TerseSVM : A Scalable Approach for Learning Compact Models in Large-scale Classification

Rohit Babbar, Krikamol Maundet, Bernhard Schölkopf
Max-Planck Institute for Intelligent Systems, Tübingen

Abstract

For large-scale multi-class classification problems, consisting of tens of thousand target categories, recent works have emphasized the need to store billions of parameters. For instance, the classical l_2-norm regularization employed by a state-of-the-art method results in the model size of 17GB for a training set whose size is only 129MB. To the contrary, by using a mixed-norm regularization approach, we show that around 99.5% of the stored parameters is dispensable noise. Using this strategy, we can extract the information relevant for classification, which is constituted in remaining 0.5% of the parameters, and hence demonstrate drastic reduction in model sizes. Furthermore, the proposed method leads to improvement in generalization performance compared to state-of-the-art methods, especially for under-represented categories. Lastly, our method enjoys easy parallelization, and scales well to tens of thousand target categories.

1 Introduction

With the emergence of big data, large-scale classification problems have gained considerable momentum in recent years. In this respect, classification with large number of target categories has attracted attention of machine learning researchers and practitioners. Datasets consisting of tens of thousand target categories are common in various domains such as product categorization for e-commerce [6, 22, 1, 19], large-scale classification of images [16] and text [13, 7]. Various open challenges such as Large Scale Hierarchical Text Classification (LSHTC, `http://lshtc.iit.demokritos.gr/`) and ImageNet Large Scale Visual Recognition Challenge (ILSVRC, `http://image-net.org/challenges/LSVRC/2015/`) have been organized to provide a benchmark for state-of-the-art methods and push the frontiers of contemporary research in this direction. Large-scale text datasets released as part of LSHTC typically consist of hundreds of thousand training instances which are distributed among tens of thousand target categories. The training instances live in a high dimensional space spanning upto millions of dimensions. However, since each instance only consists of few hundred words, its vector-space representation is highly sparse.

Training classifiers for such large-scale scenarios, leads to computational and statistical challenges. From the *computational aspect*, for the learning process to finish in reasonable amount of time, it is desirable that the learning machine scales well with number of target categories. Though One-vs-Rest [23] is a natural way to parallelize the training of categories, but it is often compared with Crammer-Singer Multi-class SVM (`CS-MSVM`) proposed in [10] which also enjoys strong theoretical guarantees. However, `CS-SVM` can not be parallelized across categories, and the optimization process cannot exploit the large-number of computing units even if these were available. Furthermore, it has to simultaneously store the weight vectors of all the categories in the main memory, thereby acting as a bottleneck when dealing with tens of thousand categories.

From the *statistical view-point*, since most of categories have very few training instances which belong to them, it becomes harder to detect them in test set. For instance, 76% of the categories in the Yahoo! Directory have less than 5 instances that belong to them and 72% of the Open Directory project have less than 4 instances [13]. As a result, most state-of-the-art methods exhibit low values of Macro-F1 measure, a metric which captures the method's performance on rare categories.

1.1 State-of-the-art Methods

To address the challenges arising in the above-mentioned scenarios, various approaches have been proposed. We focus on two recent works which represent the state-of-the-art on LSHTC datasets.

To address the **computational challenge**, the work in [12] focusses on parallelizing training of large-scale Regularized Multinomial Logistic Regression (`RMLR`). The cross-entropy based loss function was relaxed such that the weight vectors for all the target categories can be trained concurrently. For a training set $\{\mathbf{x}_i, c_i\}_{i=1}^m$ such that $\mathbf{x}_i \in \mathbb{R}^D$ and $c_i \in 1 \dots K$ where K denotes the number of target categories, let b_i^k denote the indicator variable that instance $\mathbf{x}_i$ belong to category k, i.e., $b_i^k = I\{c_i = k\}$. Let $\mathbf{W} = [\mathbf{w}_1 \dots \mathbf{w}_k \dots \mathbf{w}_K]$ denote the matrix constituted by the weight vectors for each category, then the l_2 regularized version of the optimization problem for minimizing the neg-

Dataset	Training instances	Categories	Features	Parameters	Train Size	Model Size	$\frac{\text{Model Size}}{\text{Train Size}}$
CLEF	10,000	63	80	5,040	3MB	40KB	0.013
NEWS20	11,260	20	53,975	1,079,500	11MB	4MB	0.363
LSHTC-small	4,463	1,139	51,033	58,126,587	**5MB**	**911MB**	**182.2**
LSHTC-large	93,805	12,294	347,256	4,269,165,264	**129MB**	**17GB**	**134.9**
Imagenet-2012	1.3M	1,000	150,528	144M	**138GB**	**245MB**	0.0001

Table 1: Reproduced from Table 1 of [12] except the last row(showing statistics for Imagenet data) and the last column(ratio of training file size to model file size). Extra row and column are added to for a relative comparison. M stands for million.

