Learning, Classification.

1 Introduction

The increasing availability of text in digital form demands fast and accurate methods for classifying it. Classification is the process of assigning elements into a set of predefined classes. Over the last years the problem has found several important applications. For example, in text classification it is used for text filtering [21,22,5,3], in social networks to automatically apply tags (annotation) to new posts for spamming detection, in medicine and many others.

Binary classification is a special case, where instances are assigned into two classes namely, the positive class and its complement or negative class. A multi-labeled classification problem can always be transformed to a set of binary classification problems. To solve these binary classification problems, several Machine Learning techniques and algorithms have been developed. For example, there are Naive Bayesian techniques, Decision Trees, Neural Networks, Linear Classifiers [15], Instance Based algorithms, Support Vector Machines [2], etc. These methods train a model by using a training set and then use this model to classify new unseen instances (Supervised Learning techniques).

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 defines a hyperplane in a high dimensional feature space. The goal of a linear

L. Cao, J. Zhong, and Y. Feng (Eds.): ADMA 2010, Part I, LNCS 6440, pp. 86–97, 2010.

classifier is to find a perfect hyperplane that splits the space into two subspaces: one that contains all the positive examples and another that contains the negative ones. There are a lot perfect hyperplanes for a given binary classification problem.

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 [19]. It has been shown that when trained on a linearly separable set of instances, it converges to a separating hyperplane in a finite number of steps [17]. The convergence rate depends on the geometric characteristics of the instances on their feature space.

In this paper we briefly present a method, which has been already presented by [6] on the dataset of the ECML/PKDD 2009 Discovery Challenge. Although at a first glance looks very similar to Perceptron method, it uses a similar main iterative form of model refinement but exploits the geometric characteristics of the train instances in order to find faster a separating hyperplane. In the case of a non separable set it acts in a best-effort manner, preserving the best hyperplane, after applying a max predefined number of steps. In this paper the proposed algorithm is compared with the original perceptron, on various datasets used primarily in the text classification field and also on several artificially generated linearly separable datasets. Experimental results show that the algorithm converges much faster than perceptron. As far as the performance is concerned there are no major differences in the case of the artificial data, since both methods eventually find a separating hyperplane. However there are significant differences in favor of the proposed algorithm in the case of real text datasets (these datasets in general are not linearly separable). Finally the method is compared to the state-of-the-art method in the Machine Learning Classification field, the SVM method [9,10], and the results show that it behaves comparably if not better.

The next Section 2 describes briefly the Linear Classifiers and Perceptron. In Section 3 we briefly present the proposed algorithm and a proof of its convergence. In Section 4 we describe the datasets used for comparison and evaluation and present the experimental results. Finally Section 5 concludes with our findings and future extensions.

2 Linear Classifiers

Assuming that the feature space is of n dimensions, an instance x_i will be represented as a n dimensional vector

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

where w_{ik} denotes the weight of the k^{th} feature in the TF*IDF weighting scheme [20]. Each instance in the training set is accompanied with the class label, a value y_i defined by:

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

were C_+ denotes the positive class and C_- the negative one. The training set Tr would be a set of tuples

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

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

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

and a decision for an instance is taken by function

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

The distinction of the different linear classifiers in the literature lie on the approach of defining the weight vector $\overrightarrow{W}$ and the bias b. Thus there are several algorithms ranging from very simple ones, like the centroid classifier, to more sophisticated ones, like the Perceptron, winnow, Widrow-Hoff, algorithms that try to minimize a cost function, like steepest descent algorithm, to the state of the art algorithm of SVMs that constructs the best separating hyperplane by solving a quadratic optimization problem.

2.1 Centroid Classifier

A Centroid classifier is a simple linear classifier, that will help us to understand the notion behind the proposed algorithm presented in the next Section. We first construct the centroid vectors of the two classes, C_+ and C_-,

$$\overrightarrow{C}_+ = \frac{1}{|C_+|} \sum_{\overrightarrow{x_i} \in C_+} \overrightarrow{x_i} \quad , \quad \overrightarrow{C}_- = \frac{1}{|C_-|} \sum_{\overrightarrow{x_i} \in C_-} \overrightarrow{x_i} \ , \tag{6}$$

and then the classifier is defined by vector $\overrightarrow{W}$:

$$\overrightarrow{W} = \overrightarrow{C}_+ - \overrightarrow{C}_- \ , \tag{7}$$

which is the normal vector of the separating hyperplane, and a bias value b which defines its displacement from the origin.

