Large-Scale Learning with Structural Kernels for Class-Imbalanced Datasets

Aliaksei Severyn and Alessandro Moschitti

Department of Computer Science and Engineering
University of Trento
Via Sommarive 5, 38123 POVO (TN) - Italy
{severyn,moschitti}@disi.unitn.it

Abstract. Much of the success in machine learning can be attributed to the ability of learning methods to adequately represent, extract, and exploit inherent structure present in the data under interest. Kernel methods represent a rich family of techniques that harvest on this principle. Domain-specific kernels are able to exploit rich structural information present in the input data to deliver state of the art results in many application areas, e.g. natural language processing (NLP), bio-informatics, computer vision and many others. The use of kernels to capture relationships in the input data has made Support Vector Machine (SVM) algorithm the state of the art tool in many application areas. Nevertheless, kernel learning remains a computationally expensive process. The contribution of this paper is to make learning with structural kernels, e.g. tree kernels, more applicable to real-world large-scale tasks. More specifically, we propose two important enhancements of the approximate cutting plane algorithm to train Support Vector Machines with structural kernels: (i) a new sampling strategy to handle class-imbalanced problem; and (ii) a parallel implementation, which makes the training scale almost linearly with the number of CPUs. We also show that theoretical convergence bounds are preserved for the improved algorithm. The experimental evaluations demonstrate the soundness of our approach and the possibility to carry out large-scale learning with structural kernels.

Keywords: Machine Learning, Kernel Methods, Structural Kernels, Support Vector Machine, Natural Language Processing

1 Introduction

Different domain-specific kernels have been successfully applied to various Natural Language Processing (NLP) tasks, e.g. [10, 13, 9, 1]. However, previous work scales poorly to the real-world datasets, where the number of examples is typically in the order of millions. Indeed, kernel methods require to carry out learning in dual spaces, where training complexity is typically quadratic in the number of instances.

To reduce such training time [16] proposed an approximate version of the cutting plane algorithm (CPA) [14] for training SVMs with general kernels. [12] showed that the same algorithm can be successfully applied to train SVMs with structural kernels on very large data obtaining speedup up factors up to 10. These studies employ 1-slack optimization problem reformulation [5], which is much faster than conventional cutting plane methods on large-scale datasets and produces sparser solutions.

Unfortunately, the 1-slack reformulation prevents to accomplish cost-sensitive classification using a standard approach in SVMs, i.e. outweighing the positive or negative errors. This is a critical drawback for applications in NLP where data is often imbalanced, which requires optimization of Precision/Recall measures.

In this paper, we provide two important improvements of the approximate CPA that enable the use of structural kernels, e.g. tree kernels, for large-scale learning: (i) an effective and sound method for tuning up Precision and Recall on imbalanced datasets and (ii) parallelization of the training algorithm improving its scalability even further. Regarding the application side, we show that our method allows for experimenting with tree kernels on very large real-world datasets such as Yahoo! Answers.

The experimental results confirm the validity of our approach as (i) it greatly outperforms previous approximate CPA when tuning of P/R is needed and (ii) it still matches the F$_1$-score of exact SVMs. Regarding the running time evaluation: our approach is as fast as CPA with sampling and, when parallelized, the speedup scales almost linearly with the number of available CPUs.

2 Cutting Plane Algorithm with Sampling

Let us consider an equivalent reformulation of SVM QP training problem, known as a 1-slack reformulation, which produces a much more efficient version of the CPA [5]:

$$\begin{aligned} &\underset{\boldsymbol{w},\xi\geq 0}{\text{minimize}} && \frac{1}{2}\|\boldsymbol{w}\|^2 + C\xi \\ &\underset{\forall \boldsymbol{c}\in\{0,1\}^n}{\text{subject to}} && \frac{1}{n}\sum_{i=1}^{n} c_i y_i \boldsymbol{w}\cdot\boldsymbol{x}_i \geq \frac{1}{n}\sum_{i=1}^{n} c_i - \xi, \end{aligned} \tag{1}$$

where each vector $\boldsymbol{c} \in \{0,1\}^n$ forms all possible linear combinations of the classical constraints $y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i) \geq 1 - \xi_i$.