ative log-likelihood is given by

$$OPT1 : \min_{\mathbf{w}_1 \ldots \mathbf{w}_K} \frac{\lambda}{2} \sum_{k=1}^{K} ||\mathbf{w}_k||_2^2 - \sum_{k=1}^{K} \sum_{i=1}^{m} b_i^k \mathbf{w}_k^T \mathbf{x}_i + \sum_{i=1}^{m} \log \left(\sum_{k'=1}^{K} \exp(\mathbf{w}_{k'}^T \mathbf{x}_i) \right)$$

where $||\mathbf{w}_k||_2 = \left(\sum_{d=1}^{D} \mathbf{W}_{k,d}^2 \right)^{1/2}$. The last term in $OPT1$ consisting of log-sum-exp couples the parameters $\mathbf{w}_k$ for each category, and hence is not amenable to concurrent training. The work in [12] exploits the first-order concavity of log-function by replacing it with an upper bound given by

$$log(\gamma) \leq a\gamma - \log(a) - 1 \quad \forall \gamma, a > 0$$

As a result, the log-sum-exp in OPT1 is upper bounded as follows

$$\log \left(\sum_{k'=1}^{K} \exp(\mathbf{w}_{k'}^T \mathbf{x}_i) \right) \leq a_i \sum_{k'=1}^{K} \exp(\mathbf{w}_{k'}^T \mathbf{x}_i) - \log(a_i) - 1$$

At the cost of additional parameters a_i, by replacing the last term in $OPT1$ by the above bound, the RMLR optimization objective given in $OPT1$ is amenable to category level concurrent training. In principle, the method can exploit as many computing unit as the number of target categories.

To address the **statistical challenge** arising from large fraction of rare categories in large-scale datasets, another approach, Hierarchically Regularized SVM (HR-SVM), was proposed by the same authors in [13]. In this technique, they exploit the taxonomy information and the regularizer is designed such that the weight vector of an internal node is encouraged to be closer (in terms of l_2-norm) to its parent and children nodes in the tree taxonomy. HR-SVM is a state-of-the-art method in terms of Micro-F1 metric and Macro-F1 measures on LSHTC datasets.

1.2 Model Size and more Although text classification has been studied extensively for decades, a straightforward extension to large-scale scenarios, such as those consisting of large number of target categories, can lead to unexpected outcomes. In the context of state-of-the-art methods, we illustrate this using Table 1 which has been reproduced from [12], (except for the last row and the last column). We would focus on addressing the following three concerns:

1. **Model size**: As shown in Table 1, the work in [12] emphasizes the need to store billions of parameters, 4 Billion for **LSHTC-large**. The issues we would like to point out are : For **LSHTC** datasets, is it reasonable to have 911MB model for a 5MB training file, and 17GB model for a 129MB training file? Are these trained models storing a large-amount of undesirable and dispensable information? To the contrary, how does it compare to ImageNet dataset, where 138GB of training data can be compressed into a 245MB model by using 16 convolutional layers and 3 fully connected layers [24]?

2. **Classification accuracy** : Is it possible to achieve better generalization accuracy than the state-of-the-art HR-SVM method [13]. Furthermore, can better accuracy be achieved simultaneously with less number of parameters and hence have trained model sizes of the same order as the training data size? How should one be able to find those parameters which are dispensable?

3. **Scalability** : To ensure scalability to large-scale scenarios, can we also achieve concurrent training in addition to the above two goals of small model sizes and better generalization performance? More importantly, is it possible to achieve parallel training without making upper bound approximations such as that made by using log concavity upper bound?

In this work, we show that all the three concerns can be effectively addressed under a single framework which goes beyond the commonly employed l_2-norm based regularization strategy typically employed in most state-of-the-art methods and off-the-shelf solvers such as Liblinear. We demonstrate that l_2-norm may not be a suitable regularizer for large-scale classification involving thousands of target categories wherein the distribution of training instances

among target categories exhibits fit to power-law distribution. This being the cause, large model sizes generated by state-of-the-art methods is a symptom and indication towards the unsuitability of regularization scheme to capture the data distribution.

In these scenarios, we apply mixed-norm $l_{1,2}$ regularization strategy which models the distribution of large-scale datasets such as **LSHTC** more appropriately as compared to most regularization schemes typically employed. The mixed-norm approach provides a fine balance between the two extremes of l_2-norm and l_1-norm as two most applied regularization methods. Firstly, we show that only a very small fraction of the parameters that are learnt by state-of-the-art methods such as [12], 0.5% for **LSHTC-large**, are retained in the final model. Secondly, we show that our method yield better generalization performance compared to state-of-the-art methods especially for rare categories which are under-represented in the training set. Finally, from the implementation point of view, our method enjoys easy parallelization without having to make upper bound approximation on the loss function and easily scales to approximately 30,000 target categories.

Matrix Norms Before proceeding, we recall some of the useful matrix norms. For the parameter matrix $\mathbf{W}$, the $l_{p,q}$ norm is defined as

$$||\mathbf{W}||_{p,q} = \left[\sum_{k=1}^{K}\left[\sum_{d=1}^{D}|\mathbf{W}_{d,k}|^p\right]^{q/p}\right]^{1/q} \tag{1.1}$$

Typically employed convex regularizers in various machine learning algorithms are special cases of the form $||\mathbf{W}||_{pq}^q$. The regularization $\sum_{k=1}^{K}||\mathbf{w}_k||_2^2$ used in `RMLR` in $OPT1$ can be obtained by setting $p = q = 2$. Similarly, the commonly employed sparsity inducing regularizer, Lasso, can be obtained by setting $p = q = 1$. Group Lasso as a regularization mechanism, which induces sparsity for an entire group of variables is obtained by setting $p = 2, q = 1$. However, in this work we show that the setting $p = 1, q = 2$, i.e. using $||\mathbf{W}||_{1,2}^2 = \sum_{k=1}^{K}||\mathbf{w}_k||_1^2$ as the regularization mechanism simultaneously addresses the three challenges raised above. In the next section, we also argue the suitability of this regularizer against the other choices for solving multi-class classification problems consisting of tens of thousand target categories. Sparsity inducing norms and applications of mixed-norm regularization to control the structure of the induced sparsity has been studied in works such as multiple kernel learning, signal processing, multi-class and multi-task learning [5, 15, 25, 14]. However, to the best of our knowledge, this is the first time it is applied for large-scale multi-class problems to yield state-of-the-art results by an efficient implementation which is scalable to tens of thousand target categories. More importantly, the contribution of our work is also to dispel the notion of having to store billions of parameters in large model files under the guise of knowledge gained by spending computing power through the learning algorithm when the training data is orders of magnitude smaller.