Figure 1 illustrates a simple case of a centroid classifier in a 2-dimensional space. We note that in this simple example, it is possible to find a value for the bias b such that $< \overrightarrow{W}, b >$ is a perfect separating hyperplane. The bias takes a value

$$b_i = \overrightarrow{W} \cdot \overrightarrow{x}_i, \forall \overrightarrow{x}_i \in Tr \ , \tag{8}$$

which maximizes the performance measure F_1 of the classifier $< \overrightarrow{W}, b_i >$. This value is referred in the literature as the Scut [23] value. In Figure 1 this instance is marked by the point $\overrightarrow{x}_{Scut}$.

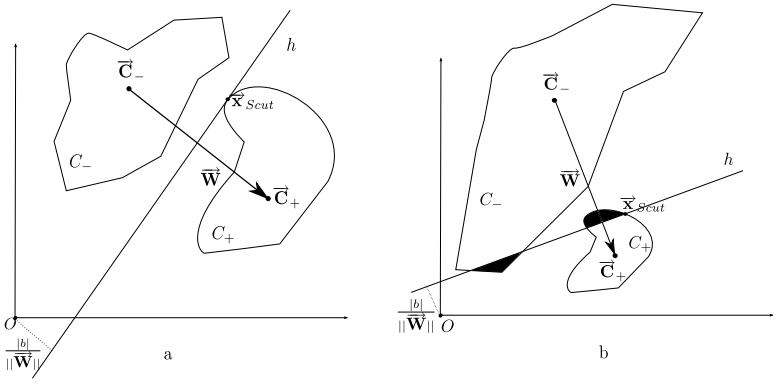

Fig. 1. A simple Centroid Classifier in 2 dimensions for two linearly separable datasets

This simple algorithm has been investigated and several methods have been proposed for altering the initial centroids in order to achieve a better classifier [12,7,1].

2.2 Batch Perceptron

Perceptron is a simple algorithm that has been extensively used in the literature. The algorithm appears in two modes: as a single-sample (online algorithm) and as batch mode with a fixed or a variable learning rate. In this section we briefly describe the batch version with variable learning rate that resembles a similarity with the proposed algorithm.

The algorithm starts with an arbitrary vector $\overrightarrow{W}_0$ which is updated at each iteration step based on the misclassified examples. If we define at iteration step k the set:

$$Err_k = \{(\overrightarrow{x}_i, y_i)\}, f_k(\overrightarrow{x}_i) \neq y_i\} \ , \tag{9}$$

of the examples misclassified by the classifier, then $\overrightarrow{W}_k$ is updated as:

$$\overrightarrow{W}_{k+1} = \overrightarrow{W}_k + \alpha_k \sum_{(\overrightarrow{x}_i, y_i) \in Err_k} y_i \overrightarrow{x}_i \ . \tag{10}$$

Similarly the bias is updated by the relation:

$$b_{k+1} = b_k + \alpha_k \sum_{(\overrightarrow{x}_i, y_i) \in Err_k} y_i \ . \tag{11}$$

The learning rate denoted with α_k should form a decreasing sequence in order for the method to converge [4]. Such a value is for example $\alpha_k = c/k$, where c is a constant.

3 The Proposed Algorithm

Figure 1a shows an ideal case where the centroid classifier defines a perfect separating hyperplane. This is not however in general the case as it is shown

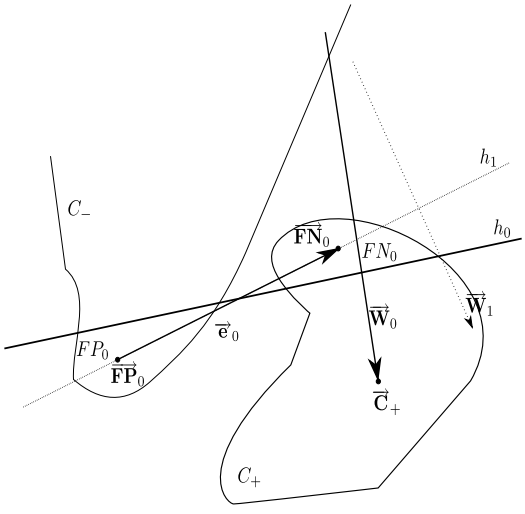

Fig. 2. The new hyperplane passes through the centroids of the misclassified sets, FN and FP

in Figure 1b. In such a case a Perceptron type algorithm may use the centroid classifier as a starting vector to find a separating hyperplane following the process described in the previous section.

