Sparsification of Linear Models for Large-Scale Text Classification

Simon Moura*[1,2], Ioannis Partalas†[1], and Massih-Reza Amini‡[2]

[1]Viseo R.&D.
[2]University of Grenoble Alpes

Abstract

In this paper we propose a simple yet effective method for sparsifying a posteriori linear models for large-scale text classification. The objective is to maintain high performance while reducing the prediction time by producing very sparse models. This is especially important in real-case scenarios where one deploys predictive models in several machines across the network and constraints apply on the prediction time. We empirically evaluate the proposed approach in a large collection of documents from the Large-Scale Hierarchical Text Classification Challenge. The comparison with a feature selection method and LASSO regularization shows that we achieve to obtain a sparse representation improving in the same time the classification performance.

Keywords: large-scale text classification, sparsification, feature selection.

1 Introduction

In large-scale scenarios like for example, ad-click prediction, text and gene classification, much attention has been given to the deployment of linear models, mostly due to their simplicity and efficiency. In such scenarios, the vector representation of data is often sparse and the size of the feature space exceeds the size of the available training examples. Also, in several applications constraints in both space and test-time prediction may apply making cumbersome the maintenance of large models. For instance, for the DMOZ dataset of the LSHTC challenge [PKB+15] which contains over 27,000 classes and over half a million of features a linear model would require approximately 123 Gb of memory. Besides this, many features in such datasets are correlated or uninformative and can harm the performance of a predictive model.

For reducing the size of the feature space and remove irrelevant attributes one may rely to feature selections algorithms as described by Guyon et al.[GE03]. An effective method for feature selection is Recursive Feature Elimination which uses an estimator (for example a Support Vector Machine (SVM)) in order to assign weights to the features. Progressively, the method selects and evaluates subsets of features. It is evident that such wrapper methods are costly for large-scale cases as the performance of the model has to be evaluated repeatedly. Beside this, a popular method for reducing memory requirements is the use of L_1-norm as penalization term (also known as LASSO), which induces sparsity to the model [Tib94]. Koshiba et al.[KA03] demonstrates after extensive experiments on multiple datasets that L_1-loss SVM gives very sparse models with accuracy results close to the one obtained with L_2-loss SVM.

However in applications, like in text classification, the size of the feature space exceeds the number of available examples and the features are correlated; in these cases the performance in terms of accuracy of LASSO is dominated by the use of L_2-norm [Tib94, ZH05, AAG+11]. But, L_2-norm produces dense solutions which cannot scale well in large classification problems.

In this work we consider large-scale text classification (LSTC) tasks. We target a method that should:

- Provide a similar or better performance to L_2-norm regularization.
- Produce very sparse models.

For achieving this we propose a simple approach for sparsifying linear models, a posteriori, learned with L_2-norm regularization. The proposed method achieves to produce very sparse models which reduces the required storage space and improves inference time. This is especially important in large-scale problems where

*simon.moura@viseo.com
†ioannis.partalas@viseo.com
‡massi-reza.amini@imag.fr

predictive models are deployed in several machines and constraints apply on prediction time. While, such rounding techniques are not well suited for on-line methods [LLZ09] we find that in the case of batch methods and text applications are effective.

We empirically evaluate the proposed approach on a large dataset from the LSHTC competition achieving not only to produce sparse models reducing thus memory requirements, but also to improve the predictive performance. We also highlight important factors for sparse models in text classification by analyzing the obtained solutions.

In Section 2, we position our work with respect to the state of the art. In Section 3, we then present the a posterior pruning approach we propose for LSTC. In Section 4, we present experimental results obtained with our approach on a subset of the DMOZ dataset. Finally, in Section 5 we discuss the outcomes of this study and give some pointers to further research.

