Contents

Todo list

Chapter 2

EnsembleSVM: A Library for Ensemble Learning Using Support Vector Machines

{ch:ensemblesvm}

This chapter has been previously published as:

Abstract

`EnsembleSVM` is a free software package containing efficient routines to perform ensemble learning with support vector machine (SVM) base models. It currently offers ensemble methods based on binary SVM models. Our implementation avoids duplicate storage and evaluation of support vectors which are shared between constituent models. Experimental results show that using ensemble approaches can drastically reduce training complexity while maintaining high predictive accuracy. The `EnsembleSVM` software package is freely available online at `http://esat.kuleuven.be/sista/ensemblesvm`.

2.1 Introduction

Data sets are becoming increasingly large. Machine learning practitioners are confronted with problems where the main computational constraint is the amount of time available. Problems become particularly challenging when the training sets no longer fit into memory. Accurately solving the dual problem for SVM training with nonlinear kernels requires a run time which is at least quadratic in the size of the training set n, thus training complexity is $\Omega(n^2)$ [? ?].

`EnsembleSVM` employs a divide-and-conquer strategy by aggregating many SVM models, trained on small subsamples of the training set. Through subdivision, total training time decreases significantly, even though more models need to be trained. For example, training p classifiers on subsamples of size n/p, results in an approximate complexity of $\Omega(n^2/p)$. This reduction in complexity helps in dealing with large data sets and nonlinear kernels.

Ensembles of SVM models have been used in various applications [? ? ?]. Collobert et al. [?] use ensembles for large scale learning and employ a neural network to aggregate base models. Valentini and Dietterich [?] provide an implementation which allows base models to use different kernels. For efficiency reasons, we require base models to share a single kernel function.

While other implementations mainly focus on improving predictive performance, our framework primarily aims to (i) make nonlinear large-scale learning feasible through complexity reductions and (ii) enable fast prototyping of novel ensemble algorithms.

2.2 Software Description

The `EnsembleSVM` software is freely available online under a LGPL license. `EnsembleSVM` provides ensembles of instance-weighted SVMs, as defined in Equation (2.1). The default approach we offer is bagging, which is commonly used to improve the performance of unstable classifiers [?]. In bagging, base models are trained on bootstrap subsamples of the training set and their predictions are aggregated through majority voting.

Base model flexibility is maximized by using instance-weighted binary support vector machine classifiers, as defined in Equation (2.1). This formulation lets users define misclassification penalties per training instance $C_i,\ i = 1, \ldots, n$ and encompasses popular approaches such as C-SVC and class-weighted SVM

[? ?].

{weightedsvm}

$$\min_{\mathbf{w},\xi,\rho} \frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{i=1}^{n} C_i\xi_i, \tag{2.1}$$

$$\text{subject to } y_i(\mathbf{w}^T\phi(\mathbf{x}_i)+\rho) \geq 1-\xi_i, \qquad i=1,\ldots,n$$

$$\xi_i \geq 0, \qquad i=1,\ldots,n$$

When aggregating SVM models, the base models often share support vectors (SVs). The EnsembleSVM software intelligently caches distinct SVs to ensure that they are only stored and used for kernel evaluations once. As a result, EnsembleSVM models are smaller and faster in prediction than ensemble implementations based on wrappers.

2.2.1 Implementation

EnsembleSVM has been implemented in C++ and makes heavy use of the standard library. The main implementation focus is training speed. We use facilities provided by the C++11 standard and thus require a moderately recent compiler, such as gcc $\geq$ 4.7 or clang $\geq$ 3.2. A portable Makefile system based on GNU autotools is used to build EnsembleSVM.

EnsembleSVM interfaces with LIBSVM to train base models [?]. Our code must be linked to LIBSVM but does not depend on a specific version. This allows users to choose the desired version of the LIBSVM software in the back-end.

The EnsembleSVM programming framework is designed to facilitate prototyping of ensemble algorithms using SVM base models. We particularly provide extensive support to define novel aggregation schemes, should the available options be insufficient. Key components are extensively documented in the code and on a wiki, which serves as a high-level guideline.[1] Intuitive APIs are provided for convenient features such as thread pools, command line interface and deserialization to enable users to develop new tools efficiently.

[1]The EnsembleSVM development wiki is available at https://github.com/claesenm/EnsembleSVM/wiki.

The `EnsembleSVM` library was built with extensibility and user contributions in mind. Major API functions are well documented to lower the threshold for external development. The executable tools provided with `EnsembleSVM` are essentially wrappers for the library itself. The tools can be considered as use cases of the main API functions to help developers.

2.2.2 Tools

The main tools in this package are `esvm-train` and `esvm-predict`, used to train and predict with ensemble models. Both of these are pthread-parallelized. Additionally, the `merge-models` tool can be used to merge standard `LIBSVM` models into ensembles. Finally, `esvm-edit` provides facilities to modify the aggregation scheme used by an ensemble.