A weak point of Perceptron algorithm is determining the value of the learning rate. A too large choice of α_k causes the hyperplane to swing wildly moving away the optimum, while a too small value slows down the convergence rate which makes the algorithm not scalable to large problems.

The proposed algorithm starts with an initial weight vector and bias as defined by the centroid classifier, relations 7 and 8. Then the algorithm proceeds by iteratively modifying the weight vector $\overrightarrow{W}_k$ by an error vector

$$\overrightarrow{e}_k = \overrightarrow{FN}_k - \overrightarrow{FP}_k \ , \tag{12}$$

defined by the difference of the centroid vectors of the misclassified regions FP (False Positive examples) and FN (False Negative) (see figure 2). We define these misclassified centroids at each iteration as

$$\overrightarrow{FP}_k = \frac{1}{|FP_k|} \sum_{\overrightarrow{x_i} \in FP_k} \overrightarrow{x_i} \ , \tag{13}$$

$$\overrightarrow{FN}_k = \frac{1}{|FN_k|} \sum_{\overrightarrow{x_i} \in FN_k} \overrightarrow{x_i} \ . \tag{14}$$

The update rule of the algorithm thus becomes:

$$\overrightarrow{W}_{k+1} = \overrightarrow{W}_k + \alpha_k \overrightarrow{e}_k \ . \tag{15}$$

The bias b_{k+1} is estimated by forcing the resulting hyperplane to pass through the centroids of the misclassified regions, that is

$$b_{k+1} = -\overrightarrow{W}_{k+1} \cdot \overrightarrow{FP}_k = -\overrightarrow{W}_{k+1} \cdot \overrightarrow{FN}_k \ . \tag{16}$$

This enforcement means that $\overrightarrow{W}_{k+1} \perp \overrightarrow{e}_k$ or $\overrightarrow{W}_{k+1} \cdot \overrightarrow{e}_k = 0$.

From the last relation and the update rule 15 follows that:

$$\alpha_k = -\frac{\overrightarrow{W}_k \cdot \overrightarrow{e}_k}{||\overrightarrow{e}_k||^2} \ . \tag{17}$$

From the description of the algorithm it is evident that although similar is not a Perceptron algorithm. Thus a convergence proof is necessary.

3.1 Convergence Proof of the Proposed Algorithm

Theorem. Given a set of linearly separable training examples x_i if $\overrightarrow{W}^*$ denotes a perfect solution hyperplane such that: $R^2 = \min ||e_k||^2$ and $\gamma = \min W^* e_k$ over all possible nonempty subsets of S, then the update rule in Eq. (15) converges to W^* if $\alpha_i > 0$ and

$$\lim_{k \to \infty} \sum_{i=1}^{k} \alpha_i = \infty \ . \tag{18}$$

Proof. We shall show that each update brings the weight vector closer to the solution region. In particular we prove that $||W_{k+1} - \lambda W^*|| \leq ||W_k - \lambda W^*||$ for W^* sufficiently long ($\lambda > 0$).

$$\begin{aligned} ||\overrightarrow{W}_{k+1} - \lambda \overrightarrow{W}^*||^2 &= ||\overrightarrow{W}_k - \lambda \overrightarrow{W}^*||^2 + \alpha_k^2 ||\overrightarrow{e}_k||^2 + 2\alpha_k \overrightarrow{W}_k \cdot \overrightarrow{e}_k - 2\lambda \alpha_k \overrightarrow{W}^* \cdot \overrightarrow{e}_k \\ &\leq ||\overrightarrow{W}_k - \lambda \overrightarrow{W}^*||^2 + \alpha_k^2 ||\overrightarrow{e}_k||^2 + 2\alpha_k \overrightarrow{W}_k \cdot \overrightarrow{e}_k \end{aligned}$$