2 Related Work

Bi et al. [BBE+03] propose a bootstrap method for selecting variables by constructing a series of linear SVMs and eliminating, after performing a linear combination of the classifiers, the variables for which the weight values do not exceed a certain threshold. Langford et al. [LLZ09] introduces a method, called truncated gradient, which modifies the gradient rule in the standard stochastic gradient descent algorithm. The idea is to shrink the weights that are smaller than a predefined threshold gradually. Our method works in a similar manner but we focus on batch learning methods while inducing sparsity in the model a posteriori as we focus mainly in improving prediction time. Bolasso [Bac08] is a bootstrapped version of Lasso which selects the intersection of the set of variables selected by several replications of Lasso using bootstrap samples.

Aseervatham et al. [AAG+11] propose a sparse version of ridge logistic regression for altering the solution provided by ridge logistic regression. The authors define a strictly convex optimization problem for finding a sparse solution around the ridge solution using L_1 regularization. This method is in-line with our work as it performs a posteriori the sparsification on the learned model. Thresholding techniques have also been studied in the context of wavelets in signal processing. For a thorough treatment of multi-disciplinary sparse methods the interested reader is referred to [MBP14]. A method for features selection for SVMs is introduced in [TWT10]. More specifically, by introducing a binary vector which controls the selection or not of the features, the authors pose a mixed integer programming problem which is further relaxed in order to be efficiently solved.

Golovin et al. [GSMY13] propose a method for projecting the real valued weight vector to a coarse discrete set using randomized rounding for on-line learning methods. The regret analysis show that the accumulated error during learning is small. In a different line, recent work focused on feature hashing for reducing the memory footprint which projects the original feature space to a low dimensional space [WDL+09]. To avoid collisions and thus deteriorate predictive performance the dimension should not be decreased a lot. Finally, Takamatsu [TG14] propose a leverages rounding technique for compressing the data for learning methods. While this method compresses efficiently real-valued data it does not perform any sort of sparsification.

3 A posterior pruning

We consider single-label multi-class classification problems. Let $\mathbf{x} \in \mathbb{R}^d$ represent an input example obtained with the vector space model [SWY75] and $y \in \mathcal{Y} = \{1, \ldots, K\}$ its associated class label. We consider here linear classifiers and the One-Versus-Rest (OVR) approach that is widely used in LSTC tasks ; as the number of classes may be to large not allowing the use of uncombined multi-class approaches like the one proposed by Crammer and Singer [CS02]. The rationale is that in such extreme cases, OVR can be parallelized as the binary problems are supposed to be independent, while the uncombined approaches could not.

To learn the weights vector w, considering the L_2 regularized linear classifiers, we solve the following optimization problem:

$$w^* = \arg\min_w \frac{1}{2} w^T w + C \sum_{i=1}^{m} e(\mathbf{w}; \mathbf{x}_i, y_i)$$

where e denotes the instantaneous loss. Typical cases include the hinge loss or its squared version for SVMs. The ridge solution of this optimization problem is dense and thus cannot be used in large-scale problems were one should handle hundreds of thousands of features. So, we seek a sparse solution close to w^*. One way would be to force to zero weights that are inferior to a certain threshold τ (also known as hard thresholding). But this approach is to aggressive and may deteriorate the performance for bigger values of τ. Note that there is a trade-off between the sparsity that we aim to achieve and the value of the threshold. Bigger values will lead to very sparse models but may hurt the performance

Algorithm 1 Hinge thresholding of a linear classifier.

Require: A linear classifier with weight vector $\mathbf{w}$, thresholds τ and ρ
  for $j = 1 \dots len(w)$ **do**
    if $|w_i| < \tau$ **then**
      if $w_i < 0$ **then**
        $w_i \leftarrow min(0, w_i + \rho)$
      end if
      if $w_i > 0$ **then**
        $w_i \leftarrow max(0, w_i - \rho)$
      end if
    end if
  end for

by removing crucial weights. In order to ease this technique we propose to apply a softer thresholding procedure which sparsifies a posteriori a linear classifier by shrinking linearly the weights inferior to a second threshold ρ. Algorithm 1 presents the thresholding procedure. The intuition is that in text categorization problems while small weight values may correspond to unimportant features many of them may correspond to rare features which is the case for the minority classes containing very few training examples. So, one should avoid discarding these weights.

