Original Article
International Journal of Fuzzy Logic and Intelligent Systems
Vol. 16, No. 2, June 2016, pp. 81-86
http://dx.doi.org/10.5391/IJFIS.2016.16.2.81

ISSN(Print) 1598-2645
ISSN(Online) 2093-744X

Robust Algorithms for Combining Multiple Term Weighting Vectors for Document Classification

Minyoung Kim
Department of Electronics & IT Media Engineering, Seoul National University of Science & Technology, Seoul, Korea

ljfis

Abstract

Term weighting is a popular technique that effectively weighs the term features to improve accuracy in document classification. While several successful term weighting algorithms have been suggested, none of them appears to perform well consistently across different data domains. In this paper we propose several reasonable methods to combine different term weight vectors to yield a robust document classifier that performs consistently well on diverse datasets. Specifically we suggest two approaches: i) learning a single weight vector that lies in a convex hull of the base vectors while minimizing the class prediction loss, and ii) a mini-max classifier that aims for robustness of the individual weight vectors by minimizing the loss of the worst-performing strategy among the base vectors. We provide efficient solution methods for these optimization problems. The effectiveness and robustness of the proposed approaches are demonstrated on several benchmark document datasets, significantly outperforming the existing term weighting methods.

Keywords: Machine learning, Document/text classification, Term weighting, Optimization

1. Introduction

Increased availability of a large volume of documents data generated from social networking, Internet blogs, and news articles among others, naturally demands automatic and accurate tagging programs that unburden the human efforts to read huge amount of data. In turn, the document classification emerges rapidly, and now is recognized as one of the most important problems in machine learning and artificial intelligence recently. Popular applications include: sentiment analysis [1], email filtering [2, 3], news article clusterings [4, 5], to name just a few. The first step in text classification is how to represent a document, comprised of several sentences/paragraphs each of which is a sequence of a varying number of words. One of the most successful representations is the *bag-of-words (BoW)*, a vector defined to be of the vocabulary size V (i.e., the number of all distinct words considered), where the i-th entry has the counts of the term i in the document. Thus it is often called the *term-frequency (tf)* representation of a document. While the *tf* representation ignores the spatial information of the words (i.e., words ordering in sentences/paragraphs), it contains valuable statistical information about the document, namely which terms are used and which are not in the document, and how often certain terms occur in a document.

Received: Apr. 3, 2016
Revised : Jun. 17, 2016
Accepted: Jun. 20, 2016

Correspondence to: Minyoung Kim
(mikim@seoultech.ac.kr)

While one can use *tf* vectors directly in classifier training, it is often more effective to weigh individual terms differently. The motivation is that some terms are more important and salient than others with regard to distinguishing documents by topics or contents. For instance, word counts of certain stop words (e.g., articles *a* or *the*) are apparently considered as noise having little impacts on document classification. The most popular but traditional term weighting scheme is the so-called *inverse document frequencies (idf)*, where each term frequency is weighted by the *idf* which is a reciprocal of the number of training documents that contain the term. Intuitively, *idf* values tend to be high for those terms that occur exclusively in a small number of documents (more likely keywords), while ubiquitous terms like stop words would get low weights.

Beyond the success of *tf-idf*, recently several *supervised* term weighting schemes have been proposed where the idea is to exploit the class labels in learning the weight vector. The *tf-idf* is clearly an unsupervised approach for its ignorance of class labels at all, and often outperformed by the supervised methods. These methods typically use certain information-theoretic statistics such as χ^2 metric, information gain, and odds ratio [6, 7]. More recently the *tf-rf* strategy [8] aimed to weigh each term frequency by the so-called *relevance frequency* (*rf*), the ratio between the numbers of positive and negative documents that contain the term. Terms with *rf* values highly distant from 1 can thus be considered as more discriminative indicators for classification.