The last inequality holds since $\overrightarrow{W}^* \cdot \overrightarrow{e}_k > 0$. The term $\alpha_k^2 ||\overrightarrow{e}_k||^2 + 2\alpha_k \overrightarrow{W}_k \cdot \overrightarrow{e}_k \leq 0$ when $\alpha_k = -\frac{2\overrightarrow{W}_k \cdot \overrightarrow{e}_k}{||\overrightarrow{e}_k||^2} \geq 0$. Therefore the optimal value of the learning rate α_k is: (derivative equal to zero)

$$\alpha_k = -\frac{\overrightarrow{W}_k \cdot \overrightarrow{e}_k}{||\overrightarrow{e}_k||^2} \geq 0 \ . \tag{19}$$

By substituting the optimal value of α_k we get:

$$\begin{aligned} ||\overrightarrow{W}_{k+1} - \lambda \overrightarrow{W}^*||^2 &= ||\overrightarrow{W}_k - \lambda \overrightarrow{W}^*||^2 - \alpha_k^2 ||e_k||^2 - 2\lambda \alpha_k \overrightarrow{W}^* \cdot \overrightarrow{e}_k \leq \\ &\leq ||\overrightarrow{W}_k - \lambda \overrightarrow{W}^*||^2 - \alpha_k^2 R^2 - 2\lambda \alpha_k \gamma \end{aligned}$$

where R^2 and γ are defined over all the misclassified subsets of the training set. After k updates we have:

$$||\overrightarrow{W}_{k+1} - \lambda \overrightarrow{W}^*||^2 \leq ||\overrightarrow{W}_0 - \lambda \overrightarrow{W}^*||^2 - R^2 \sum_{i=1}^{k} \alpha_i^2 - 2\lambda\gamma \sum_{i=1}^{k} \alpha_i \ . \tag{20}$$

From (20) follows that there is a k_0 such that $\forall k \geq k_0$ the right hand side is negative if relation (18) holds. Thus for $k \geq k_0$, $||\overrightarrow{W}_{k+1} - \lambda \overrightarrow{W}^*||^2 = 0$ or $\overrightarrow{W}_{k+1} \to \lambda \overrightarrow{W}^*$.

4 Experimental Results

In this Section we describe the datasets and present experimental results of text classification from several real datasets as well as from artificially generated linearly separable datasets.

4.1 Text Classification Datasets

For evaluation of the performance, scalability and robustness of the algorithms, five main text corpuses were used; namely the Reuters-21578 corpus, the Reuters-RCV1, the OHSUMED, the TREC-AP corpus and the 20 Newsgroup corpus.

From these corpuses, we have constructed various subsets in order to explore the behavior of the algorithms under different synthesis and size of both the train and the testing set. Training and testing instances are represented by vectors with tf*idf weights. Stemming was applied using Porter's stemmer [18] and stop words were removed. In the following paragraphs the text datasets are briefly described. Some statistics of the datasets are presented in Table 1.

Table 1. Statistics of text datasets used for evaluation

Dataset Name	Train Set Documents	Test Set Documents	Train Set Unique Terms	Test Set Unique Terms
Reuters-10	9,603	3,299	19,970	11,673
Reuters-90	9,603	3,299	19,970	11,673
Reuters-10-x	6,490	2,545	16,442	9,916
Reuters-90-x	7,770	3,019	18,029	11,129
Reuters-RCV1-103	23,149	781,265	45,860	275,268
Ohsumed-23	6,286	7,643	20,338	22,566
Ohsumed-96	183,229	50,216	98,893	57,497
Ohsumed-49	183,229	50,216	98,893	57,497
Ohsumed-28	183,229	50,216	98,893	57,497
Ohsumed-96-x	12,822	3,760	17,323	10,842
Ohsumed-49-x	12,445	3,632	17,031	10,625
Ohsumed-28-x	953	274	5,201	3,153

Reuters-21578. Reuters-21578[1] corpus. In the Mod-Apte split [14,15], which it is used, only 90 categories have at least one document in the training and one in the testing set. This dataset is called, Reuters-90. Reuters-10 dataset contains only the 10 largest classes. The subsets, Reuters-90-x and Reuters-10-x, contain documents exclusively from the corresponding, 90 or 10, categories.

