SvmFu: Software For SVMs

Ryan Rifkin & Michelle Nadermann

Artificial Intelligence Laboratory
Massachusetts Institue Of Technology
Cambridge, Massachusetts 02139

http://www.ai.mit.edu

The Problem: The Support Vector Machine (SVM) is a popular and powerful technique for binary classification [1]. Training an SVM requires the solution of the following quadratic program:

$$\begin{aligned} \text{maximize} \quad & \textstyle\sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j Q_{ij} && (1) \\ \text{subject to :} \quad & \textstyle\sum_i y_i \alpha_i = 0 && (2) \\ & 0 \leq \alpha_i \leq C,\ \forall i && (3) \end{aligned}$$

$Q_{ij} \equiv y_i y_j K(x_i, x_j)$, where K is a user-supplied positive-semidefinite kernel function, C is a user-supplied regularization constant.

Motivation: Because the Q matrix in the SVM training problem is in general dense and of full rank, and has size equal to the number of data points squared, traditional "off the shelf" QP methods are not applicable for solving large problems (above 2,000 or so points). Nevertheless, there is a need to solve large instances: on many problems, SVMs have provided the best performance of any methods to date.

Previous Work: Many attempts have been made to solve large SVM problems. Osuna et. al. [2] provided the first practical algorithm for training large SVMs. They essentially showed that one can train a large SVM by solving a sequence of smaller SVM training problems, and use an off-the-shelf QP code to solve the resulting small problems. John Platt [3] made the additional observations that the subproblems could be chosen to be of size 2, and that these minimal subproblems could easily be solved analytically, avoiding the need for a costly or difficult-to-code QP solver.

Approach: We have developed a new approach that combines the strengths of previous methods. Although Platt's method can solve the individual subproblems very quickly, it often has to look at all the data points, which is problematic. We decompose our problems into a sequence of (large, but small enough so that the associated Q matrix fits in memory) subproblems, and then solve the subproblems using a substantially modified version of Platt's algorithm. Space limitations prohibit discussing the details of the algorithm, which is presented in [4].

In addition to the theoretical advances, SvmFu includes a number of important engineering advances compared to other contemporary "fast" codes ([5], [6]). SvmFu was implemented as a templated C++ library. Therefore, it can easily be adjusted to handle data of arbitrary types (floats, doubles, intgers, etc.), providing the required tradeoff between speed, memory and precision. Additionally, SvmFu allows the user to specify their own representations of the data, allowing it to handle both dense and sparse (high-dimensional with very few non-zero entries) formatted data. Finally, SvmFu is released under the GPL, and is therefore easily adaptable by other researchers and practictioners.

Impact: SvmFu works as desired. It is fast and easy-to-use. We present here a single example — more extensive testing results can be found in [4].

We considered a training set consisting of a database of 31,022 faces. We used successive nested subsets consisting of .2, .4, .6, .8, and the entire dataset. We tested SvmFu against two other currently popular SVM training programs, SvmLight and SvmTorch. These are both "second generation" systems, designed with speed in mind. We set the training parameters so that the algorithms would behave as similarly as possible: we used an identical kernel, an identical value of C, and made sure that all programs had (approximately) the same amount of memory available, and tried to use similar settings as regards shrinking.[1] The table shows that in all cases, SvmFu trained substantially faster than SvmLight or SvmTorch — essentially the same solution was obtained by the three systems.

SvmFu is a fast, flexible, and freely available implementation for SVM training. It is already in use at many universities and companies.

Future Work: We have implemented an SVM training algorithm that represents both a theoretical and engineering advance over competing codes. In the near future, we plan to include additional value-added software for performing multiclass classification and converting SVM outputs to probabilities. We will also continue to improve the SVM algorithm, and look at various applications in bioinformatics, object recognition, and text categorization.

Dataset Size elements in	SvmFu some shrinking	Svmlight floats, no	SvmTorch doubles,
6204	87.45	182.31	97.12
12409	188.79	381.19	245.72
18613	294.96	453.96	523.78
24818	536.11	717.74	1035.32
31022	1080.57	1232.49	2865.86

Research Support: Research at CBCL is supported by ONR, Darpa, NSF, Kodak, Siemens, Daimler, ATR, ATT, Compaq, Honda, CRIEPI.

References:

[1] V. Vapnik. Statistical Learning Theory. John Wiley & Sons, 1998.

[2] E. Osuna, R. Freund and F. Girosi. An Improved Training Algorithm for Support Vector Machines. Neural Networks for Signal Processing VII, 1997.

[3] J. Platt. Sequential Minimal Optimization: A Fast Algorithm For Training Support Vector Machines. Microsoft Research, MST-TR-98-14, 1998.

[4] R. Rifkin, M. Nadermann and P. Moreno. SvmFu: A Fast, Flexible Support Vector Machine Classification Algorithm. (in preparation).

[5] T. Joachims. Making Large-Scale Support Vector Machine Learning Practical. In "Advances in Kernel Methods – Support Vector Learning", 1998.

[6] R. Collobert and S. Bengio. Support Vector Machines for Large-Scale Regression Problems. Institut Dalle Molle d'Intelligence Artificielle Perceptive, IDIAP-RR-00-17, 2000.

[1] Shrinking is essentially the idea of noting that points that are far away from the separating hyperplane over a long period of time are likely never to become support vectors, and may in all likelihood be removed from the dataset. It is discussed in much greater detail in [5] and [4].