2 Problem setup

Let $\mathcal{X} \subseteq \mathbb{R}^D$ be the input space and let $\mathcal{C} = \{1 \dots K\}$ be a finite set of target categories. We assume that training examples are m i.i.d pairs $(\mathbf{x}_i, c_i)$ drawn according to a fixed but unknown distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{C}$. The goal is to choose an appropriate function class $\mathcal{F}$, control its capacity by a regularization mechanism which is appropriate for the problem at hand and learn a function $f \in \mathcal{F}$ such that test error, defined as $\mathbb{E}_{(\mathbf{x},c)\sim\mathcal{D}}[f(\mathbf{x}) \neq c]$, is minimal. In practice, this is typically achieved by minimizing a conic combination of a convex loss function and a convex regularizer which controls the complexity of $\mathcal{F}$. For the settings considered in this paper, m is of the order of 10^4 to 10^5, K is of the order of 10^3 to 10^4, and D is in the range 10^5 to 10^6. For high dimensional input spaces, $\mathcal{F}$ is typically chosen to be the class of linear functions, and for multi-class classification $f(\mathbf{x})$ is of the form $f(\mathbf{x}) = \arg\max_{k\in\{1\dots K\}} f_k(\mathbf{x}) = \arg\max_{k\in\{1\dots K\}} \mathbf{w}_k^T\mathbf{x}$. The goal is therefore to efficiently learn the weight vectors $\mathbf{w}_k, \forall k \in \{1 \dots K\}$ such that the generalization error is minimal. It should be also be noted that in most naturally occurring large-scale category systems, such as those manifested by **LSHTC** datasets in Table 1, the distribution of training instances among target categories exhibit fit to power-law distribution.

The above problem, simple to state, bears semblance to one of the first machine learning problems encountered by a practitioner. However, applying conventional ideas such as using l_2-norm regularization to large-scale problems involving very high-dimensional data, can lead to unexpected observations, for instance, unreasonably large model sizes in Table 1 reproduced from [12]. We show that in such situations, our proposed mixed-norm regularization framework is more adaptive in discovering the underlying structure.

3 TerseSVM

We now formulate the optimization problem for our proposed method which simultaneously handles the three drawbacks of state-of-the-art methods i.e., extremely large models size, low generalization performance and distributed training. We call our method `TerseSVM`, as this can learn models which are concise in size and more effective in classification and faster in training.

Using the mixed-norm regularization by setting $p = 1, q = 2$ in equation 1.1, and setting the loss function to squared hinge loss, we get the proposed formulation for learning the weight vectors $\mathbf{W} = [\mathbf{w}_1 \dots \mathbf{w}_k \dots \mathbf{w}_K]$. The

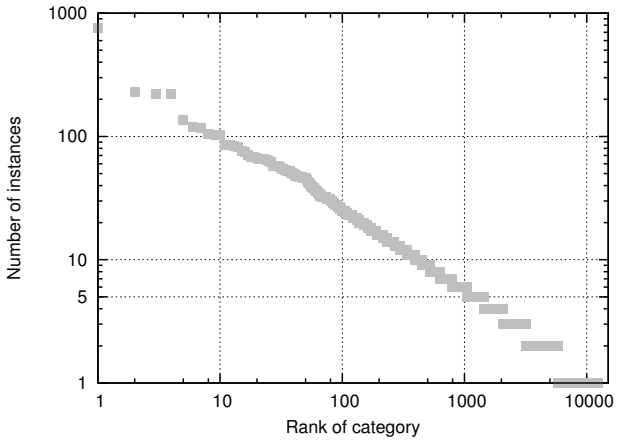

Figure 1: Distribution of training instances among categories in LSHTC-large dataset. Only 1,000 of the 12,294 categories have more than 5 training instances in them.

objective function is given by

(3.2)
$$\min_{\mathbf{w}_1 \ldots \mathbf{w}_K} \left[\frac{\lambda}{2} \sum_{k=1}^{K} ||\mathbf{w}_k||_1^2 + \sum_{k=1}^{K} \sum_{i=1}^{m} (\max(0, 1 - y_i^k \mathbf{w}_k^T \mathbf{x}_i))^2 \right]$$

where λ is the regularization parameter, and $y_i^k = +1$ if $c_i = k$, and -1 otherwise and $||\mathbf{w}_k||_1^2 = \left(\sum_{d=1}^{D} |\mathbf{W}_{k,d}|\right)^2$. The $l_{1,2}$-norm regularization is the square of the l_2-norm of the l_1-norm of individual weight vectors $\mathbf{w}_k$. In other words, it computed by first individually taking the l_1-norm of K weight vectors, each of which is D-dimensional. Then l_2-norm of the resulting K-dimensional vector is squared. Note that the objective function is a sum of smooth loss function and a non-differentiable regularization term, a property which will be used later in using the proximal gradient method for solving the optimization problem.