4 Experiments

This section describes the experiments we conducted in order to evaluate the proposed approach. We focused on large datasets from the text domain and more specifically we used datasets from the LSHTC challenge[1]. We present and discuss the results of the proposed approach in terms of the model sparsity and its predictive performance.

Model sparsity refers to the number of weights that equal a value of zero in the underlying linear model and it is calculated as follows:

$$s = 1 - \frac{\#\text{ of non-zero weights}}{\text{total }\#\text{ of weights}} \quad (1)$$

4.1 Experimental setup

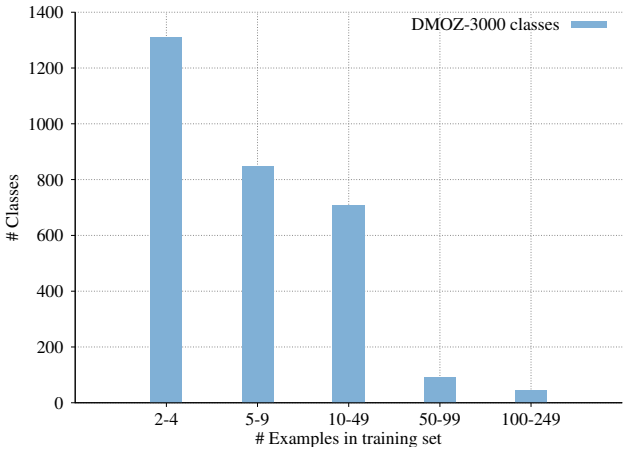
We evaluated the proposed method in a large-scale scenario in multi-class text classification using the 2011 DMOZ dataset of the LSHTC challenge [PKB+15]. This dataset contains 27875 categories, around $394k$ examples and $594k$ features and it is provided in a preprocessed format using stop-word removal and stemming. For each document in the training set the term frequency is provided along with the assigned label. We transformed the tf vectors to the $tf * idf$ representation. We randomly sampled the DMOZ dataset with increasing number of classes.

Table 1 details the important characteristics of the sampled datasets. We note that the number of classes range from 500 to 3000 and the number of features from 68268 to 216545. It has been shown that LSTC collections generally follow a power law distribution; that is a large number of classes contain very few number of examples and most examples are contained in very small number of classes [BMP+14]. Figure 1 presents the class distribution for the 3000 classes dataset. In this case, half of the classes contain less than five examples (1309 classes).

Table 1: The main characteristics (number of classes, features and training instances) of the sample datasets used for the experiments.

Dataset	#Classes	#Features	#Examples
DMOZ-500	500	68268	5818
DMOZ-1000	1000	104768	11123
DMOZ-2000	2000	168153	23346
DMOZ-3000	3000	216545	35533

Figure 1: Class distribution in the DMOZ-3000 dataset. The minority classes are over represented.



As mentioned previously we work with linear classifiers and more specifically we use SVMs as our base model due to its state-of-the-art performance in text classification tasks. In all experiments we used the SVM

[1] http://lshtc.iit.demokritos.gr

Table 2: Number of features kept using χ^2 and grid search. The number in parenthesis represent the percentage of features kept from the original dataset.

# Classes	# Features
500	13600 (19.92%)
1000	20953 (19.99%)
2000	42038 (24.99%)
3000	43309 (20%)

library *LIBLINEAR* to train the linear models with the squared hinge loss function [FCH+08]. We compare the following approaches on each of the samples datasets:

- L_2-SVM: L_2-regularized SVM (ridge penalization).
- L_1-SVM: L_1-regularized SVM (LASSO).
- χ^2: Apply the χ^2 variable selection technique at the top of L_2-SVM. The χ^2 variable selection method has been proved to be very effective for text classification [YP97]. It calculates a contingency table for each term t and class c and then estimates:

$$\chi^2(t,c) = \frac{N \times (AD - CB)^2}{(A+C) \times (B+D) \times (A+B) \times (C+D)}$$

  where A is the number of times t and c co-occur, B is the number of time the t occurs without c, C is the number of times c occurs without t, D is the number of times neither c nor t occurs and N is the total number of documents.

