

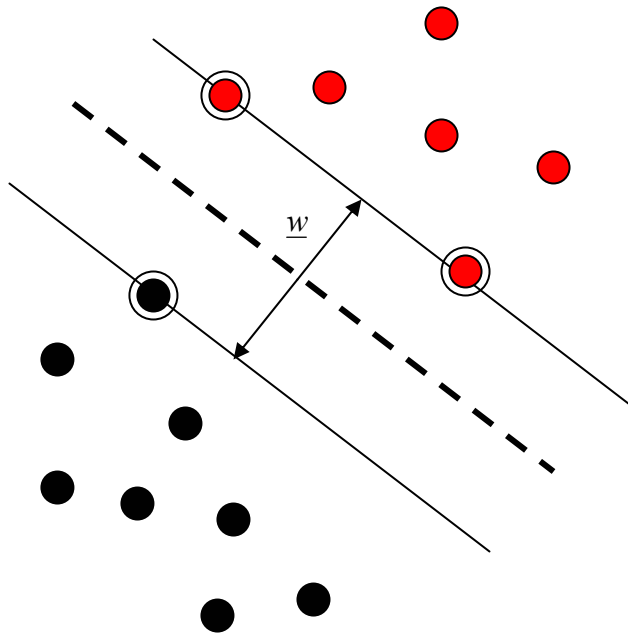
Data Parallelism and the Support Vector Machine

Solomon Gibbs

- The support vector machine is a common algorithm for pattern classification. However, many of the most popular implementations are not suitable for parallelization in a distributed-data sense. We discuss current serial techniques and several alternative methods already in literature as well as their potential suitability to the Matlab-MPI system.
- The support vector machine has gained popularity because it produces a classifier with maximum generality, can be adapted to produce non-linear classifiers, and has mitigating factors for the curse of dimensionality.
- However, most SVM algorithms are serial in nature. We believe that a parallel implementation could be used to handle very large datasets efficiently.

Linear Support Vector Machine

- Canonical problem: The maximal margin classifier.



Maximize $\|w\|^2$ subject to the constraint that all points are correctly classified. Notice that the circled points are sufficient to define the hyperplane. These are the support vectors.

Decomposition Techniques

- The above problem leads to the quadratic program:

$$\begin{aligned} & \text{minimize } w(\underline{\Lambda}) = -\underline{\Lambda}^T \underline{1} + 1/2 \underline{\Lambda}^T D \underline{\Lambda} \\ & \text{subject to } \underline{\Lambda}^T \underline{y} = 0, \\ & \quad 0 \leq \underline{\Lambda} \leq C \underline{1} \end{aligned}$$

where $\underline{\Lambda}$ is the vector of multipliers,
 D is defined as $[D_{ij}] = y_i y_j K(\underline{x}_i, \underline{x}_j)$,
and C is a positive constant.

It is easy to see that the problem quickly becomes intractable for large datasets. (D grows with the square of the number of training points)

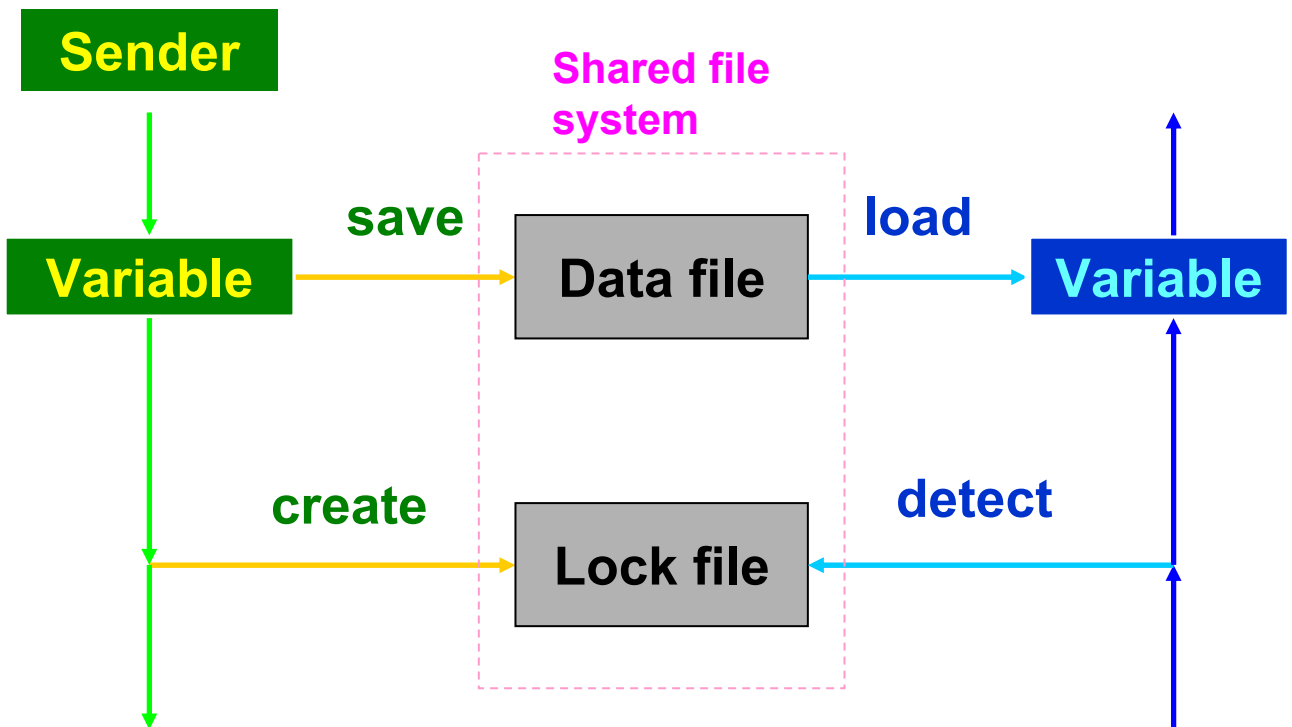
Various authors [1, 2] have proposed decomposition techniques that solve a series of smaller programming problems and converge to a global solution.