3.1 Why make this non-obvious choice? We now argue that the proposed mixed-norm regularization is the most suitable capacity control mechanism as compared to the other options : l_2-norm or the Frobenius norm, l_1-norm, also referred to as Lasso and finally Group Lasso.

- **l_2-norm, (Frobenius norm)**: The standard l_2-norm based regularization given by $||\mathbf{W}||_{2,2}^2 = \sum_{k=1}^{K} ||\mathbf{w}_k||_2^2$ has been applied in most machine learning algorithms such as Ridge regression, SVM, `CS-SVM` and also the most modern variants such as parallel-version of `RMLR` as discussed in optimization problem $OPT1$ in Section 1. We would like to point out that the extremely large models sizes are the result of using l_2-norm as a regularization. Intuitively, this can be understood as follows: Firstly, the distribution of training instances among target categories in large-scale datasets such as **LSHTC** in Table 1 follows a fat-tailed power-law distribution [2]. This is shown in Figure 1 for the LSHTC-large dataset in Table 1. Formally, this can be described by the following:

$$m_r = m_1 r^{-\beta}$$

where m_r represents the number of training instances in the r-th ranked category, when ranked in descending order according to the number of instances, and $\beta > 0$ is the exponent of the power-law distribution. As a result of this phenomenon, coupled with the fact that each training instance is sparsely represented by only few tens of words/features, most of the categories are represented by only a few hundreds features. The remaining features, of the order of hundreds of thousands, which do not occur in the few training instances representing that category, are taken to be negative features. The classifier learns to assign small negative weights to these negative features. Secondly, l_2-norm regularization does not lead to a heavy penalty by squaring a small negative weight, which becomes even smaller. Therefore, the model file ends up storing large number of small negative weights, and hence leading to model file sizes of 17GB for a 129MB training file. The situation for rare categories is worsened by the fact that lots of small negative weights cumulate together and do not allow the magnitude of weights corresponding to the present hundred words to go higher. As a result, the trained model though having size 100 times more than the training file, fails to detect them in the test set.

- **l_1-norm, (Lasso)**: On the other extreme is the sparsity inducing l_1-norm regularization given by $||\mathbf{W}||_{1,1}^1 = \sum_{k=1}^{K} ||\mathbf{w}_k||_1$ which promotes sparsity both at a per category level and also at the global level. As l_1-norm regularization is known to promote excessive sparsity, this method leads to extremely compact models. As we will show in our experiments, its major drawback is that it leads to the worst test set accuracy and detection of rare categories for all datasets.

- **$l_{2,1}$-norm, (Group-Lasso)**: This is also a sparsity inducing (mixed) regularization scheme, and is given by $||\mathbf{W}||_{2,1}^1 = \sum_{k=1}^{K} ||\mathbf{w}_k||_2$. Given the grouping of parameters, it first takes the l_2-norm of each group, and yields a scalar per group and then takes an l_1-norm of the resulting vector. As a result, it either keeps an entire group of variables or suppresses it completely. In our scenario, the natural grouping of parameters is by individual categories, i.e., all the entries in the D-dimensional vector $\mathbf{w}_k$ are grouped together. Since l_2-norm is based on squaring the individual entries, it al-

lows the entries to take small values in the neighborhood of zero. As D is quite large, of the order hundreds of thousand, the l_2-norm of each $\mathbf{w}_k >> 0 \forall k \in \{1 \dots K\}$, and hence, none of the group is suppressed. Therefore, it leads to negligible sparsity, and the same was observed in our experiments as it results in unreasonably large models consisting of large degree of noisy values.

- $l_{1,2}$**-norm**: The proposed $l_{1,2}$-norm regularization is also a sparsity inducing (mixed) norm, and is given by $||\mathbf{W}||_{1,2}^2 = \sum_{k=1}^{K} ||\mathbf{w}_k||_1^2$. In this strategy, firstly $||\mathbf{w}_k||_1 \forall k$ is computed, since l_1-norm penalizes the quantities which are near zero as much as they are far away from zero, those near zero are forced to become zero. Therefore, it tends to keep only useful entries per $\mathbf{w}_k$, i.e., those which are far away from zero, depicting high confidence, are kept. On the other hand, the ones close to zero, depicting lesser confidence, are forced to become zero, hence leading to more compact models. From the first step, a K-dimensional vector is created which has smaller entries corresponding to rare categories having few training instances in the training set. As a second step, l_2-norm of the resulting K-dimensional vector is taken. Since, unlike l_1-norm, l_2-norm is lenient towards small entries, the weight vectors for rare categories are preserved. As a result, this choice of regularization is more suitable for the large-scale classification consisting of hundreds of thousand training instances whose distribution among tens of thousand target categories exhibits fit to power-law distribution.

3.2 Parallelizing the Optimization To ensure scalability to tens of thousand target categories and perform concurrent training to learn $\mathbf{w}_k$'s simultaneously, we observe that the mixed-norm optimization problem in equation (3) can be easily split as follows:

$$\begin{aligned}
&\min_{\mathbf{w}_1 \dots \mathbf{w}_K} \left[\frac{\lambda}{2} \sum_{k=1}^{K} ||\mathbf{w}_k||_1^2 + \sum_{k=1}^{K} \sum_{i=1}^{m} (\max(0, 1 - y_i^k \mathbf{w}_k^T \mathbf{x}_i))^2 \right] \\
&= \min_{\mathbf{w}_1} \left[\frac{\lambda}{2} ||\mathbf{w}_1||_1^2 + \sum_{i=1}^{m} (\max(0, 1 - y_i^1 \mathbf{w}_1^T \mathbf{x}_i))^2 \right] \\
&+ \min_{\mathbf{w}_2} \left[\frac{\lambda}{2} ||\mathbf{w}_2||_1^2 + \sum_{i=1}^{m} (\max(0, 1 - y_i^2 \mathbf{w}_2^T \mathbf{x}_i))^2 \right] \\
&+ \dots \\
&+ \min_{\mathbf{w}_K} \left[\frac{\lambda}{2} ||\mathbf{w}_K||_1^2 + \sum_{i=1}^{m} (\max(0, 1 - y_i^K \mathbf{w}_K^T \mathbf{x}_i))^2 \right]
\end{aligned}$$

