Closest Pairs Data Selection for Support Vector Machines

Chaofan Sun
Department of Computer Science, University of Houston
4800 Calhoun Rd., Houston TX 77204-3010
cfsun@cs.uh.edu

Abstract

This paper presents data selection procedures for support vector machines (SVM). The purpose of data selection is to reduce the dataset by eliminating as many non support vectors (non-SVs) as possible. Based on the fact that support vectors (SVs) are those vectors close to the decision boundary, data selection keeps only the closest pair vectors of opposite classes. The selected dataset will replace the full dataset as the training component for any standard SVM algorithm.

Introduction

As a state-of-the-art learning approach, SVM based on maximal margin (Vapnik 1998) has demonstrated to be advantageous over other learning approaches. Conventionally, L2-SVMs training needs to solve a quadratic programming (QP) problem which has $O(n^3)$ time and $O(n^2)$ space complexities, where n is the size of dataset. For big datasets, conventional SVM training becomes intolerably slow. Also, the amount of memory needed for storing the data is often insufficient. To overcome the time and space problems, some studies focus on data selection as a form of data preprocessing. These studies are based on the fact that support vectors (SVs) are only a subset of the full dataset. Training a selected data subset containing all SVs will generate the same model as that from training the full dataset. As a preprocessing procedure, data selection will boost the speed of any SVM training algorithm. It is also possible to use more complicated kernel functions to improve the performance as the dataset is reduced.

Data selection can be conducted based on different approaches. The simplest one is random selection. Since there is no preference during random selection, it is very likely that some SVs will not be selected. Another approach uses cluster purity(Koggalage & Halgamuge 2004); clusters with different class vectors are more likely to be close to the decision boundary. Pure clusters with same class vectors are assumed to be far from the decision boundary. Selection is then made based on cluster purity. Similarly, data selection can be done based on k nearest neighbor (k-NN) (Wang & Cooper 2005; Shin & Cho 2003). It can be shown that both clustering and k-NN identify SV candidates using an enclosing ball (with a constant L2-distance radius). Some data may not be reachable by any such ball; these data are assumed far from the decision boundary and are normally kept in a selected dataset. These algorithms tend to select a certain number of non-SVs. Another problem with these algorithms is the time complexity because clustering or k-NN computing is very time consuming without any index.

SVs are interpreted as the closest vectors of opposite class convex hulls (Bennett & Bredensteiner 2000). Based on this relation, closest pairs of opposite classes can be used in data selection. To find these data pairs, one approach investigates data selection based on all Gabriel neighbors (Zhang & King 2002). It first computes all Gabriel neighbors and only Gabriel neighbors of different classes are kept as SV candidates. Its time complexity is the order of $O(n^3)$. This approach can give all exact closest pairs. However, it also computes many Gabriel neighbors that eventually will not be selected. Other examples related to closest pairs are DirectSVM (Roobaert 2000) and CriticalSVM (Raicharoen & Lursinsap 2002). They use a brute force approach to find the best closest pair as SVs, then construct the initial hyperplane which in turn is used to find other possible SVs to self-update. Using a brute force approach to find the best closest pair requires $O(n^2)$ operations. To avoid this high cost, SimpleSVM (Vishwanathan & Murty 2002) uses a random subset data to approximate the exact best pair. These algorithms not only do data selection but also hyperplane update. However they need to scan the full dataset several times before converging to the optimal hyperplane.

In principle, time complexity for finding all exact closest pairs crossing the decision boundary is the order $\Omega(n^2)$ without any index, because all pair distances have to be computed and compared. This is too costly for big datasets. Instead of finding all exact closest pairs, we propose an approximation. In the next section, we will discuss our data selection approach. Finally, we conclude this study and outline our future research.

Data Selection Based on Closest Pairs

In order to evaluate the efficiency of the proposed approximation, we first investigate the closest pairs approach using pair distance matrix. The procedure works as follows: (a) divide the full dataset into positive and negative subsets, with sizes n_1 and n_2 respectively, (b) compute the distance matrix

D with $n_1 \times n_2$ elements. Element D_{ij} is the distance of positive vector $\mathbf{x_i^+}$ to negative vector $\mathbf{x_j^-}$, (c) for i = 1, 2, ..., n_1, find the minimum distance D_{ik} in i^{th} row and mark vector $\mathbf{x_k^-}$, (d) for each k, find the minimum distance D_{lk} in k^{th} column and mark vector $\mathbf{x_l^+}$, (e) ouput the marked elements. This algorithm is simple and easy to implement. Its time complexity is of $O(n^2)$. Selected datasets and full dataset are trained and tested using LIBSVM (Chang & Lin 2001) under the same conditions. Investigations on several 2-class UCI datasets show that closest pairs can cover around 90% of SVs. If we select slightly more data by adding more closest pairs in step (d), say 110% -120% of the size of SVs, we can cover almost all SVs. After this point, selecting more data is not helpful to improve performance. For example, for breast cancer dataset, there are a total of 683 data in which 69 (around 10%) are SVs. If we select 79 (around 110% of SVs) and train with this subset, we can get almost the same model as training with the full dataset. In practice, without prior knowledge on how many SVs exist in a dataset, we have to check the models generated from training different subsets until no more improvement on performance is observed.

