Learning Support Vector Machines from Distributed Data Sources

Cornelia Caragea, Doina Caragea, and Vasant Honavar
Artificial Intelligence Research Laboratory, Department of Computer Science
Iowa State University, Ames, Iowa 50011-1040, USA
{cornelia,dcaragea,honavar}@cs.iastate.edu

Abstract

In this paper we address the problem of learning Support Vector Machine (SVM) classifiers from distributed data sources. We identify sufficient statistics for learning SVMs and present an algorithm that learns SVMs from distributed data by iteratively computing the set of sufficient statistics. We prove that our algorithm is *exact* with respect to its centralized counterpart and *efficient* in terms of time complexity.

Introduction

With the recent advances in technology, it is possible to gather and store large volumes of data. This data contain valuable information that can help people characterize specific domains and make predictions about new data from those domains.

Many machine learning algorithms have been proposed and have proven to be very effective in the case of centralized data. Formally, the problem of learning from centralized data can be summarized as follows (Mitchell 1997): Given a data set D, a hypothesis class H, and a performance criterion P, the learning algorithm L outputs a hypothesis $h \in H$ that optimizes P. In pattern classification applications, h is a classifier (e.g., a Naive Bayes classifier, a Decision Tree, a Support Vector Machine, etc.). The data D typically consists of a set of training examples. Each training example is an ordered tuple of attribute values, where one of the attributes corresponds to a class label and the remaining attributes represent inputs to the classifier. The goal of learning is to produce a hypothesis that optimizes the performance criterion of minimizing some function of the classification error (on the training data) and the complexity of the hypothesis. Under appropriate assumptions, this is likely to result in a classifier that assigns correct labels to unlabeled instances.

Unavoidably, in many application domains, data are physically distributed. We assume that the data of interest D are distributed over the data sources $D_1, \cdots, D_N$, where each data source D_i contains sub-tuples of data tuples. The distributed setting typically imposes a set of constraints Z on the learner that are absent in the centralized setting. Here we assume that due to communication complexity issues, it is not feasible to ship all the raw data but only small subsets of the distributed data sources. Thus the problem of learning classifiers from distributed data can be formulated as follows (Caragea, Silvescu, & Honavar 2004): given the data fragments $D_1, \cdots, D_N$ of a data set D distributed across N sites, a set of constrains Z, a hypothesis class H, and a performance criterion P the task of the learner L_d is to output a hypothesis $h \in H$ that optimizes P using only operations allowed by Z. As in the case of centralized learning, this is likely to result in a *classifier* that can be used to classify new unlabeled data.

We say that an algorithm L_d for learning from distributed data sets $D_1, \cdots, D_N$ is *exact* relative to its centralized counterpart L if the hypothesis produced by L_d is identical to that obtained by L from the complete data set D obtained by appropriately combining the data sets $D_1, \cdots, D_N$.

Having defined the framework of learning from distributed data, we proceed to describe our iterative algorithms for learning exact SVMs from distributed data sources.

Support Vector Machine Algorithm

Support Vector Machines (SVM) algorithm (Burges 1998) has been shown to be one of the most effective machine learning algorithms. It gives very good results in terms of accuracy when the data are linearly or non-linearly separable. When the data are linearly separable, the SVM result is a separating hyperplane, which maximizes the margin of separation between classes, measured along a line perpendicular to the hyperplane. If data are not linearly separable, the algorithm works by mapping the data to a higher dimensional *feature* space (where the data becomes separable) using an appropriate kernel function ϕ and a maximum margin separating hyperplane is found in this space. Thus the weight vector that defines the maximal margin hyperplane is a *sufficient statistic* for the SVM algorithm (it contains all the information needed for constructing the separating hyperplane). Since this weight vector can be expressed as a weighted sum of a subset of training instances, called *support vectors*, it follows that the support vectors and the associated weights also constitute sufficient statistics for learning SVM from centralized data.

Learning SVMs from Distributed Data

We assume without loss of generality that the training instances are represented (if necessary, using a suitable kernel

function) in a (*feature*) space in which data $D = \cup_{k=1}^{N} D_k$ is linearly separable.

A naive approach to learning SVMs from distributed data (Syed, Liu, & Sung 1999) works as follows: apply the SVM algorithm for each data source $D_k (k = \overline{1, N})$, and send the resulting support vectors $SV_k (k = \overline{1, N})$ to the central location. At the central location, apply SVM algorithm to $\cup_{k=1}^{N} SV_k$. Use the final set of support vectors and their corresponding weights to generate the separating hyperplane.

Although this algorithm may work reasonably well in practice if the data sets $D_1, \cdots, D_N$ are individually representative of the entire set D, if that is not the case, Caragea et al. (Caragea, Silvescu, & Honavar 2000) showed that $SV(\cup_{k=1}^{N} D_k) \neq SV(\cup_{k=1}^{N} SV_k)$, i.e. union of the set of support vectors obtained from each data source does not represent sufficient statistics for the learning from distributed data. Thus, the naive approach to learning SVM from distributed data is not *exact*.