The above expansion shows that the joint objective as a function of the weight matrix $\mathbf{W}$, can be individually optimized over the weight vectors $\mathbf{w}_k$'s, hence enable concurrent training for all categories. In addition, the above expansion enjoys the following advantages :

- Unlike in [12], it does not make any approximation, and hence minimizes the true objective, and
- From an implementation view-point, there are no constraints for storing all the $\mathbf{w}_k$'s in the main memory, which is the case of Crammer-Singer Multi-class SVM, `CS-SVM`.

Each individual optimization problem above is convex but non-smooth due to the presence of the mixed-norm regularization. Such problems can be solved by proximal algorithms which are a general class of optimization methods and sit at a higher layer of abstraction than classical algorithms such as Newton's method [20, 21]. For our setup, we use the following forward-backward proximal gradient method [9]:

Algorithm 1 Procedure for minimizing over individual weight vector $\mathbf{w}_k$

1: Initialize $\mathbf{w}_k^{(0)}$=**0**, t=0
2: **while** not converged **do**
3: $\mathbf{w}_k'^{(t)} = \mathbf{w}_k^{(t)} - \gamma_k^{(t)} \nabla_{\mathbf{w}_k^{(t)}} \left[\sum_{i=1}^{m} (\max(0, 1 - y_i^k {\mathbf{w}_k^{(t)}}^T \mathbf{x}_i))^2 \right]$
4: $\mathbf{w}_k^{(t+1)} = \arg\min_{\mathbf{w}} \left[\frac{\lambda \gamma_k^{(t)}}{2} ||\mathbf{w}||_1^2 + \frac{1}{2} ||\mathbf{w} - \mathbf{w}_k'^{(t)}||_2^2 \right]$
5: $t = t + 1$
6: **end while**

The algorithmic procedure consists of two main steps:

- **Forward Step** (Line 3) : In this step, gradient of the differentiable part at the current iterate is computed, and intermediate point (referred to as $\mathbf{w}_k'^{(t)}$) before the next iterate is reached. Here $\gamma_k^{(t)}$ represents the step-length at the t-th iteration.
- **Backward/Proximal Step** (Line 4) : In this step, a point in the *proximity* of the intermediate point is computed by solving another optimization sub-problem, and it serves as the next iterate $\mathbf{w}_k^{(t+1)}$. This is also known as the proximal operator, and for the function $||\mathbf{w}||_1^2$, the computation of the proximal operator admits a closed-form solution.

The algorithm was implemented in C++ and OpenMP was used to enable concurrent training.

4 Experimental evaluation

4.1 Dataset description We present results on six publicly available datasets where number of target categories range from 20 to 27,875, and the dimensionality ranges

Dataset	# Training instances	# Test instances	# Categories	# Features
NEWS20	15,935	3,993	20	62,061
IPC	46,324	28,926	451	1,123,497
LSHTC-small	4,463	1,858	1,139	51,033
LSHTC-large	128,710	34,880	12,294	381,581
LSHTC-2012	383,408	103,435	11,947	575,555
LSHTC-2011	394,756	104,263	27,875	594,158

Table 2: Datasets and their properties

from 51,033 to 1,123,497. The other relevant statistics of the datasets are shown in Table 2. **NEWS20** dataset is taken from `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass.html`, **IPC** dataset is taken from `http://gcdart.blogspot.de/2012/08/datasets_929.html`, and **LSHTC-*** datasets are taken from `http://lshtc.iit.demokritos.gr/`. For comparison of performance measures and model sizes, these datasets are exactly the same (except **NEWS20**) as used in state-of-the-art methods [13, 12].

These datasets have different properties, (c.f. last two columns of Table 2): (i) **NEWS20** is high-dimensional text dataset, but is extremely well balanced among the 20 target categories, (ii) **IPC** is high-dimensional dataset but unlike **NEWS20** is imbalanced among categories. However, it does not exhibit fit to power-law distribution.(iii) **LSHTC-*** datasets are also high-dimensional text datasets and exhibit fit to power-law distribution (see Figure 1).

4.2 Evaluation Metrics The metrics used for comparison are Micro-F1 measure and Macro-F1 measure, which are computed as follows:

- **Micro-F1** : It is an instance based evaluation measure and therefore weighs higher those categories which have higher fraction in the test set. Let TP_c, FP_c and FN_c denote respectively the true-positives, false-positives, and false-negatives for the class label $c \in \mathcal{C}$. Then Micro-F1 measure is given by

$$P = \frac{\sum_{c\in\mathcal{C}} TP_c}{\sum_{c\in\mathcal{C}} TP_c + FP_c}; R = \frac{\sum_{c\in\mathcal{C}} TP_c}{\sum_{c\in\mathcal{C}} TP_c + FN_c}$$
$$Micro - F1 = \frac{2PR}{P+R}$$