$$\begin{aligned} \underline{\Lambda} &= \begin{bmatrix} \underline{\Lambda}_B \\ \underline{\Lambda}_N \end{bmatrix} \quad \underline{y} = \begin{bmatrix} \underline{y}_B \\ \underline{y}_N \end{bmatrix} \quad D = \begin{bmatrix} D_{BB} & D_{BN} \\ D_{NB} & D_{NN} \end{bmatrix} \\ & \text{minimize } \underline{w}(\underline{\Lambda}) = -\underline{\Lambda}_B^T (\underline{1} - D_{BN} \underline{\Lambda}_N) + \frac{1}{2} \underline{\Lambda}_B^T D_{BB} \underline{\Lambda}_B \\ & \quad + \frac{1}{2} \underline{\Lambda}_N^T D_{NN} \underline{\Lambda}_N - \underline{\Lambda}_N^T \underline{1} \\ & \text{subject to } \underline{\Lambda}_B^T \underline{y}_B + \underline{\Lambda}_N^T \underline{y}_N = 0, \\ & \quad 0 \leq \underline{\Lambda} \leq C \underline{1} \end{aligned}$$

And the last two terms of $\underline{w}(\underline{\Lambda})$ are held constant, and can be neglected.

Matlab-MPI

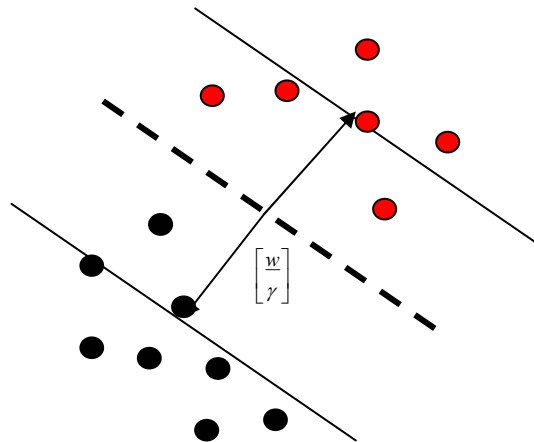
- A message-passing protocol in Matlab allows workstations to pass data and tasks among one another



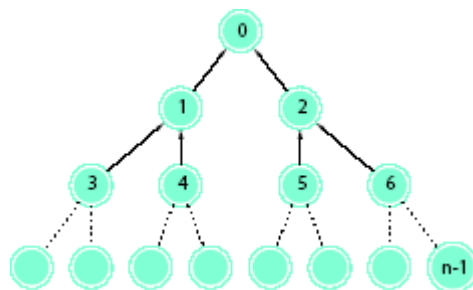
- Receiver waits until it detects the existence of the lock file.
- Receiver deletes the data and lock file, after it loads the variable from the data file.

Incremental Proximal SVM

- An alternate formulation[4] of the maximal margin problem allows an incremental update algorithm:



By altering the constraints in the optimization problem, a “proximal” classifier that can be solved by a system of linear equations is formed. This algorithm can be further modified to add or remove data from the training set, while updating the classifier.



Tveit has proposed a cascade of updating nodes, each computing an update increment in parallel.[5]

Ensemble Methods

- The above algorithms focused on a single 2-way classifier. Using ensemble methods, there are a variety of possible ways to combine SVM sub-units.
 - M-Way classifier
Train each unit on the entire dataset, but with different classes to separate.
 - Mixture of Experts[6]
Train each unit on a random subset of the data, then use weighted combination to classify.
 - Consensus methods (voting, least squared error, maximum likelihood)
Train units using different criteria, then combine results using above methods.

Conclusions and Future Work

- Serial support vector machine algorithms have been developed that are capable of handling very large datasets.
- Many of the most popular methods are unsuitable for use in a data-parallel environment.
- A number of algorithms for making use of support vector concepts in a data-parallelizing have also been proposed.
- A data-parallel classifier based on the SVM is desirable because of the accuracy and flexibility of the SVM and the speed up provided by the parallel environment, without the need for specialized parallel hardware.
- More work is needed to determine what methods will produce a computationally efficient and theoretically sound classifier for use in a high-latency message passing environment.

References

1. C. Burges, “A Tutorial on Support Vector Machines for Pattern Recognition”, *Data Mining and Knowledge Discovery* 2:121-167, 1998
2. E. Osuna and R. Freund and F. Girosi, “Improved Training Algorithm for Support Vector Machines”, *NNSP* 1997
3. J. Platt, “Sequential Minimal Optimization: A fast algorithm for training support vector machines”, 1998
4. G. Fung and O. Mangasarian, “Incremental Support Vector Machine Classification”, *SIAM* 2001
5. A. Tveit and H. Engum, “Parallelization of the Incremental Proximal Support Vector Machine Classifier Using a Heap-based Tree Topology”, 2003
6. R. Collobert and S. Bengio and Y. Bengio, “A Parallel Mixture of SVMs for Very large Scale Problems”, *Advances in Neural Information Processing Systems*, 2002