Given that we have a handful of existing term weighting algorithms, the question is which one is the best. It seems like that there is no single best scheme that outperforms the rest across all problems/datasets. Rather, one strategy performs well on some specific cases while fails severely in other situations, meaning that individual weight vectors may have their own best working regimes. This issue of deficiency in robustness originates from using a single term weighting strategy. Motivated by this, in this paper we propose several reasonable methods to combine multiple different term weight vectors to yield a robust document classifier that performs well broadly across different types of datasets. We denote the existing term weighting vectors (listed and summarized in Sec. 2) by *base weight vectors*.

We suggest two approaches of combining base vectors: i) Learning a single term weight vector that is a mixture of the base vectors. Basically we enforce the target weight vector to minimize the class prediction loss, while the target vector is confined to lie in a convex hull of the base vectors. ii) Learning a worst-case robust classifier that minimizes the loss of the worst-performing base weight vector. This method does not find a single term weight vector, but a *classifier* that takes into account all base vectors as input and tries to reduce the worst-case prediction error.

Specifically, we consider the linear support vector machine (SVM) as an underlying classifier, and formulate optimization problems in principled manners. Among other classifiers, the SVM's maximization of the data margins in the class decision boundary results in theoretical performance guarantee, also practically superb prediction accuracy [9]. Interestingly in the second approach, the proposed mini-max optimization problem becomes an instance of convex optimization. We demonstrate the effectiveness and robustness of the proposed approaches for several benchmark document datasets.

1.1 Notations and Problem Setup

We deal with the supervised learning setup where we have n training samples of document-class pairs, $\mathcal{D} = \{(d_i, y_i)\}_{i=1}^n$. We assume binary classification (i.e., $y \in \{+1, -1\}$) where multi-class problems can be converted to binary straightforwardly via standard treatments (e.g., one-vs-others in [10]). The vocabulary is defined to be a set of all terms (or words) in the training corpus $\mathcal{V} = \{t_k\}_{k=1}^V$ where we let V be the size of the vocabulary.

In the *tf* representation of a document d, denoted by a V-dim vector $\mathbf{tf}(d) = [tf_1, tf_2, \ldots, tf_V]^\top$, where tf_k contains word counts of the term t_k in d. Instead of using $\mathbf{tf}(d)$ directly as a feature vector for d in classification, the term weighting methods construct a feature vector $\mathbf{x}$ by multiplying tf values by the term weights, namely

$$\mathbf{x}(d) = \mathbf{tf}(d) \otimes \mathbf{b}, \ \ (\otimes\text{: element-wise product}), \qquad (1)$$

where $\mathbf{b} = [b_1, b_2, \ldots, b_V] \in \mathbb{R}^V$ is the base term weight vector. Several schemes of defining $\mathbf{b}$ have been introduced thus far, and we briefly summarize them in the following section.

2. Existing Term Weighting Schemes

In the term weighting schemes the importance of each term is judged and quantified, typically using statistics about the term in the training documents. It is convenient to define specific schemes using the (2×2) contingency tables, one for each term t_i. Specifically, we let p_i^1 be the number of positive documents in the training set that contain the term t_i, and n_i^0 be the number of negative training documents where t_i is absent. Similarly, we

define p_i^0 and n_i^1. Clearly, the numbers of positive and negative documents are: $n_+ = p_i^1 + p_i^0$, $n_- = n_i^1 + n_i^0$ for all i (and $n = n_+ + n_-$). We list the popular base weight vectors below with a brief comment or two.

- *idf*: Inverse document frequency; $b_i = \log(\frac{n}{p_i^1 + n_i^1})$, which is an unsupervised scheme.
- *chi2*: The χ^2-statistics based term weighting;

$$b_i = \frac{n(p_i^1 n_i^0 - p_i^0 n_i^1)^2}{n_+ n_- (p_i^1 + n_i^1)(p_i^0 + n_i^0)},$$

  which is a supervised scheme.
- *ig*: The information-gain term weighting;

$$b_i = \frac{p_i^1}{n}\log(\frac{(n/n_+)p_i^1}{p_i^1 + n_i^1}) + \frac{p_i^0}{n}\log(\frac{(n/n_+)p_i^0}{p_i^0 + n_i^0}) + \frac{n_i^1}{n}\log(\frac{(n/n_-)n_i^1}{p_i^1 + n_i^1}) + \frac{n_i^0}{n}\log(\frac{(n/n_-)n_i^0}{p_i^0 + n_i^0}),$$