For big datasets, using the matrix distance will not work. We need a kind of index structure to speed up the closest-pair search. In instance based learning, k-NN can be approximated using SASH, Spatial Approximation Sample Hierarchy (Houle 2003). The structure is a pyramid-type tree with height of $log_2 n$. The connections are established based on the pc-NN, where p and c are the numbers of parents and children of a given node. The basic idea behind SASH tree is an approximation of the triangle inequality $d(x, z) \leq d(x, y)+d(y, z)$, where d represents the distance of two vectors. In other words, if y is the closest neighbor of x, and z is the closest neighbor of y, then z is most likely one closest neighbor of x. Inspired by this structure, we use also similar structure to approximate closest pairs but from opposite classes, different from pc-NN without class distinction in SASH. Therefore node connections between two adjacent levels are established based on the top closest pairs of opposite classes. For each node, there is also one or more connections to its same class closest pairs in order to keep indirect closest pairs. The time complexity of building this tree is $O(pcn \log_2 n)$ and querying is $O(pc \log_2 n)$. Accordingly, searching closest pairs of opposite classes is one order of magnitude faster and 90% more SVs can be covered.

The two approaches given above are dependent on the input space. The selected data are physically close to each other. In kernel space, mapped data can be separated by a hyperplane. Thus closest pair selection in kernel space becomes convenient. Investigations show that slightly fewer data will be selected compared to the selection in input space in order to reach a comparable performance. However the price paid is the additional step of computing kernel evaluations for each distance. This is very costly with big datasets.

Final Remarks and Future Research

Data selection based on closest pairs of opposite classes selects fewer data compared to that based on random selection, clustering or kNN. For a given dataset, the size of selected data depends on how many SVs exist in the full dataset. Generally, the number of selected data needs to be slightly above all SVs to obtain reasonable performance. For big datasets, finding all closest pairs is very time consuming. Using SASH tree structure as an approximation can reduce the searching time one order of magnitude with more than 90% SVs covered.

The SASH tree is a static structure. Its size is equal to the size of the dataset. For over-sized datasets, we also suffer from memory shortage. By growing the SASH tree to certain level, we know that some vectors are definitely not SVs. The next step in our research is to make SASH tree dynamic by replacing vectors with those closer to the decision boundary. As to SVM training, we will use hyperplane updating based on convex hull points to avoid solving a QP problem (while avoiding scanning the data multiple times).

References

Bennett, K. P., and Bredensteiner, E. J. 2000. Duality and Geometry in SVM Classifiers. In *Proc. 17th International Conf. on Machine Learning*, 57–64. Morgan Kaufmann, San Francisco, CA.

Chang, C.-C., and Lin, C.-J. 2001. LIBSVM: A Library for Support Vector Machines.

Houle, M. E. 2003. Sash: A spatial approximation sample hierarchy for similarity search. Technical Report 16 pages, IBM Tokyo Research Laboratory Report RT-0517.

Koggalage, R., and Halgamuge, S. 2004. Reducing the Number of Training Samples for Fast Support Vector Machine Classification. In *Neural Information Processing - Letters and Reviews*, volume 2, 57–65.

Raicharoen, T., and Lursinsap, C. 2002. Critical Support Vector Machine Without Kernel Function. In *Proc. of 9th International Conference on Neural Information*, volume 5, 2532–2536.

Roobaert, D. 2000. DirectSVM: A Fast And Simple Support Vector Machine Perceptron. In *Proceedings. IEEE Int. Workshop Neural Networks for Signal Processing*, 356–365.

Shin, H., and Cho, S. 2003. Fast Pattern Selection for Support Vector Classifiers. In *Proceedings of the 7th Pacific-Asia Conference on Knowledge Discovery and Data Mining. Lecture Notes in Artificial Intelligence (LNAI 2637)*, 376–387.

Vapnik, V. N. 1998. *Statistical Learning Theory*. Addison-Wiley, New York, NY.

Vishwanathan, S., and Murty, N. M. 2002. A Simple SVM Algorithm. In *Proceedings 2002 International Joint Conference on Neural Networks. IJCNN '02*, volume 3, 2393–2398.

Wang, J., N. P., and Cooper, L. N. 2005. Training Data Selection for Support Vector Machines. In *Lecture Notes in Computer Science (LNCS)*, volume 3610, 554 – 564.

Zhang, W., and King, I. 2002. A Study of the Relationship Between Support Vector Machine and Gabriel Graph. In *IEEE*, 239–245.