The key benefit of this reformulation is that there is only a single slack variable ξ that is now shared across all the constraints. Even though the number of constraints swelled up to 2^n, the cutting plane method (Alg. 1) requires only a sufficient subset of constraints S to solve the problem (1). It works by iteratively resolving QP (line 4) over the current set S and adding a new constraint $\boldsymbol{c}^{(t)}$ violated the most by the current solution $\boldsymbol{w}$ (lines 5-7) until no constraints are violated by more than ϵ (line 10).

When using kernels, examples are mapped to the feature space via a mapping $\phi(\cdot)$ and finding the most violated constraint (lines 5-7) involves computing an

Algorithm 1 Cutting Plane Algorithm (primal)

```
1: Input: $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)$, $C$, $\epsilon$Alg.
2: $S \leftarrow \emptyset; t = 0$
3: repeat
4:     $(\boldsymbol{w}, \xi) \leftarrow$ optimize (1) over the constraints in $S$
       /* find a cutting plane */
5:     for $i = 1$ to $n$ do
6:         $c_i^{(t)} \leftarrow \begin{cases} 1 \; y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i) \leq 1 \\ 0 \; otherwise \end{cases}$
7:     end for
       /* add a constraint to the set of constraints */
8:     $S \leftarrow S \cup \{\boldsymbol{c}^{(t)}\}$
9:     $t = t + 1$
10: until $\frac{1}{n} \sum_{i=1}^{n} c_i^{(t)} (1 - y_i \boldsymbol{w} \cdot \boldsymbol{x}_i) \leq \xi + \epsilon$
11: return $w, \xi$
```

inner product between the weight vector and each training example: $\boldsymbol{w} \cdot \phi(\boldsymbol{x}_i)$. In the dual space, where $\boldsymbol{w}$ expands over the dual variables $\boldsymbol{\alpha}$, this inner-product renders as:

$$\boldsymbol{w} \cdot \phi(\boldsymbol{x}_i) = \sum_{k=1}^{n} \Big(\sum_{t=1}^{|S|} \frac{1}{n} \alpha_t c_k^{(t)} y_k \Big) K(\boldsymbol{x}_i, \boldsymbol{x}_k), \tag{2}$$

where $K(\boldsymbol{x}_i, \boldsymbol{x}_k) = \phi(\boldsymbol{x}_i) \cdot \phi(\boldsymbol{x}_i)$ is a kernel[1]. Computing (2) for each training example requires $O(n^2)$ kernel evaluations which makes the CPA training of non-linear SVMs no better than conventional decomposition methods.

To address this limitation [16] proposed to approximate the expensive computation of the most violated constraint over the full set of training examples n by using a smaller sample r. In this case the expensive double sum of kernel evaluations at each iteration reduces from $\sum_{i,j=1}^{n} K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ to a more tractable: $\sum_{i,j=1}^{r} K(\boldsymbol{x}_i, \boldsymbol{x}_j)$, such that the most violated constraint is effectively computed over a smaller set of examples uniformly sampled from the original training set. Even though at each step we compute only an approximation of the exact cutting plane, the sampling approach has been shown to provide accurate solutions and converge in a finite number of steps irrespective of the training set size.

3 Improving CPA with sampling

In this section we present two improvements to the CPA with sampling: (i) we propose an alternative sampling strategy that is effective for tuning up Precision and Recall and (ii) we parallelize the training algorithm.

[1] due to the space constraints, for a more careful treatment of the dual version of CPA with kernels we refer the reader to [8] or [12].

3.1 Sampling Strategy for Imbalanced Data

To address the problem of the imbalanced data one idea can be to use different penalty factors [15] C^+ and C^- for examples from positive and negative classes. This modification is easy to incorporate into the standard soft-margin SVM formulation where we have individual slack variables ξ_i for each constraint.