  which is a supervised scheme.
- *rf*: The relevance frequency term weighting; $b_i = \log(2 + \frac{p_i^1}{n_i^1})$, which considers the ratio between the numbers of positive and negative documents that contain the term.
- *or*: The odds-ratio term weighting; $b_i = \frac{p_i^1 n_i^0}{p_i^0 n_i^1}$, which is similar to *rf*, but also takes into account absence of the term.

3. Combining Base Term Weight Vectors

From the previous section, we denote by $\mathbf{b}^{(j)}$ the j-th base term weight vectors ($j = 1, 2, \ldots, J$); for instance, $\mathbf{b}^{(1)}$ is the *idf*, $\mathbf{b}^{(2)}$ is the *chi2*, and so on. In what follows we suggest two reasonable and robust methods of combining base vectors.

3.1 Learning a Single Weight Vector

The key idea is to find the single term weight vector that is not distant from every base vector while achieving the smallest classification error. To impose similarity between the target term weight vector and base vectors, we specifically consider to constrain the target vector to lie in the convex hull of the base vectors. Denoting the target vector to be learned as $\mathbf{b}$, we impose $\mathbf{b} = \sum_{j=1}^J \alpha_j \mathbf{b}^{(j)}$ where $\sum_{j=1}^J \alpha_j = 1$ and $\alpha_j \geq 0$. This can be seen as a mixture of base vectors, and essentially parametrizes $\mathbf{b}$ by mixing parameters α_j's. Then we solve:

$$\min_{\{\alpha_j\},\mathbf{w}} \sum_{i=1}^n l(\mathbf{x}_i, y_i; \mathbf{b}, \mathbf{w}) + C||\mathbf{w}||^2 \quad (2)$$
$$\text{s.t. } \mathbf{b} = \sum_{j=1}^J \alpha_j \mathbf{b}^{(j)}, \ \sum_{j=1}^J \alpha_j = 1, \ \alpha_j \geq 0,$$

where C is the non-negative constant that regularizes a non-smooth classifier. Here we incorporate the SVM's hinge loss, $l(\mathbf{x}, y; \mathbf{b}, \mathbf{w}) = (1 - y\mathbf{w}^\top(\mathbf{b} \otimes \mathbf{x}))_+$, where $(z)_+ = \max(z, 0)$.

While the optimization (2) is non-convex *jointly* in the mixing variables $\{\alpha_j\}$ and the classifier parameters $\mathbf{w}$, it becomes convex for each set when fixing the other as constant. Hence we solve (2) by a (block) coordinate-wise minimization fashion, meaning that we alternate (until convergence) two steps: i) fixing $\mathbf{w}$ and solve the convex minimization over $\{\alpha_j\}$, and ii) fixing $\{\alpha_j\}$ and solve the SVM problem for $\mathbf{w}$ using the features $\mathbf{b} = \sum_{j=1}^J \alpha_j \mathbf{b}^{(j)}$. The final solution $\mathbf{w}$ and $\mathbf{b}$ can be used at test time directly, that is, $y = \text{sgn}(\mathbf{w}^\top(\mathbf{b} \otimes \mathbf{x}))$.

3.2 Mini-Max Robust Classifier Learning

The approach in the previous section takes into account all the base vectors in learning the final term weight vector. This can be intuitively appealing, but it can still be less robust since the final model is comprised of a *single vector*, which can be potentially sensitive to specifics of particular problems/datasets and prone to overfit.