  We obtain the final score for a feature (a term in our case) by averaging its scores across the classes.

  To obtain the best number of features for each datasets, we used a grid-search strategy splitting each training set in two subsets (70%/30%) and validating several percentage values for the selected features (from 5% to 70%). Table 2 reports the number of features that were selected in each dataset using grid-search.

- Our method dubbed Linear SPARsification (LiSpar). We perform sparsification of L_2-SVM models according to Algorithm 1. In order to tune the hyper-parameters τ and ρ we relied on a simple cross-validation approach.

For each dataset and each algorithm we evaluated several values of the regularization parameter C ranging from 1 to 1000 . The performance of each approach is evaluated in terms of *accuracy* and *Macro F-Measure* (MaF).

Accuracy measures how often a classifier makes the correct prediction. To compute it, we used the following formula:

$$\text{Accuracy} = \frac{\text{\# of documents well classified}}{\text{\# predictions made}}$$

Considering a set of classes $\mathcal{Y} = \{1, \ldots, K\}$, the MaF is computed using the following formula:

$$MaF = \frac{2 * MaP * MaR}{MaP + MaR} \tag{2}$$

Where the macro precision (MaP) and the macro recall (MaR) are computed as:

$$MaP = \frac{\sum_{k=1}^{K} \frac{tp_k}{tp_k + fp_k}}{K} \tag{3}$$

$$MaR = \frac{\sum_{k=1}^{K} \frac{tp_k}{tp_k + fn_k}}{K} \tag{4}$$

Where tp_k, fp_k and fn_k are the true positives, false positives and false negatives respectively for class k.

As we consider imbalanced datasets, the MaF measure is interesting since it takes into account both precision and recall and gives the same importance to minority and majority classes.