[1] http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html

OHSUMED. OHSUMED[2] corpus was compiled by [8]. Ohsumed-23 is a subset of the original collection, with documents classified to one out of 23 classes from the Heart Diseases subtree of the MeSH classification scheme. Ohsumed-96 contains 96 classes, those having at least one document in the training and one in the testing set. Ohsumed-96 was further divided into two subsets. Ohsumed-49 that contains the largest classes, those with at least 75 documents in the training set and ohsumed-28 contains 28 classes with 15 to 74 documents each in the training set. The subsets with the suffix -x contain documents only from the corresponding categories.

Reuters-RCV1. Reuters-RCV1[3] corpus [16] is a new generation Reuters corpus with multi-labeled that contains 103 classes.

TREC-AP. TREC-AP[4] corpus contains 209,783 documents classified into 20 classes. (142,791 are used for training and 66,992 for testing).

Single-label Datasets. We have also experimented with 3 more single-label datasets. NG-20, with 20 classes of Newsgroups[5] [13], with 9840 documents for training and 6871 for testing. Two more single-label dataset were derived from the Reuters-21578 corpus. Reuters-8 is the subset of Reuters-10 with single label documents only, and similarly Reuters-52 is the subset of Reuters-90.

Artificially Generated Datasets. Several linearly separable datasets were generated using MATLAB scripts for various dimensions. For each dimension-value datasets of 100, 1000 and 10000 instances were created. In total there are 18 artificially generated linear separable datasets. In our notation, 2-dim-1000-inst stand for a 2-dimensional space and 1000 instances.

4.2 Evaluation Procedure

Experiments were run on the same machine with the same initial conditions, i.e. the same initial weight vector $\overrightarrow{W}^{(0)}$ (Eq. 7) and the same bias value b (Scut value). For Perceptron a fixed learning rate was used ($\alpha = 1$). Our main goal was to compare the cost for the training phase (in terms of training iterations) for the Perceptron and the proposed algorithm. In the case where the datasets are not linearly separable, training is stopped after a predefined number of iterations and the best hyperplane (in terms of the F_1 evaluation measure) is retained. The artificially generated datasets, are shortage from a testing set.

Finally a comparison of the efficiency (as measured by AUC) is applied between the proposed algorithm and the SVM. We used the SVMPerf [11].

Evaluation Measures. In both training and testing phase the microaveraged F_1 and the AUC (Area Under the ROC Curve) were used.

[2] ftp://medir.ohsu.edu/pub/ohsumed/
[3] http://www.daviddlewis.com/resources/testcollections/rcv1/
[4] http://www.daviddlewis.com/resources/testcollections/trecap/
[5] http://kdd.ics.uci.edu/databases/20newsgroups/20newsgroups.html

5 Discussion of Experimental Results

Table 2 illustrates the average number of iterations (over all binary problems of each dataset) and the micro averaged F_1 measure of all classes of each dataset). In same Table is also presented the performance of the classifiers on the testing phase. The performance is measured by the AUC score macro-averaged over all classes in each dataset. In the last column of 2 results from an SVM classifier on the same datasets are presented.

Table 2. Performance of the Perceptron and Proposed algorithm of the training an testing phase