However, when using the 1-slack formulation (1), we have a single slack variable shared across all the constraints, while in the dual each α_i no longer corresponds to the individual example but to a linear combination of examples. This makes the task of controlling class imbalance through different margin parameters C^+ and C^- non-trivial. On the other hand the idea of sampling to approximate (2) at each iteration suggests a straight-forward solution. Instead of uniformly sampling r examples to compute the most violated constraint at each step, we can use cost-proportionate rejection sampling technique. This

Algorithm 2 Rejection sampling

```
1: Pick example $(\boldsymbol{x_i}, y_i, q_i)$ at random
2: Flip a coin with bias $q_i/q'$
3: if *heads* then
4:     keep the example
5: else
6:     discard it
7: end if
```

technique is presented in Alg. 2, where q_i is the importance weight of the i-th example and q' is an upper bound on any importance value in the dataset. This process is repeated until we sample the required number of examples. This modification enables the control over the proportion of examples from different classes that will form a sample used to compute the most violated constraint. Unlike the conventional approaches for addressing the class-imbalance problem, that either under-sample the majority class or over-sample the minority class from the training data, the rejection sampling coupled with cutting plane algorithm does not discard any examples from the training set. At each iteration we form a sample according to the pre-assigned importance weights for each example, such that examples from both the majority and minority classes enter the sample in the desired proportion. This process is repeated until the algorithm converges. So no information is lost during the optimization process.

Another benefit of this approach is that by increasing the importance weight of the minority class, we give its examples more chance to end up in the most violated constraint and hence, become potential support vectors. This way the imbalanced support-vector ratio is automatically tuned to include more examples from the minority class, which gives more control over the imbalance of classes.

It can be easily shown that the new sampling technique preserves the convergence bounds proven in [16]. Note that drawing examples using rejection sampling (Alg. 2) simply re-weights the original distribution D according to the importance weights of the examples. This means that we are effectively training a cost insensitive classifier under the new transformed distribution $\hat{D}$. By invoking Translation Theorem [17], we establish that, to obtain a cost-sensitive classifier that minimizes the expected risk under the original distribution D, it is sufficient to learn a cost-insensitive classifier under the transformed distribution $\hat{D}$. The CPA that draws examples from D using the sampling scheme in Alg. 2 is equivalent to the original CPA applying uniform sampling to the transformed distribution $\hat{D}$. This allows us to invoke the proof in [16], thus establishing similar convergence bounds.

3.2 Parallelization

The modular nature of the cutting plane algorithm suggests easy parallelization. In fact, in our experiments we observed that at each iteration 95% of the total learning time is spent in the double loop (2), which involves double sum of kernel evaluations over r examples in the sample. This observation suggests high parallelizability of the code. Using p processors the complexity of this predominant part can be brought down from $O(r^2)$ to $O(r^2/p)$.

4 Experimental evaluation

The goal of our experiments is to study how the problem of imbalanced datasets can be effectively tackled by the new sampling technique that we propose to integrate into the CPA. To do so, we carry out learning on complex text classification tasks where addressing class-imbalance problem plays an important role to obtain the optimal classification performance. In the first set of experiments we compare the accuracy one can get by better parametrizing the model using our proposed method against the cutting plane algorithm with uniform sampling and the conventional SVM. Below we refer to the capability to control the penalty factors for examples from different classes as simply j option (as implemented in SVM-light software). Secondly, we bring the capability of cutting plane algorithm with rejection sampling to alleviate the class-imbalance problem to the large-scale, where training of conventional SVMs soon becomes too time-consuming. We also demonstrate the speedup factors after parallelization. This feature becomes especially appealing nowadays, when shared memory parallel architectures, i.e. multi-processor and multi-core CPUs, are becoming available for general use.

We modified the implementation of the approximate CPA with uniform sampling[16] with SVM-Light-TK[11] to include cost-proportionate sampling strategy. For brevity, we refer to the original CPA with uniform sampling as uSVM, CPA + rejection sampling as uSVM+j, and SVM-light as SVM. In all our experiments we used the subset tree (SST) kernel [2]. For uSVM+j and SVM we report the best results for the optimal value of j parameter that controls Precision/Recall ratio.To measure the classification performance we use Precision, Recall and F_1-score. All the experiments were run on machines equipped with Intel® Xeon® 2.33GHz CPUs carrying 6Gb of RAM under Linux.

We used two different natural language datasets: TREC 10 QA[2] (training: 5,483, test: 500) and Yahoo! Answers (YA)[3](train: up to 300k, test: 10k) to perform two similar tasks of QA classification. The task for the first dataset is to select the most appropriate type of the answer from a set of given possibilities. The training set consists of 5,483 questions and the test set is composed of 500 questions for each class. The goal of the experiments on these relatively small datasets is to demonstrate that rejection sampling is able to effectively handle class imbalance similar to SVM.