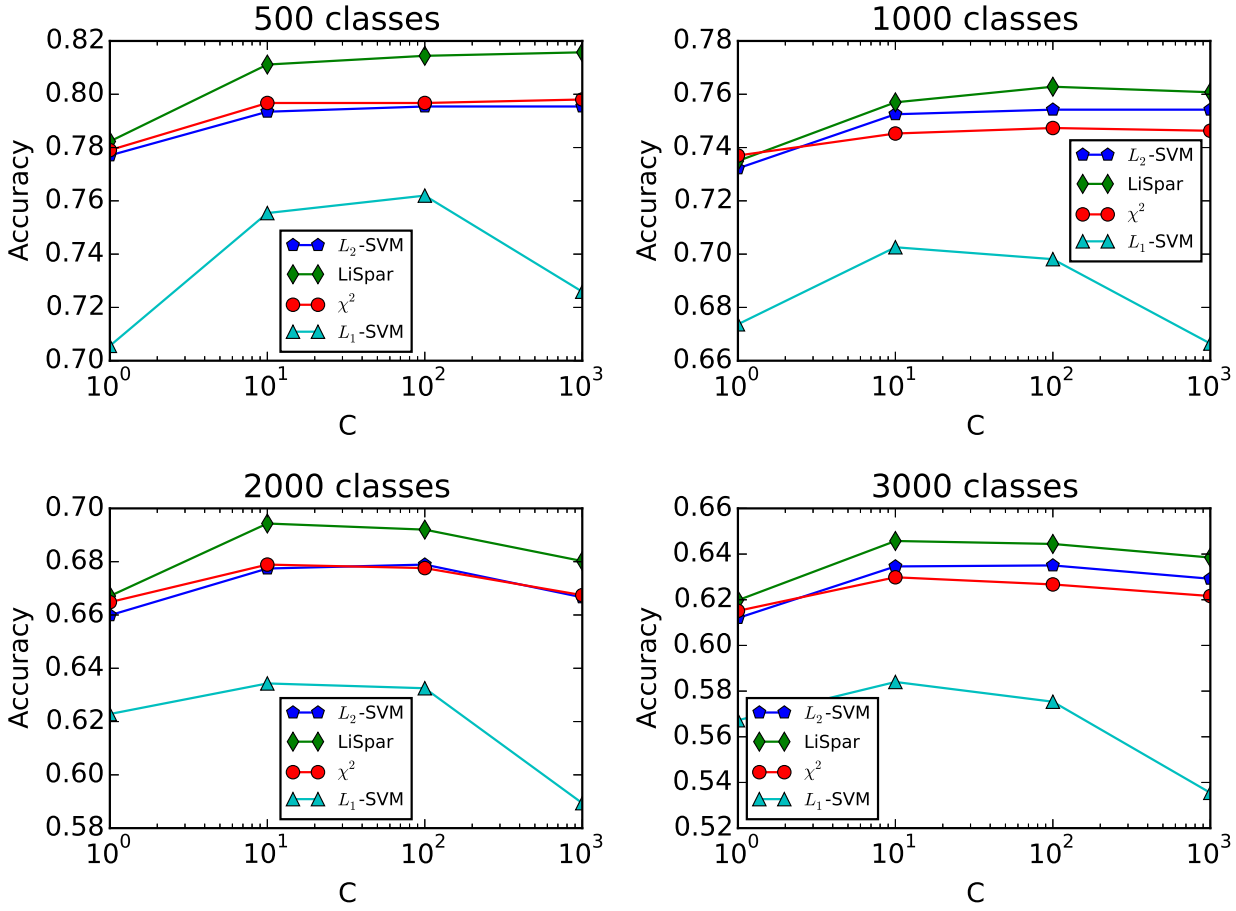
4.2 Results & Discussion

Classification Performance

Figure 2 presents the results in terms of accuracy for all competing algorithms across all datasets with respect to the penalty term C ranging from 1 to 1000 that leads to the best accuracy performance. We first observe that LiSpar outperforms all its rivals in all datasets. In most of the cases it has a stable behavior in the performance with respect to the values of the penalty term. We note that for low values of C the proposed approach will prune less values as the weight vector is bounded by the penalty term thus taking smaller values. On the other hand for larger values the weights become larger which leads to smaller percentages of pruning and in some cases to slighter improvement. LiSpar achieves its best results for parameter values C in the range of 10 to 100, where the model has not overfit the data and exhibits good performance before pruning.

L_1-SVM which produces very sparse models, hurts the performance in all cases. This is an expected behavior in large collections where the feature space exceeds the number of available examples and the features are

Figure 2: Accuracy for standard SVM (L_2-regularized, L_2-loss), LiSpar (our method, SVM and threshold), χ^2 feature selection and L_1-regularized L_2-loss SVM considering regularization $C \in \{0.1, 1, 10, 100, 1000\}$ and 500, 1000, 2000, and 3000 classes



correlated. On the other hand χ^2 maintains in most cases a similar or better performance than L_2-SVM, with the advantage of pruning a large number of uninformative features.