- **Macro-F1** : It is a category-based evaluation measure and weighs all categories equally and hence is more sensitive to the ability of the classifier to detect rare-categories. It is given by

$$P_c = \frac{TP_c}{TP_c + FP_c}; R_c = \frac{TP_c}{TP_c + FN_c}$$
$$Macro - F1 = \frac{1}{|\mathcal{C}|}\sum_{c\in\mathcal{C}} \frac{2P_cR_c}{P_c + R_c}$$

4.3 Methods for comparison We compare 6 different methods including state-of-the-art methods on the **LSHTC-*** and **IPC** datasets. These are:

- `TerseSVM` - This is our method proposed in Algorithm 1, which uses the $l_{1,2}$ regularization.
- `HR-SVM` - This method also uses the underlying semantic relationship among target categories and is state-of-the-art for large-scale classification on most large-scale **LSHTC-*** datasets in Table 2.
- `CS-MSVM` - Crammer-Singer Multiclass SVM from a very strong base-line algorithm and has been implemented in the Liblinear package [11]. This learning algorithm enjoys strong theoretical guarantees such as *universal consistency* property.

$$\min_{\mathbf{w}_1\ldots\mathbf{w}_K,\xi_i} \frac{1}{2}\sum_{k=1}^{K} ||\mathbf{w}_k||_2^2 + C\sum_{i=1}^{m}\xi_i \text{ subject to}$$
$$\mathbf{w}_{c_i}^T\mathbf{x}_i - \mathbf{w}_k^T\mathbf{x}_i \geq 1 - e_i^k - \xi_i, \xi_i \geq 0,\ \forall i = 1\ldots m,$$

  where $C > 0$ indicates the penalty for misclassification, and $e_i^k = 0$ if $c_i = k$, and $e_i^k = 1$ otherwise.

- `l2-SVM` - We used the squared hinge loss as the loss function and square of the Frobenius norm of the weight matrix as the regularizer which is given by $\sum_{k=1}^{K} ||\mathbf{w}_k||_2^2$.
- `l1-SVM` - The loss function is kept the same, and the regularizer is the l_1-norm of the l_1-norm of the weight vectors given by $\sum_{k=1}^{K} ||\mathbf{w}_k||_1$.
- `Group-Lasso` - The loss function is the same as above and regularizer used in this case is $\sum_{k=1}^{K} ||\mathbf{w}_k||_2$. The algorithm to implement Group-Lasso is same as that proposed in Algorithm 1, except the third step for computing the proximal operator, which can also be computed in closed-form and is given by

$$\mathbf{w}_k^{(t+1)} = \mathbf{w}_k'^{(t)} \times \left[\max\left(0, 1 - \frac{\lambda}{||\mathbf{w}_k'^{(t)}||_2}\right)\right]$$

Dataset	l_2−SVM	l_1−SVM	CS−MSVM	Group−Lasso	HR−SVM	TerseSVM
NEWS20						
Macro-F1	85.27	82.09	85.27	**85.39**	85.27	85.15
Micro-F1	85.37	82.04	85.37	**85.41**	85.37	85.11
IPC						
Macro-F1	51.23	47.46	51.74	51.55	47.89	**52.35**†
Micro-F1	53.12	50.73	53.82	53.90	54.26	**55.07**†
LSHTC-small						
Macro-F1	26.89	16.03	28.62	29.01	28.94	**30.60**†
Micro-F1	43.34	33.10	45.21	45.53	45.31	**46.12**†
LSHTC-large						
Macro-F1	31.27	21.54	32.64	32.60	33.12	**34.76**†
Micro-F1	45.12	39.51	45.36	45.83	46.02	**47.12**†
LSHTC-2012						
Macro-F1	31.60	25.87	33.94	33.90	33.05	**34.80**†
Micro-F1	56.44	48.21	57.25	57.20	57.17	**58.21**†
LSHTC-2011						
Macro-F1	25.11	19.12	26.56	26.34	25.69	**27.51**†
Micro-F1	41.23	36.52	43.22	43.42	43.73	**44.78**†

Table 3: Comparison of Macro-F1 and Micro-F1 for, (i) l_2−SVM, (ii) l_1−SVM (iii) CS−MSVM (iv) Group−Lasso, (v) State-of-the-art HR−SVM, and (vi) Proposed method - TerseSVM. The best approach for every row is shown in **bold**. The significance-test results are denoted for p-value less than 5%.

The comparison with other baselines such as which use the taxonomy information such as Pachinko-machine based top-down SVM [4] and orthogonal transfer [26] has been done in [13], and hence this is not repeated here. Furthermore, it may be noted that the loss function is kept the same in l_1−SVM, l_2−SVM, Group−Lasso and TerseSVM, the only difference being in using a different regularizer.

5 Experimental Analysis

5.1 Test set performance Table 3 shows the comparison of these methods on 6 different datasets based on the Macro-F1 and Micro-F1 measures. For comparing the proposed methods TerseSVM to HR−SVM, Micro-level sign test (s-test) and macro-level t-test were used for significance testing for Micro-F1 and Macro-F1 measures respectively [27]. Below are the crucial observations regarding generalization performance of various approaches:

- On both Micro-F1 and Macro-F1, the proposed TerseSVM method performs better than all the baselines and state-of-the-art HR−SVM method on all datasets, except the **NEWS20**. **NEWS20** dataset is much smaller and more balanced as compared all other datasets. However, the performance of the proposed TerseSVM is still quite competitive to the best performing Group−Lasso. More importantly, as shown in (first row, last column) of Table 4, this performance is achieved at only 19% of the parameters than used by Group−Lasso .