The second corpus is a large subset of Yahoo! Answers dataset. The dataset contains a set of 142,627 non-factoid, i.e. "how to" questions, and 364,419 answers. Testing was carried out on the 10k subset. The classification task was set up as follows. Given pairs of questions and corresponding answers learn if in a given pair the answer is the 'best' answer for a question. The goal of this experiment is to have a large-scale classification task (300k examples in our experiments) to demonstrate that class-imbalance problem can be handled effectively at this scale.

Results on TREC 10 and YA. Experimental results on six different categories of TREC corpus and on YA dataset are reported in Table 1(a) and Table 1(b) respectively. One can see that uSVM algorithm with uniform sampling obtains high precision trying to minimize the training error dominated by examples from negative class and is not able to adjust in the presence of class imbalance. This results in lower values of the recall. On the other hand uSVM+j is able to achieve a better tradeoff between precision and recall resulting in higher F_1 scores. Also the P/R ratio of SVM with the optimal set of parameters suggests that uSVM+j has a better capacity to control the imbalance problem.

(a)

Data	Ratio	uSVM		uSVM+j		SVM	
		F-1	P/R	F-1	P/R	F-1	P/R
ABBR	60:1	87.5	100.0/77.8	84.2	80.0/88.9	84.2	80.0/88.9
DESC	4:1	96.1	95.0/97.1	96.1	95.0/97.1	94.8	97.7/92.0
ENTY	3:1	72.3	91.8/59.6	79.1	79.6/78.7	80.4	82.2/78.7
HUM	3:1	88.1	98.1/80.0	90.3	94.9/86.2	87.5	88.9/86.2
LOC	5:1	81.4	96.6/70.4	87.0	87.5/86.4	82.6	86.5/79.0
NUM	5:1	86.0	98.9/76.1	91.2	96.1/86.7	89.9	98.9/82.3

(b)

10k	1.5:1	37.4	33.5/42.2	39.1	29.6/57.7	37.9	24.2/87.7
50k	2.0:1	36.5	36.0/36.9	40.6	30.0/62.5	39.6	25.7/86.9
100k	2.4:1	33.4	36.2/31.1	40.2	30.2/59.9	40.3	26.6/83.5
150k	2.8:1	33.5	36.9/30.7	41.0	30.2/64.0	-	-
300k	3.4:1	23.8	40.1/16.9	41.4	30.7/63.8	-	-

Fig. 1. Results on TREC-10 (a) YA (b) datasets. Ratio - proportion of negative examples with respect to positive; P/R - precision (P) and recall (R).

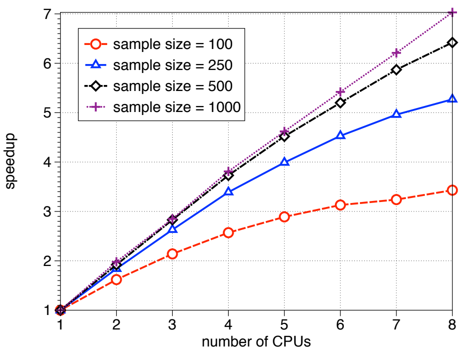

Fig. 2. Speedups vs number of CPUs after parallelization of CPA on Yahoo! Answers (50k).

[2] http://l2r.cs.uiuc.edu/cogcomp/Data/QA/QC/
[3] retrieved through the Yahoo! Webscope program.

Parallelization. To test the effects of parallelization we carried out experiments on 50,000 subset of YA dataset on 1 to 8 CPUs. The achieved speedups are reported in Fig. 2, where each curve corresponds to training using different sample sizes. Increasing the sample size leads to an increase in the time spent inside the double loop (2), which makes the speedup for larger sample sizes even more significant. Using 8 CPUs gives the speedup factor of about 7.0 using sample size equal to 1000. Since classification can also be easily parallelized it allows one to experiment with larger sample sizes to obtain a more accurate model or carry out training on larger data.

To better demonstrate the advantage of the parallel implementation we replicated the large-scale experiment in [12] on Semantic Role Labeling dataset[4] using 1 million examples. The reported training time was 4 hours for uSVM and 7.5 days for SVM, while our parallel implementation took about 30 minutes for learning on 8 CPUs.