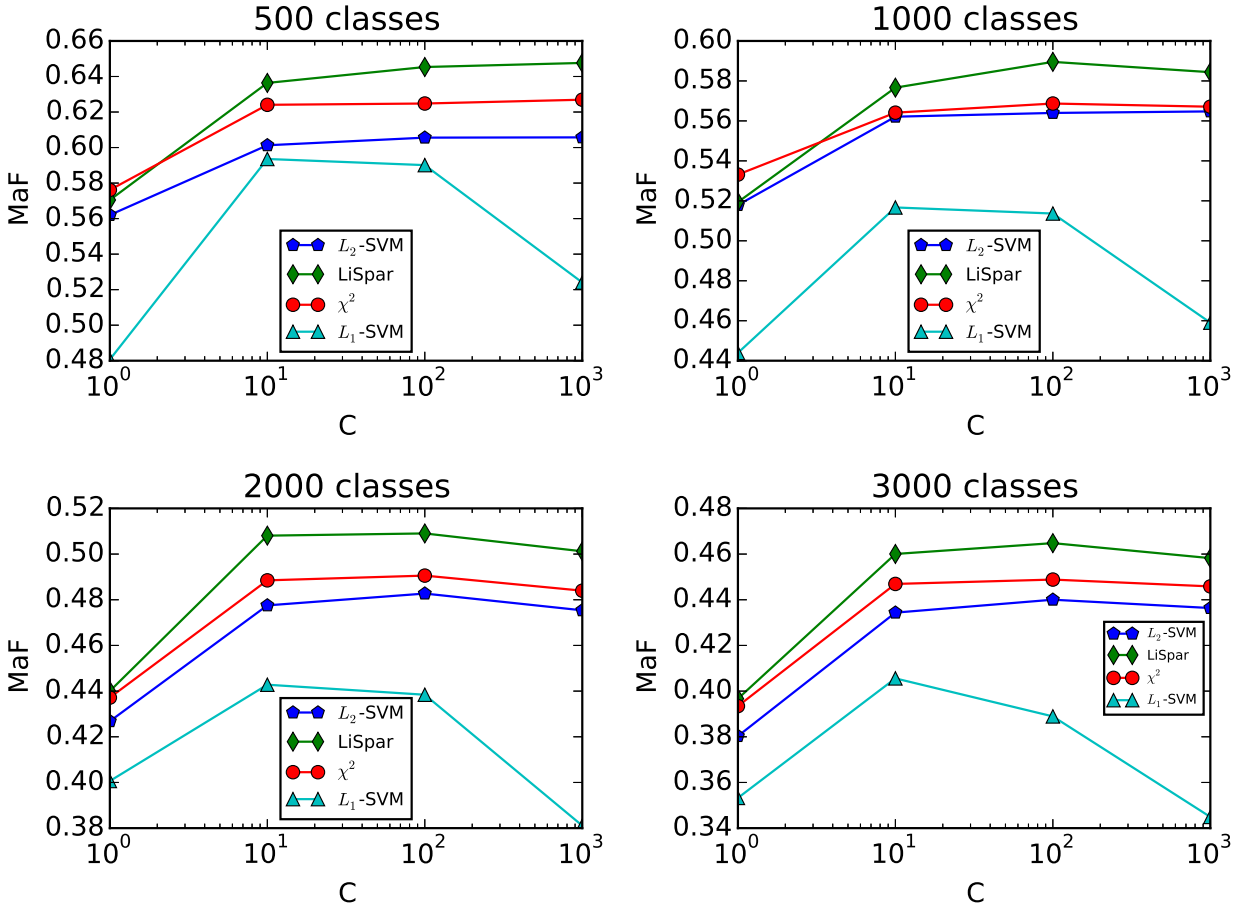
Figure 3 presents in the same fashion the comparison of the algorithms in terms of MaF. We can observe again a similar trend as LiSpar outperforms all its competing methods. The improvement of MaF confirms that the proposed pruning method improves particularly the classification of minority classes. Figure 4 presents the number of features that were totally pruned or not (which means that it remains at least one weight value for this feature) with respect to the term document frequency (DF) which is the number of documents a term appears in. Rare terms will have a small DF value while terms that are often used will have superior values.

Interestingly, the features that were completely pruned correspond to terms that either are rare or unimportant and thus have small values of DF. For rare terms this is a particularity for minority classes which have very few documents. On the other side for features that were not completely pruned the DF of the terms is higher and so the corresponding values in the models tend to be bigger.

Sparsity & Model size

In this section we present the sparsity ratio obtained by the proposed approach as well as the size of the model

Figure 3: Macro F-Measure (MaF) for standard SVM (L_2-regularized, L_2-loss), our method (SVM and threshold), χ^2 feature selection and L_1-regularized L_2-loss SVM considering regularization $C \in \{0.1, 1, 10, 100, 1000\}$ and 500, 1000, 2000, and 3000 classes



in the disk. We measure the sparsity of a model using Equation 1.