- For Macro-F1 measure, wherein each category is weighted equally, the improvement achieved by TerseSVM is relatively substantial indicating improvement in the detection of rare categories.

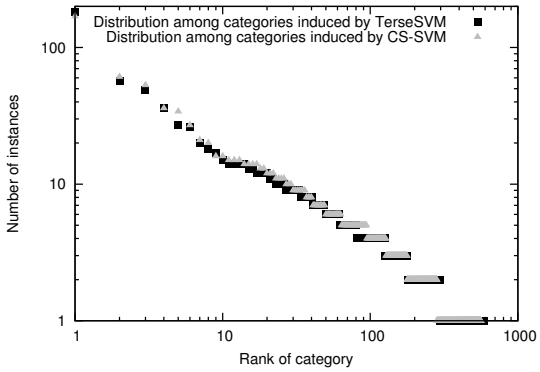

Figure 2: Comparison of distribution among categories induced on the test set by TerseSVM and CS−SVM. The X and Y axes are on logarithmic scale.

- This is also demonstrated in Figure 2 showing the comparison of distribution of categories in the test set for the **LSHTC-small** dataset. For larger-categories (on the left side of the graph), the black curve lies under the gray one which indicates a reduction in false-positive for these categories. For the rare categories (on the right side of the graph), the tail is much longer for the black graph indicating better detection of rare categories. The proposed method is able to detect about 5% more categories in the test set compared to the CS−SVM method.

Our method also performs better than the re-ranking framework proposed in [3], which is mainly a heuristic approach

Dataset	l_2-SVM	l_1-SVM	CS-MSVM	Group-Lasso	RMLR	TerseSVM
NEWS20						
Sparsity Ratio	0.97	0.02	0.55	0.99	0.97	**0.19**
Model-train ratio	0.66	0.006	0.35	0.68	0.66	**0.16**
IPC						
Sparsity Ratio	1	0.0002	0.11	1	0.99	**0.011**
Model-train ratio	35.9	0.001	11.3	24	39.2	**0.9**
LSHTC-small						
Sparsity Ratio	1	0.004	0.21	0.95	1	**0.015**
Model-train ratio	72.5	0.02	15.2	40	73.7	**0.8**
LSHTC-large						
Sparsity Ratio	0.95	0.0003	0.08	0.98	0.96	**0.004**
Model-train ratio	134.0	0.002	23.1	123.0	145.1	**0.7**
LSHTC-2012						
Sparsity Ratio	0.99	0.0004	0.03	0.96	0.98	**0.001**
Model-train ratio	91.1	0.006	17.8	119.2	138.2	**0.5**
LSHTC-2011						
Sparsity Ratio	0.97	0.0001	0.1	0.95	0.96	**0.002**
Model-train ratio	105.4	0.004	15.3	132.4	129.6	**0.6**

Table 4: Comparison of ratio of model size to training data size and sparsity structure. Except for l_1-SVM, the model with most sparsity and least model-to-train ratio is shown in bold

based on post-processing the probability scores obtained from the classifier on the test instances. It is also worth noting that the rare category problem associated to distribution of instances among categories is more accute than the problem of imbalanced classification [8]. In our scenario, most of the categories span a very low-dimensional sub-space of the entire feature space, and applying classical methods of imbalanced classification does not yield any improvement.

5.2 Compressing 16.8GB of information in 1-bit It is important to note that the better performance of our proposed method on **LSHTC-large** dataset was obtained with only 0.4% parameters i.e., only 4 in 1,000 parameters were useful. In other words, better performance was achieved by using a model consisting of only 18 Million entries instead of using a model consisting of 4 Billion entries as suggested in [12]. Furthermore, the model size was compressed from 17GB (as shown in Table 1) to 200MB, thereby compressing 16.8GB of information in 1-bit. Similar results were obtained for all the large-scale **LSHTC-*** datasets. This confirms our initial observations that most of the values stored in such learnt models are spurious.

For further analysis, we compare two quantities for data-algorithm pair : (i) sparsity ratio which is $\frac{\#\text{non-zero parameters}}{\#\text{Total number of parameters}}$ where Total number of parameters = $\#\text{categories} \times \#\text{features}$, and (ii) ratio of trained model size to the training data size. Sparsity is a measure of the redundant information stored by methods such as l_2-SVM and RMLR. It must be noted that since the HR$-$SVM method stores the weight vectors of the internal nodes as well, hence its space complexity is much higher than l_1-SVM. The crucial observations from results in Table 4 are :

- This sparsity inducing property lets us to reduce the model size by upto two orders of magnitude as in datasets compared to RMLR.
- The model sizes learnt by TerseSVM are of the same order as the training data size. This is contrast to RMLR in Table 1 wherein the model size was upto bigger than training data size by upto two order of magnitude.

In the light of generalization performance in Table 3 and insights into the sparsity structure of large-models as shown in Table 4, it is clear that the mixed-norm regularization is better at capturing the statistical properties of large-scale sparse data. The ratio of model size to training data size shows that model learnt using this regularization scheme is also consistent from an information-theoretic viewpoint.

5.3 Training Time We used OpenMP for concurrent training which can be incorporated in the Liblinear solver. Based on the number of cores per node, the tasks were split such that as many cores as possible could be used. The trained models from different nodes were then concatenated to yield the single model for prediction. Depending on the work-load, our application was allotted between 300 to 2500 cores for computation. The training time was approximately 1 minute for **LSHTC-small**, 2 hours for **LSHTC-large**, and 4 hours for **LSHTC-2011**.