	Training Phase				Testing Phase		
Dataset Name	Avg Iterations		microF_1 Score		macro AUC Score		
	Perceptron	Proposed	Perceptron	Proposed	Perceptron	Proposed	SVM
Reuters-10	462	136	0.9933	0.9976	0.9721	0.9719	0.9940
Reuters-90	318	28	0.9843	0.9979	0.9793	0.9845	0.9011
Reuters-10-x	434	150	0.9968	0.9984	0.9745	0.9744	0.9970
Reuters-90-x	300	32	0.9899	0.9983	0.9834	0.9842	0.9138
Reuters-RCV1	455	37	0.8845	0.9918	0.9385	0.9418	0.9491
Ohsumed-23	340	14	0.9423	1.0	0.9347	0.9326	0.9196
Ohsumed-96	362	19	0.8122	1.0	0.9688	0.9907	0.7218
Ohsumed-49	454	30	0.8235	1.0	0.9769	0.9846	0.9289
Ohsumed-28	228	11	0.6983	1.0	0.9411	0.9985	0.4810
Ohsumed-96-x	309	12	0.9101	1.0	0.9681	0.9747	0.9267
Ohsumed-49-x	317	18	0.9155	1.0	0.9516	0.9594	0.9436
Ohsumed-28-x	57	5	1.0	1.0	0.9792	0.9786	0.9685
Trec-20	483	59	0.8963	0.9997	0.9847	0.9885	0.9922
NG-20	395	89	0.9989	0.9996	0.9792	0.9802	0.9766
Reuters-8	339	85	0.9984	1.0	0.9846	0.9846	0.9980
Reuters-52	251	27	0.9961	0.9993	0.9855	0.9855	0.9402

The rate of convergence on the artificially generated datasets is illustrated by the number of iterations each method takes to converge. In all the experiments the initial conditions were used. Figure 3 summarizes the results from these datasets. The y-axis indicates the number of iterations for convergence and the x-axis represents the 18 artificial data sets.

Considering the results of Table 2, three main conclusions can be derived.

First, both the perceptron and the proposed algorithm, perform at least as good (if not better) as the state-of-the-art method SVM, on new unseen data.Measurements of the AUC score on testing datasets show that they perform extremely well, at least on those text datasets we evaluated them on.Out of the 16 text datasets we tested, SVM outperforms on 5 of them, perceptron on 3 of them and the proposed algorithm on 9 of them. However, their differences

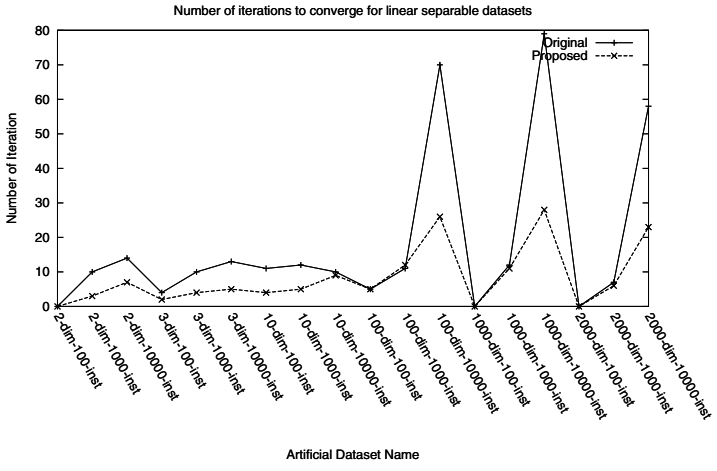

Fig. 3. Convergence results of perceptron and proposed algorithm on artificially generated datasets

on the AUC score are minor except on the Ohsumed dataset were the proposed algorithm significantly outperforms both of the other methods.

Second, the proposed algorithm finds a perfect or a better separating hyperplane more often than the original fixed increment batch perceptron. During the training phase of the text datasets, the proposed method achieved better micro-averaged F_1 score on all 16 datasets. Indeed on 7 of the text datasets the algorithm found a perfect separating hyperplane (that is why micro$F_1 = 1.0$).

Third, the proposed method converges much faster than the original perceptron. This is shown by the average number of iteration each method needs to converge.

As far for the linearly separable artificial datasets ($F_1 = 1$ at the training phase), see Figure 3, we may conclude that: On spaces of high dimensionality (thousands dimensions) and large datasets the proposed method needs significantly less iterations than the original perceptron. On spaces of low dimensionality (less than 100 dimensions), and small datasets both methods require about the same number of iteration to converge. However in latter case and with large datasets then the proposed method needs slightly less iterations. This is an interest case for further investigation using kernel functions to accelerate convergence.