For measuring the size of the models we used a representation of similar to that of LIBLINEAR for storing a dataset by keeping only the non-zero values for each class vector. Specifically, for each class we represent the vector of weights as follows:

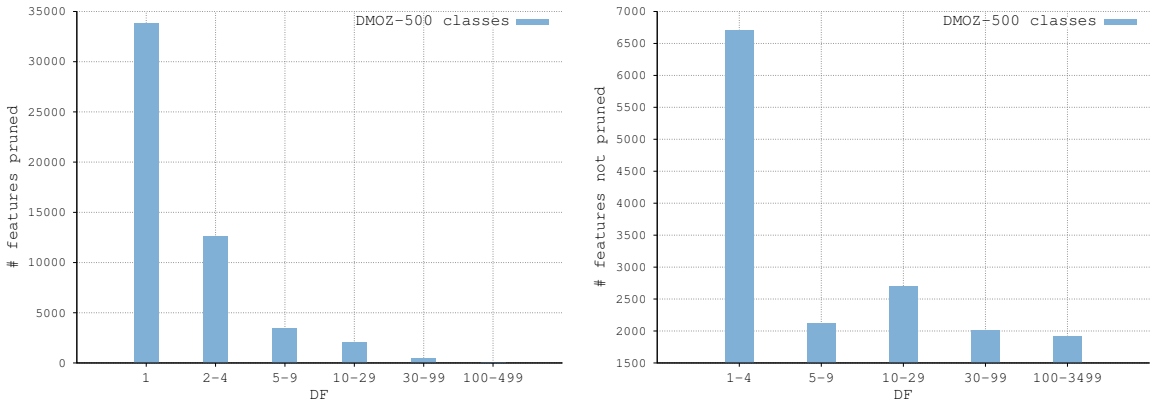
$$\begin{aligned} &\textit{class } 1 \; f_{a_1} : v_{a_1}, \ldots, f_{b_1} : v_{b_1} \\ &\ldots \\ &\textit{class } K \; f_{a_k} : v_{a_K}, \ldots, f_{b_K} : v_{b_k} \end{aligned}$$

where f_{a_j} and v_{a_j} are correspondingly the index and the value of feature a for the class j.

Table 3 presents the sparsity along with the model size in Megabytes for all methods across all datasets. We considered only the results for C= 100 as it usually gives the best results in accuracy. First we note that L_1-SVM achieves maximum sparsity in all cases leading to very small models. The proposed approach gives very sparse models (close to L_1-SVM) and the resulting models are at the worst case 10 times smaller than those produced by L_2-SVM. For larger number of classes, which means more features, LiSpar gives sparser models. For χ^2 as the algorithm selects informative features before the learning phase, it ends up with a lower sparsity than the other methods.

The behavior of LiSpar in terms of sparsity ratio has a two-fold implication. First, for huge datasets the models can be efficiently compressed and used in light embedded computing applications where size is

Figure 4: Number of features pruned with respect to their document frequency (number of documents they appear in).



(a) Fully pruned features.

(b) Not fully pruned features.

Table 3: Sparsity and models size for penalty parameter C= 100.

#Classes	L_2-SVM	LiSpar	L_1-SVM	χ^2
500	68.583% (260 Mb)	96.788% (27 Mb)	99.612% (3.1 Mb)	51.708% (72 Mb)
1000	72.479% (723 Mb)	97.062% (77 Mb)	99.701% (7.3 Mb)	57.004% (202 Mb)
2000	75.308% (2200 Mb)	98.228% (148 Mb)	99.769% (18 Mb)	66.072% (674 Mb)
3000	76.415% (4000 Mb)	99.194% (125 Mb)	99.786% (33 Mb)	66.524% (1100 Mb)

critical. Second, and more importantly, it allows to reduce significantly the prediction time as one can rely on a sparse scalar product for classifying a new example.