Caragea, Silvescu, & Honavar (2000) showed that the convex hulls of the instances that belong to the two classes represent sufficient statistics for learning SVMs from distributed data. Let $VConv(D)$ denote the training instances that uniquely define the convex hull of the convex set D. The algorithm for learning SVMs from distributed data using convex hulls works as follows: compute $VConv(D_k(+))$ and $VConv(D_k(-))$ for each data source D_k, $k = \overline{1, N}$ and send these sets to the central location. At the central location the SVM algorithm is applied to the union of positive and negative convex hull vertices received from all distributed sites. It can be easily seen that: $VConv(\cup_{k=1}^{N} VConv(D_k)) = VConv(\cup_{k=1}^{N} D_k)$ (Gruber & Wills 1993), therefore this algorithm is exact. However, it is exponential in the number of dimensions, which makes it impractical in general.

Thus, we have seen two algorithms for learning SVMs from distributed data, but one of them is not *exact* and the other one is not *efficient*. We will show that it is possible to desing an *efficient* and *exact* algorithm for learning SVM-like classifiers from distributed data.

We transform SVM from centralized data into an algorithm for learning from distributed data similar to the Linear Programming Chunking Algorithm (LPC) described in (Bradley & Mangasarian 2000). More precisely, this approach is similar to the naive approach described above (Syed, Liu, & Sung 1999), except that several iterations through the distributed data sets are made. At each iteration i, the central location sends the current (global) set of support vectors GSV_i to the distributed data sources (initially, $GSV_0 = \phi$). Each data source D_k adds GSV_i to its data and applies the SVM algorithm to find a new set of local support vectors $SV_i(D_k) = SV_i(D_k \cup GSV_i)$ given the global set of support vectors GSV_i. The resulting set $SV_i(D_k)$ is sent back to the central location. The sets of support vectors received from all distributed data sources are combined and the SVM algorithm is applied to determine the new set GSV_{i+1} of global support vectors. The process is repeated until no changes in the set of global support vectors appear.

Theorem 1 (Exactness) *Our strategy yields a provably exact algorithm for learning SVM classifiers from distributed data, which terminates in a finite number of iterations.*

Proof sketch: We have: $GSV_{i+1} = SVM(GSV_i \cup SV_i(D_1 \cup GSV_i) \cdots \cup SV_i(D_K \cup GSV_i))$. We first notice that GSV_{i+1} contains only border points, otherwise there would exist misclassified instances, which contradicts the linear separability assumption. Second, we observe that GSV_{i+1} contains only the border points of a class that are "visible" from the other class; the invisible points could never be on the border between the two classes, so they could never be selected as support vectors. Thus, the final global set of support vectors represents an extended set of support vectors (subset of convex hull) which is a set of sufficient statistics learning from distributed data. Therefore, $SV(D) \subseteq GSV$ which implies $SV(D) = SV(GSV)$. This proves that our algorithm is *exact*. Note also that at each iteration i we add more points to GSV_i, therefore GSV_i is nondecreasing in size. Since the set of (extended) support vectors represents a small fraction of the whole data set and is finite, the algorithm terminates in a finite number of steps.

Preliminary experimental results have shown that our algorithm converges to the exact solution in a relatively small number of iterations.

Summary and further work

We designed an algorithm for learning SVMs from distributed data and showed theoretically and experimentally that our algorithm is exact with respect to its centralized counterpart and also efficient. This makes it preferable to previous algorithms as those described in (Caragea, Silvescu, & Honavar 2000) (inefficient) and (Syed, Liu, & Sung 1999) (inexact). We assumed that data are horizontally fragmented. Future work directions include design and analysis of algorithms for learning SVMs from other kinds of data fragmentation, e.g.,relational data fragmentation.

References

Bradley, P. S., and Mangasarian, O. L. 2000. Massive data discrimination via linear support vector machines. *Optimization Methods and Software* 13(1):1–10.

Burges, C. 1998. A tutorial on support vector machines for pattern recognition. *Journal of Data Mining and Knowledge Discovery*.

Caragea, D.; Silvescu, A.; and Honavar, V. 2000. Agents that learn from distributed dynamic data sources. In *Proceedings of the Workshop on Learning Agents*.

Caragea, D.; Silvescu, A.; and Honavar, V. 2004. A framework for learning from distributed data using sufficient statistics and its application to learning decision trees. *International Journal of Hybrid Intelligent Systems* 1(2).

Gruber, P., and Wills, J. 1993. *Handbook of Convex Geometry*. Elsevier Science Publishers B.V.

Mitchell, T. 1997. *Machine Learning*. McGraw Hill.

Syed, N.; Liu, H.; and Sung, K. 1999. Incremental learning with support vector machines. In *Proceedings of the KDD Conference*.