ECE5801: Support Vector Machine, Project 5

In this project, you will be writing codes and perform two experiments with the Support Vector Machines (SVMs) that you have learned in class. As it was discussed in class, RBF networks can be effectively trained as SVM. This approach is effective and shows excellent performance.

Although SVMs have proven to be highly effective in learning many practical problems, its original implementation (Vapnik) requires use of a quadratic programming tool (commercial library packages), which is inconvenient for this class and for general use. To avoid this inconvenience, in this project you will use the Adatron algorithm that was provided in the class. Upon successful implementation, you will notice that this algorithm works efficiently.

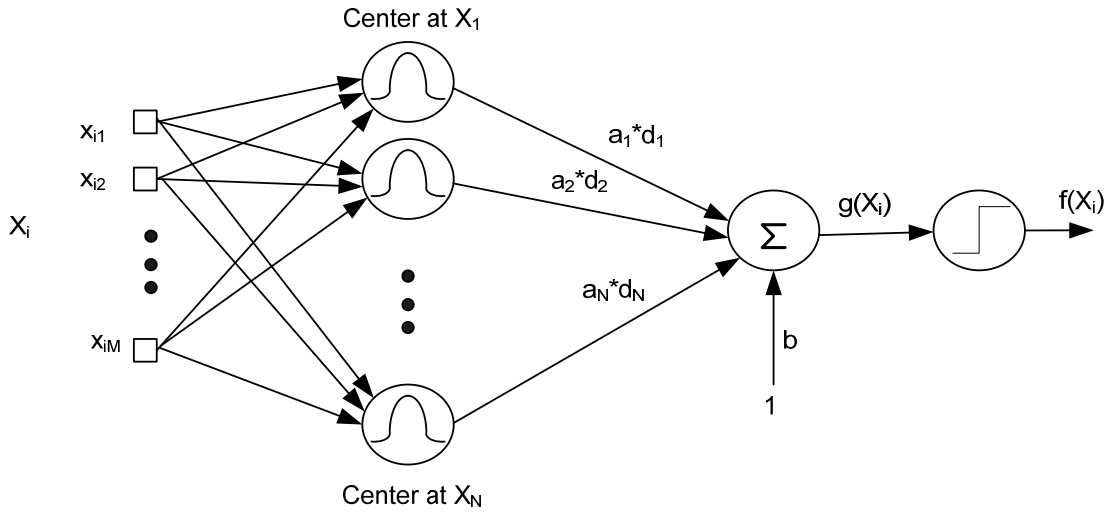


Figure 1: Adatron SVM network

The Adatron learning algorithm is described below.

The training data set is assumed given as

$$\{(X_1, d_1), (X_2, d_2), ..., (X_N, d_N)\}$$

The Gaussian functions used as the radial basis functions for this project are defined as

$$G(X - X_i, 2\sigma^2) = \exp(-\| X - X_i \|^2 / 2\sigma^2)$$

From the diagram, a_i are the output layer weights and d_i are the desired values which are multiplied to compute $g(X_i)$, i.e.,

$$g(X_i) = \sum_{j=1}^{N} d_j a_j G(X_i - X_j, 2\sigma^2) + b$$

Let

$$z(X_i) = d_i g(X_i) = d_i [\sum_{j=1}^{N} d_j a_j G(X_i - X_j, 2\sigma^2) + b] \quad (.1)$$

For the stopping criterion, a minimum among $z(X_i)$ is selected, i.e.,

$$\theta = \underset{i}{Min}\, z(X_i)$$

and the weights are updated while θ is greater than a preset threshold or until the maximum number of iterations are reached. The complete algorithm is summarized below.

1: Initialize a_i for all i with small positive random numbers. Also, set the threshold for θ to a small number (say, 0.01)

2. Update

$$\text{If } a_i(n) + \Delta_i a_i > 0,$$
$$a_i(n+1) = a_i(n) + \Delta_i a_i, \qquad b(n+1) = b(n) + d_i \Delta_i a_i$$
$$\text{else}$$
$$a_i(n+1) = a_i(n), \qquad b(n+1) = b(n)$$

where

$$\Delta a_i = \eta(1 - z(X_i))$$

and η is the learning rate.

3. Repeat the step 2 while θ >0.01 or until the update reaches the maximum iteration.

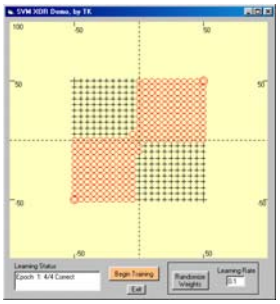
The testing of which class the network classified is given by,

$$f(X_i) = \text{sgn}(g(X_i))$$

In Figure 1, the final stage is this relation, which is simply a threshold operation. For this project, this should be used to monitor what percentage of the patterns are correctly classified.

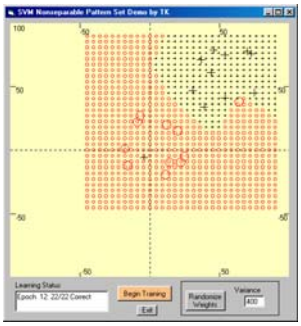
Simulation Tasks

1. Using the SVM algorithm described, train the RBF network for the XOR problem (all data set values should be {1,-1}), and draw the decision boundary. The output should be similar to the following figure:



I recommend that you try to generate a little bit more detailed decision boundary than the example given above. The most important part is how the learning rate and the variance, σ^2, of each basis function affect the out come. Try different learning rates or variances, and report your observation and reasoning.

2. Implement the 22 pattern data-set given for the Perceptron and Widrow-Hoff project using the SVM algorithm given above. Draw the decision boundary on completion of the training. The result should look similar to the below figure:



Again, observe how the variance affects the decision boundary and report your observation and reasoning.

Report:

1. Describe the basic principles of SVM.
2. Describe how the learning rate and variances affect the decision boundary and provide your reasoning using the experimental results
3. Can you find the support vectors using the above algorithm? If yes, describe the support vectors you found.
4. How would you modify the SVM, if you wish to apply this algorithm to SNP500 data?
5. Attach the source listing