To be more robust and improve generalization performance, we consider to learn a classifier (instead of learning a single fixed weight vector) that can be applied to all different base vectors. When training the classifier, we deal with the worst-performing base vector for each instance, and try to reduce its worst-case loss function. At test time, however, we take the majority vote over predictions from different base vectors. This strategy can be beneficial for not only increasing robustness (by worst-case treatment over the base vectors), but also improving the generalization performance (by voting over different term weighting schemes). Our idea can be formulated as a mini-max optimization:

$$\min_{\mathbf{w}} \sum_{i=1}^n \Big[\max_{1 \leq j \leq J} l(\mathbf{x}_i, y_i; \mathbf{b}^{(j)}, \mathbf{w})\Big] + C||\mathbf{w}||^2. \quad (3)$$

Interestingly, (3) is an instance of convex optimization, since the summand in the objective can be written as: $(\max_j(1 - y_i\mathbf{w}^\top(\mathbf{b}^{(j)} \otimes \mathbf{x}_i)))_+$, which is point-wise maximum of the

affine functions. However, due to the non-differentiability of the objective function (from the max function), we use the subgradient descent method [11], namely, $\mathbf{w} \leftarrow \mathbf{w} - \eta(2C\mathbf{w} + \sum_i \partial l_i)$, where $\eta > 0$ is the learning rate, and ∂l_i is the subgradient of $\max_{1 \le j \le J} \; l(\mathbf{x}_i, y_i; \mathbf{b}^{(j)}, \mathbf{w})$:

$$\partial l_i = \begin{cases} -y_i(\mathbf{b}^{(j^*(i))} \otimes \mathbf{x}_i) & \text{if } p_i(\mathbf{w}) < 1 \\ 0 & \text{if } p_i(\mathbf{w}) \ge 1 \end{cases} \tag{4}$$

where $p_i(\mathbf{w}) = y_i \mathbf{w}^\top (\mathbf{b}^{(j^*(i))} \otimes \mathbf{x}_i)$ and $j^*(i) = \arg\min_j y_i \mathbf{w}^\top (\mathbf{b}^{(j)} \otimes \mathbf{x}_i)$. Once the optimal classifier $\mathbf{w}$ is found, at test time the class label is determined by majority vote over predictions using the J base weight vectors, that is, $y = \text{majority-vote}_j (\text{sgn}(\mathbf{w}^\top (\mathbf{b}^{(j)} \otimes \mathbf{x})))$.

4. Experiments

We evaluate the prediction performance of the proposed approaches on several benchmark document classification datasets. The details and the experimental setups about the datasets are described in Sec. 4.1. As mentioned, we train a linear SVM classifier for a given feature weighting vector (either a base vector or our mixed term weight vector estimated from Sec. 3.1). For the mini-max classifier learning (Sec. 3.2), we directly solve the optimization (3). The hyperparameter (i.e., the loss-regularization trade-off parameter C) is chosen by cross validation on a held-out portion of the training set. We list below the competing approaches (with abbreviations) described in the previous sections.

- ***xxx***: Here *xxx* indicates one of the five base weight vectors discussed in Sec. 2 (*idf*, *chi2*, *ig*, *rf*, and *or*).
- ***mix-ch***: Our convex-hull mixture of the five base vectors as described in Sec. 3.1. We repeat alternating the block coordinate descent steps until convergence (e.g., the norm of the relative changes in the consecutive iterates becomes smaller than 10^{-4}).
- ***mini-max***: Our mini-max robust classifier in Sec. 3.2. In the subgradient update step, we use the diminishing step size (e.g., $\eta_k = 1/k$ at the k-th iteration).

4.1 Datasets

The competing approaches are tested and contrasted on three benchmark datasets in text classification: Reuters-21578, WebKB, and 20 newsgroup. We pre-process the data using the famous Porter's Stemmer algorithm [12] for *term stemming* which essentially removes derived words from a stem word (e.g., *fishing*, *fisher*, and *fished* replaced by the same root word *fish*). Details and experimental setups for each dataset are described in the following.

4.1.1 Reuters

The dataset[11] consists of documents on the Reuters newswire in 1987, and among several variants we use the $R8$ dataset comprised of documents from 8 classes including *crude*, *earn*, *ship*, and so on. Overall there are about 7,000 instances where the corpus has approximately 16,000 distinct terms. We split the data randomly into $50\%/50\%$ training/test sets, and this is repeated for 10 random runs to report the average performance. Each multi-class problem is transformed into 8 different binary classification problems by the one-vs-others treatment.