5.4 Overall Comparison In Table 5, we show the comparison of various algorithms on different evaluations measures which include, (i) Solution Sparsity, (ii) Generalization Performance and detection of rare categories, and (iii) Concurrent training. Clearly, the proposed mixed-norm regularization based algorithm TerseSVM exhibits all the three desirable properties simultaneously. As a result, it is more suitable to handle multi-class classification involving large

Dataset	l_2-SVM	l_1-SVM	CS-MSVM	Group-Lasso	HR-SVM	TerseSVM
Solution Sparsity	✗	✓	✗	✗	✗	✓
Accuracy	✗	✗	-	-	-	✓
Concurrent Training	✓	✓	✗	✓	✓	✓

Table 5: Comparison of different methods based on the criterion considered. The better suitability of a method to a metric is indicate with a ✓, non-suitability is marked with ✗, and improvable behavior is indicated with −.

number of target categories as compared most off-the-shelf solvers and state-of-the-art methods.

6 Conclusion and Future work

In large-scale high dimensional scenarios, the difference between true knowledge, i.e., parameters, and noise can become obscured if one neglects the nature of underlying data distribution [12, 13]. In this work, we have shown that the true knowledge may just be hidden in a significantly small fraction of the entire model obtained using off-the-shelf packages. By incorporating mixed-norm regularization, our proposed method TerseSVM is able to learn better making the model more compact with better generalization performance compared to the state-of-the-art methods. Furthermore, our method enjoys concurrent training easily scales for classification problems involving 30,000 target categories. We would like to extend this framework to address multi-label problems in large-scale scenarios. Another research direction is to explore synergies with recent efficient methods for learning sparse linear models [18] and theoretical developments in [17].

References

[1] R. Agrawal, A. Gupta, Y. Prabhu, and M. Varma. Multi-label learning with millions of labels: Recommending advertiser bid phrases for web pages. In *World Wide Web Conference*, 2013.

[2] R. Babbar, C. Metzig, I. Partalas, E. Gaussier, and M.-R. Amini. On power law distributions in large-scale taxonomies. *ACM SIGKDD Explorations Newsletter*, 16(1):47–56, 2014.

[3] R. Babbar, I. Partalas, E. Gaussier, and M.-R. Amini. Re-ranking approach to classification in large-scale power-law distributed category systems. In *ACM SIGIR*, 2014.

[4] R. Babbar, I. Partalas, C. Metzig, E. Gaussier, and M.-R. Amini. Comparative classifier evaluation for web-scale taxonomies using power law. In *European Semantic Web Conference*, 2013.

[5] F. Bach, R. Jenatton, J. Mairal, and G. Obozinski. Convex optimization with sparsity-inducing norms. *Optimization for Machine Learning*.

[6] S. Bengio, J. Weston, and D. Grangier. Label embedding trees for large multi-class tasks. In *NIPS*, 2010.

[7] K. Bhatia, H. Jain, P. Kar, M. Varma, and P. Jain. Sparse local embeddings for extreme multi-label classification. In *NIPS*, 2015.

[8] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: synthetic minority over-sampling technique. *JAIR*, 2002.

[9] P. L. Combettes and J.-C. Pesquet. Proximal splitting methods in signal processing. In *Fixed-point algorithms for inverse problems in science and engineering*, pages 185–212. Springer, 2011.

[10] K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *JMLR*, 2002.

[11] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. Liblinear: A library for large linear classification. *JMLR*, 2008.

[12] S. Gopal and Y. Yang. Distributed training of large-scale logistic models. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 289–297, 2013.

[13] S. Gopal and Y. Yang. Recursive regularization for large-scale classification with hierarchical and graphical dependencies. In *SIGKDD*, 2013.

[14] M. Kowalski. Sparse regression using mixed norms. *Applied and Computational Harmonic Analysis*, 27(3):303–324, 2009.

[15] M. Kowalski, M. Szafranski, and L. Ralaivola. Multiple indefinite kernel learning with mixed norm regularization. In *ICML*, 2009.

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[17] Y. Lei, U. Dogan, A. Binder, and M. Kloft. Multi-class svms: From tighter data-dependent generalization bounds to novel algorithms. In *NIPS*, 2015.

[18] Z. C. Lipton and C. Elkan. Efficient elastic net regularization for sparse linear models. *arXiv preprint arXiv:1505.06449*, 2015.

[19] P. Mineiro and N. Karampatziakis. Fast label embeddings for extremely large output spaces. *arXiv preprint arXiv:1412.6547*, 2014.

[20] J.-J. Moreau. Proximité et dualité dans un espace hilbertien. *Bulletin de la Société mathématique de France*, 93:273–299, 1965.

[21] N. Parikh and S. Boyd. Proximal algorithms. *Foundations and Trends in Optimization*, 1(3):127–239, 2014.

[22] Y. Prabhu and M. Varma. Fastxml: A fast, accurate and stable tree-classifier for extreme multi-label learning. In *SIGKDD*, August 2014.

[23] R. Rifkin and A. Klautau. In defense of one-vs-all classification. *The Journal of Machine Learning Research*, 2004.

[24] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[25] B. Z. Vatashsky and K. Crammer. Multi class learning with individual sparsity. In *IJCAI*, 2013.

[26] L. Xiao, D. Zhou, and M. Wu. Hierarchical classification via orthogonal transfer. In *ICML, 2011*.

[27] Y. Yang, J. Zhang, and B. Kisiel. A scalability analysis of classifiers in text categorization. In *SIGIR*, 2003.