5 Related work

The most popular method to address class-imbalance problem in SVMs is to introduce cost factors in the primal problem [15]. It is implemented in SVM-light [4] that has a super-linear scaling behavior, which prohibits the experiments on very large datasets.

To improve the scaling properties of SVM-light, a number of CPA-based methods have been proposed (for example, SVM^{perf} [5]). [3] have further improved the convergence rate of the underlying CPA. Another approach to directly optimize for F_1-score, was proposed in [7]. While the aforementioned algorithms deliver fast and accurate solutions, they scale well only when linear kernels are used. Another approach to iteratively extract basis vectors as a part of a cutting plane algorithm is studied in [6]. This, however, leads to a non-trivial optimization problem when arbitrary kernel functions are used.

6 Conclusions

In this paper we proposed a method that combines the benefits of CPA with sampling for training non-linear SVMs on large-scale data together with the flexibility to control the problem of imbalanced data. This improvement becomes particularly significant when learning on large text classification datasets, where class-imbalance plays an important role to obtain the optimal balance between precision and recall. The proposed sampling strategy has shown superior ability to parametrize the model with respect to conventional approach implemented in SVM-light on two Question/Answer classification tasks. We also take advantage of the possibility to parallelize the code to make learning even faster.

The distinctive property of the proposed method is that it directly integrates the cost-proportionate sampling into the CPA optimization process, unlike the

[4] http://danielepighin.net/ cms/research/MixedFeaturesForSRL

other sampling approaches based on the reductions idea of [17]. In other words, sampling is carried out iteratively, such that no information is discarded from training examples as in “one-shot” sampling methods.

Acknowledgements

This work has been partially supported by the EC project FP247758: Trustworthy Eternal Systems via Evolving Software, Data and Knowledge (EternalS).

References

1. Cancedda, N., Gaussier, E., Goutte, C., Renders, J.M.: Word sequence kernels. Journal of Machine Learning Research 3, 1059–1082 (2003)
2. Collins, M., Duffy, N.: New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In: ACL. pp. 263–270 (2002)
3. Franc, V., Sonnenburg, S.: Optimized cutting plane algorithm for support vector machines. In: ICML. pp. 320–327 (2008)
4. Joachims, T.: Making large-scale SVM learning practical. In: Advances in Kernel Methods - Support Vector Learning, chap. 11, pp. 169–184. MIT Press, Cambridge, MA (1999)
5. Joachims, T.: Training linear SVMs in linear time. In: KDD (2006)
6. Joachims, T., Yu, C.N.J.: Sparse kernel svms via cutting-plane training. Machine Learning 76(2-3), 179–193 (2009), european Conference on Machine Learning (ECML) Special Issue
7. Joachims, T.: A support vector method for multivariate performance measures. In: ICML. pp. 377–384 (2005)
8. Joachims, T., Finley, T., Yu, C.N.J.: Cutting-plane training of structural svms. Machine Learning 77(1), 27–59 (2009)
9. Kate, R.J., Mooney, R.J.: Using string-kernels for learning semantic parsers. In: ACL (July 2006)
10. Kudo, T., Matsumoto, Y.: Fast methods for kernel-based text analysis. In: Proceedings of ACL’03 (2003)
11. Moschitti, A.: Making tree kernels practical for natural language learning. In: EACL. The Association for Computer Linguistics (2006)
12. Severyn, A., Moschitti, A.: Large-scale support vector learning with structural kernels. In: ECML/PKDD (3). pp. 229–244 (2010)
13. Shen, L., Sarkar, A., Joshi, A.k.: Using LTAG Based Features in Parse Reranking. In: Proceedings of EMNLP’06 (2003)
14. Tsochantaridis, I., Joachims, T., Hofmann, T., Altun, Y.: Large margin methods for structured and interdependent output variables. Journal of Machine Learning Research 6, 1453–1484 (2005)
15. Veropoulos, K., Campbell, C., Cristianini, N.: Controlling the sensitivity of support vector machines. In: Proceedings of the International Joint Conference on AI. pp. 55–60 (1999)
16. Yu, C.N.J., Joachims, T.: Training structural svms with kernels using sampled cuts. In: KDD. pp. 794–802 (2008)
17. Zadrozny, B., Langford, J., Abe, N.: Cost-sensitive learning by cost-proportionate example weighting. In: Proceedings of ICDM (2003)