4.1.2 WebKB

The dataset[22] is collected by the world-wide knowledge base project of the CMU text learning group, containing approximately 4,000 web pages from various universities in US. The web pages are manually labeled as one of the four topcis (classes): *student*, *faculty*, *course*, and *project*. The vocabulary size is about 8,000. We use the similar training/test set split protocol as the Reuters set.

4.1.3 20 Newsgroups

This large-scale dataset[33] consists of about 20,000 newsgroup documents annotated as 20 different subject categories. Some class categories are hierarchically related, for instance, the *science* class has *crypt* and *electronics* as its children. This introduces some degree of ambiguity, which in turn can make accurate prediction more difficult. However, some other categories are inherently unrelated from others (e.g., *guns* under *politics*). The whole dataset has about 18,000 documents where we randomly partition them into training/test sets with equal numbers. Again the 20-way classification problems are converted into 20 binary problems by the one-vs-others transformation. The corpus is comprised of about 70,000 unique terms after term stemming.

[11] http://www.daviddlewis.com/resources/testcollections/reuters21578/.
[22] http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/.
[33] http:// people.csail.mit.edu/jrennie/20Newsgroups/.

Table 1. Test prediction F-scores on the Reuters dataset

Methods	Micro F-score	Macro F-score
idf	0.7942 ± 0.0054	0.7950 ± 0.0050
chi2	0.8346 ± 0.0055	0.8349 ± 0.0054
or	0.8198 ± 0.0061	0.8207 ± 0.0064
ig	0.8328 ± 0.0046	0.8333 ± 0.0044
rf	0.8453 ± 0.0073	0.8460 ± 0.0071
mix-ch	0.9132 ± 0.0050	0.9090 ± 0.0077
mini-max	0.9087 ± 0.0076	0.9136 ± 0.0052

Table 2. Test prediction F-scores on the WebKB dataset

Methods	Micro F-score	Macro F-score
idf	0.8129 ± 0.0101	0.8132 ± 0.0116
chi2	0.8984 ± 0.0033	0.8989 ± 0.0071
or	0.8512 ± 0.0050	0.8520 ± 0.0073
ig	0.8982 ± 0.0029	0.8990 ± 0.0070
rf	0.8629 ± 0.0033	0.8617 ± 0.0023
mix-ch	0.9012 ± 0.0035	0.9046 ± 0.0011
mini-max	0.9190 ± 0.0040	0.9219 ± 0.0046

4.2 Results

As a performance measure for the competing classifiers, we use the F-score, a standard metric for information retrieval tasks. The accuracy (i.e., the proportion of the correctly predicted instances) alone is not sufficient due to potentially high imbalance in the amount of positive and negative documents, and one should deal with both precision and recall scores. In the binary classification, the precision (p) is defined as the proportion of the true positives (TP) out of the predicted positives (PP), i.e., $p = \frac{TP}{PP}$, while the recall (r) is defined as $r = \frac{TP}{WP}$ where WP is the number of whole positive documents in the test set. Then one defines F-score as $\frac{2pr}{p+r}$.

Since the datasets we test with have all multi-class setups which have been binarized via one-vs-others transformation, one needs a way to combine F-score measures for individual problems. We basically follow the averaged F-scores introduced in [8], specifically: i) micro-averaged F-score defined as $\frac{2PR}{P+R}$ where $P = \frac{\sum_c TP(c)}{\sum_c PP(c)}$ and $R = \frac{\sum_c TP(c)}{\sum_c WP(c)}$ with $TP(c)$, PP(c), and $RP(c)$ from the c-th binary problem, and ii) macro-averaged F defined to be $\frac{2P'R'}{P'+R'}$ $P' = (1/K)\sum_c p(c)$ and $R' = (1/K)\sum_c r(c)$ where $p(c)$ and $r(c)$ are the precision and the recall for the c-th problem.