6 Conclusions

In this paper, an algorithm,introduced in [6], was presented for training a classifier, without the need of using an arbitrary value or prior estimated learning rate parameter. A proof of convergence was given. Results from training text

classifiers and from artificially generated linear separable datasets, show that the method is both fast and accurate. In concluding we summarize that:

- The proposed method is very simple and easy to implement and operate.
- It is computationally cheap and experimental results show that it is very fast,
- It can be used in any domain and any data without any modification or parameter estimation.
- It converges after a finite number of steps to a separating hyperplane in the case of linearly separable data.

Finally, we note that there is a lot of space for further research of the proposed algorithm, which is currently under investigation, in order to find the best classifier, the one with maximal margins from the datasets as well as to improve convergence for low-dimension data using kernel functions.

References

1. Buckley, C., Salton, G.: Optimization of relevance feedback weights. In: SIGIR 1995: Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 351–357. ACM, New York (1995)
2. Cristianini, N., Shawe-Taylor, J.: An Introduction To Support Vector Machines (and other kernel-based learning methods). Cambridge University Press, Cambridge (2000)
3. Dagan, I., Karov, Y., Roth, D.: Mistake-driven learning in text categorization. In: 2nd Conference on Empirical Methods in Natural Language Processing, EMNLP 1997, pp. 55–63 (1997)
4. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification, 2nd edn. Wiley Interscience, Hoboken (November 2000)
5. Dumais, S., Platt, J., Heckerman, D., Sahami, M.: Inductive learning algorithms and representations for text categorization (1998)
6. Gkanogiannis, A., Kalampoukis, T.: A modified and fast perceptron learning rule and its use for tag recommendations in social bookmarking systems. In: ECML PKDD Discovery Challenge 2009 - DC 2009 (2009)
7. Harman, D.: Relevance feedback and other query modification techniques, pp. 241–263 (1992)
8. Hersh, W., Buckley, C., Leone, T., Hickman, D.: Ohsumed: an interactive retrieval evaluation and new large test collection for research (1994)
9. Joachims, T.: Text categorization with support vector machines: learning with many relevant features (1998)
10. Joachims, T.: Making large-scale support vector machine learning practical. In: Schölkopf, B., Burges, C.J.C., Smola, A.J. (eds.) Advances in Kernel Methods: Support Vector Learning, pp. 169–184. MIT Press, Cambridge (1999), http://portal.acm.org/citation.cfm?id=299104
11. Joachims, T.: Training linear svms in linear time. In: KDD 2006: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 217–226. ACM, New York (2006), http://dx.doi.org/10.1145/1150402.1150429

12. Karypis, G., Shankar, S.: Weight adjustment schemes for a centroid based classifier (2000)
13. Lang, K.: Newsweeder: learning to filter netnews (1995)
14. Lewis, D.D.: Evaluating text categorization. In: Workshop on Speech and Natural Language HLT 1991, pp. 312–318 (1991)
15. Lewis, D.D., Schapire, E.R., Callan, P.J., Papka, R.: Training algorithms for linear text classifiers. In: 19th ACM International Conference on Research and Development in Information Retrieval SIGIR 1996, pp. 298–306 (1996)
16. Lewis, D.D., Yang, Y., Rose, T., Li, F.: Rcv1: A new benchmark collection for text categorization research. Journal of Machine Learning Research 5, 361–397 (2004)
17. Novikoff, A.B.: On convergence proofs for perceptrons. In: Proceedings of the Symposium on the Mathematical Theory of Automata, vol. 12, pp. 615–622 (1963), http://citeseer.comp.nus.edu.sg/context/494822/0
18. Porter, M.F.: An algorithm for suffix stripping. Program 14(3), 130–137 (1980)
19. Rosenblatt, F.: The perceptron: a probabilistic model for information storage and organization in the brain. Psychological Review 65(6), 386–408 (1958)
20. Salton, G.: Automatic Text Processing – The Transformation, Analysis, and Retrieval of Information by Computer. Addison-Wesley, Reading (1989)
21. Schapire, R.E., Singer, Y., Singhal, A.: Boosting and rocchio applied to text filtering. In: SIGIR 1998: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 215–223. ACM, New York (1998)
22. Sebastiani, F.: Machine learning in automated text categorization. ACM Computing Surveys 34(1), 1–47 (2002)
23. Yang, Y.: A study on thresholding strategies for text categorization (2001)