Discussion
In short, these experiments have demonstrated that it is possible to use an a posteriori method to obtain a very sparse model while improving the predictive performance. Intuitively, by removing small values from the model, we remove noise and thus small perturbations in predictions. The proposed method provides better and faster classifiers by removing the impact of irrelevant features on the decision.

5 Conclusion and Future Work

In this paper we proposed a simple approach to sparsify linear models. Whereas most approaches work a priori, this method works a posteriori and gives very sparse models while it achieves to improve the performance for text classification tasks. Compared to other sparse approaches such as LASSO, this method gives better results in terms of accuracy and MaF while the level of sparsity remain close.

For future work several directions will be investigated:

- Find a way to formally obtain the best values of the threshold's parameters investigating the loss occurred during prediction.
- Use LiSpar as a feature selection model, for instance by removing all the features for which all the weight values equal zero.
- Test the same approach on different kind of datasets (e.g. images, biological datasets and so on).

References

[AAG+11] Sujeevan Aseervatham, Anestis Antoniadis, Éric Gaussier, Michel Burlet, and Yves Denneulin. A sparse version of the ridge logistic regression for large-scale text categorization. *Pattern Recognition Letters*, 32(2):101–106, 2011.

[Bac08] Francis R. Bach. Bolasso: Model consistent lasso estimation through the bootstrap. In

Proceedings of the 25th International Conference on Machine Learning, pages 33–40, 2008.

[BBE+03] Jinbo Bi, Kristin Bennett, Mark Embrechts, Curt Breneman, and Minghu Song. Dimensionality reduction via sparse support vector machines. *Journal Machine Learning Research*, 3:1229–1243, 2003.

[BMP+14] Rohit Babbar, Cornelia Metzig, Ioannis Partalas, Eric Gaussier, and Massih-Reza Amini. On power law distributions in large-scale taxonomies. *SIGKDD Explor. Newsl.*, 16(1):47–56, September 2014.

[CS02] Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, 2002.

[FCH+08] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[GE03] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3:1157–1182, 2003.

[GSMY13] Daniel Golovin, D. Sculley, H. Brendan McMahan, and Michael Young. Large-scale learning with less ram via randomization. In *International Conference on Machine Learning*, volume 28 of *JMLR Proceedings*, pages 325–333. JMLR.org, 2013.

[KA03] Yoshiaki Koshiba and Shigeo Abe. Comparison of l1 and l2 support vector machines. In *Neural Networks, 2003. Proceedings of the International Joint Conference on*, volume 3, pages 2054–2059. IEEE, 2003.

[LLZ09] John Langford, Lihong Li, and Tong Zhang. Sparse online learning via truncated gradient. *Journal of Machine Learning Research*, 10:777–801, 2009.

[MBP14] Julien Mairal, Francis Bach, and Jean Ponce. *Sparse Modeling for Image and Vision Processing*, volume 8 of *Foundations and Trends in Computer Graphics and Vision*. now publishers, 2014.

[PKB+15] Ioannis Partalas, Aris Kosmopoulos, Nicolas Baskiotis, Thierry Artieres, George Paliouras, Eric Gaussier, Ion Androutsopoulos, Massih-Reza Amini, and Patrick Galinari. Lshtc: A benchmark for large-scale text classification. *CoRR*, abs/1503.08581, march 2015.

[SWY75] G. Salton, A. Wong, and C.S. Yang. A vector space model for automatic indexing. *ACM Communications*, 18:613–620, 1975.

[TG14] Shingo Takamatsu and Carlos Guestrin. Reducing data loading bottleneck with coarse feature vectors for large scale learning. In *Proceedings of the 3rd International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications, BigMine 2014, New York City, USA, August 24, 2014*, pages 46–60, 2014.

[Tib94] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288, 1994.

[TWT10] Mingkui Tan, Li Wang, and Ivor W. Tsang. Learning sparse SVM for feature selection on very high dimensional datasets. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, pages 1047–1054, 2010.

[WDL+09] Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. Feature hashing for large scale multitask learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1113–1120, 2009.

[YP97] Yiming Yang and Jan O. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, pages 412–420, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[ZH05] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.