The prediction results are summarized in Table 1–3. We depict the F-scores averaged over 10 random folds with standard deviations. The proposed methods consistently outperform the base term weighting schemes. Also, it turns out that the individual base term weight vectors alone exhibit less robust prediction: some schemes perform well on one dataset while being inferior to others in other datasets. For instance, the *rf* scheme performs best among other base vectors on the Reuters dataset, while worse on the WebKB. The proposed algorithms of combining base weighting vectors can yield consistently superior prediction performance across different datasets.

Table 3. Test prediction F-scores on the 20-Newsgroup.

Methods	Micro F-score	Macro F-score
idf	0.5575 ± 0.0063	0.5582 ± 0.0055
chi2	0.5732 ± 0.0037	0.5743 ± 0.0036
or	0.5928 ± 0.0059	0.5932 ± 0.0054
ig	0.5524 ± 0.0216	0.5685 ± 0.0095
rf	0.5958 ± 0.0058	0.5961 ± 0.0053
mix-ch	0.6613 ± 0.0005	0.6595 ± 0.0025
mini-max	0.6742 ± 0.0004	0.6622 ± 0.0009

5. Conclusion

We have proposed two robust approaches to combining base term weight vectors that yield superior document classification performance consistently across diverse datasets. The first method finds the best mixing of the base vectors to minimize the overall classification errors, while the second one, the mini-max classifier, aims at improving the performance of the worst-performing base weighting vector to yield a more robust predictor. On several benchmark text datasets, our approaches exhibit robustness and consistently superior performance to the existing term weighting methods.

Conflict of Interest

No potential conflict of interest relevant to this article was reported.

Acknowledgments

This study was supported by Seoul National University of Science & Technology.

References

[1] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," 2011. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*.

[2] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz, "A bayesian approach to filtering junk e-mail," 1998. *Proceedings of the 21st national conference on Artificial intelligence (AAAI)*.

[3] Upasana and S. Chakravarty, "A survey on text classification techniques for e-mail filtering," 2010. *International Conference on Machine Learning and Computing*.

[4] T. Joachims, "Text categorization with suport vector machines: Learning with many relevant features," 1998. *European Conference on Machine Learning*.

[5] A. Chy, M. Seddiqui, and S. Das, "Bangla news classification using naive Bayes classifier," 2014. International Conference on Computer and Information Technology.

[6] F. Debole and F. Sebastiani, "Supervised term weighting for automated text categorization," 2003. *Proceedings of the ACM symposium on Applied computing*.

[7] Z.-H. Deng, S.-W. Tang, D.-Q. Yang, M. Z. L.-Y. Li, and K.-Q. Xie, "A comparative study on feature weight in text categorization," *Advanced Web Technologies and Applications, Lecture Notes in Computer Science*, vol. 3007, pp. 588–597, 2004.

[8] M. Lan, C. Tan, J. Su, and Y. Lu, "Supervised and traditional term weighting methods for automatic text categorization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 4, pp. 721–735, 2009.

[9] V. N. Vapnik, *The Nature of Statistical Learning Theory*. Springer, 1995.

[10] K. Crammer, Y. Singer, N. Cristianini, J. Shawe-taylor, and B. Williamson, "On the algorithmic implementation of multiclass kernel-based vector machines," *Journal of Machine Learning Research*, vol. 2, 2001.

[11] D. P. Bertsekas, *Nonlinear Programming*. Athena Scientific, Belmont, MA, 1999.

[12] M. F. Porter, "An algorithm for suffix stripping," *Program*, vol. 14, no. 3, pp. 130–137, 1980.

Minyoung Kim received his B.S. and M.S. degrees both in computer science and engineering from Seoul National University, South Korea. He earned a Ph.D. degree in computer science from Rutgers University in 2008. From 2009 to 2010 he was a postdoctoral researcher at the Robotics Institute of Carnegie Mellon University. He is currently an assistant professor in the Department of Electronics and IT Media Engineering at Seoul National University of Science and Technology in Korea. His primary research interest is machine learning and computer vision. His research focus includes graphical models, motion estimation/tracking, discriminative models/learning, kernel methods, and dimensionality reduction.