`EnsembleSVM` includes a variety of extra tools to facilitate basic operations such as stratified bootstrap sampling, cross-validation, replacing categorical features by dummy variables, splitting data sets and sparsifying standard data sets. We recommend retaining the original ratio of positives and negatives in the training set when subsampling.

2.3 Benchmark Results

{bench}

To illustrate the potential of our software, `EnsembleSVM` 2.0 has been benchmarked with respect to `LIBSVM` 3.17. To keep the experiments simple, we use majority voting to aggregate predictions, even though more sophisticated methods are offered. For reference, we also list the best obtained accuracy with a linear model, trained using `LIBLINEAR` [?]. Linear methods are common in large-scale learning due to their speed, but may result in significantly decreased accuracy. This is why scalable nonlinear methods are desirable.

We used two binary classification problems, namely the `covtype` and `ijcnn1` data sets.[2] Both data sets are balanced. Features were always scaled to $[0, 1]$. We have used C-SVC as SVM and base models ($\forall\ i : C_i = C$). Reported numbers are averages of 5 test runs to ensure reproducibility. We used the RBF kernel, defined by the kernel function $\kappa(\mathbf{x}_i, \mathbf{x}_j) = e^{-\gamma||\mathbf{x}_i - \mathbf{x}_j||^2}$. Optimal parameter selection was done through cross-validation.

The `covtype` data set is a common classification benchmark featuring 54 dimensions [?]. We randomly sampled balanced training and test sets of

[2] Available at: `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html` and UCI.

100,000 and 40,000 instances respectively and classified class 2 versus all others. The `ijcnn1` data set was used in a machine learning challenge during IJCNN 2001 [**?**]. It contains 35,000 training instances in 22 dimensions.

data set	**test set accuracy**			**no. of SVs**		**time (s)**	
	LIBSVM	LIBLINEAR	ESVM	LIBSVM	ESVM	LIBSVM	ESVM
`covtype`	0.92	0.76	0.89	26516	50590	728	35
`ijcnn1`	0.98	0.92	0.98	3564	7026	9.5	0.3

Table 2.1: Summary of benchmark results per data set: test set accuracy, number of support vectors and training time. Accuracies are listed for a single `LIBSVM` model, `LIBLINEAR` model and an ensemble model.

{resultstable}

Results in Table 2.1 show several interesting trends. Training `EnsembleSVM` models is orders of magnitude faster, because training SVMs on small subsets significantly reduces complexity. Subsampling induces smaller kernels per base model resulting in lower overall memory use.

Ensembles can end up with more support vectors than a single SVM. Due to our parallelized implementation, prediction with ensemble models was faster than with `LIBSVM` models in both experiments even though the ensembles comprise twice as many SVs.

The ensembles in these experiments are competitive with a traditional SVM even though we used simple majority voting. For `covtype`, ensemble accuracy is 3% lower than a single SVM and for `ijcnn1` the ensemble is marginally better (0.2%). Linear SVM falls far short in terms of accuracy for both experiments, but is trained much faster ($<$ 2 seconds).

We obtained good results with very basic aggregation. **?**] illustrated that more sophisticated aggregation methods can improve the predictive performance of ensembles. Others have reported performance improvements over standard SVM for ensembles using majority voting [**? ?**].

2.4 Conclusions

`EnsembleSVM` provides users with efficient tools to experiment with ensembles of SVMs. Experimental results show that training ensemble models is significantly faster than training standard `LIBSVM` models while maintaining competitive predictive accuracy.

Linear methods are frequently applied in large-scale learning, mainly due to their low training complexity. Linear methods are known to have competitive accuracy for high dimensional problems. As our benchmarks showed, the difference in accuracy may be large for low dimensional problems. As such, fast nonlinear methods remain desirable in large-scale learning, particularly for low dimensional tasks with many training instances. Our benchmarks illustrate the potential of the ensemble approaches offered by `EnsembleSVM`.

Ensemble performance may be improved by using more complex aggregation schemes. `EnsembleSVM` currently offers various aggregation schemes, both linear and nonlinear. Additionally, it facilitates fast prototyping of novel methods through its `Pipeline` framework.[3]

`EnsembleSVM` strives to provide high-quality, user-friendly tools and an intuitive programming framework for ensemble learning with SVM base models. The software will be kept up to date by incorporating promising new methods and ideas when they are presented in the literature. User requests and suggestions are welcome and appreciated.

[3]`https://github.com/claesenm/EnsembleSVM/wiki/Pipeline`

Bibliography

List of publications

Input file chapters/publications/publications.tex does not exist. Make sure its starts with “\chapter{List of publications}”. To not include this chapter in the table of contents, use the starred version of the \chapter command...

Instructies van de faculteit:

Lijst van de publicaties door de doctorandus/a (auteur of co-auteur).