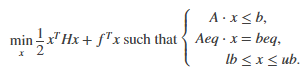
James Hahn

CS1675 Machine Learning

Dr. Adriana Kovashka

**Algorithm Presentation**

In MATLAB, the *quadprog(H, f, A, b, Aeq, beq, lb, ub)* provides a user friendly interface to solve quadratic optimization problems in the form:



Given the nature of SVM optimization, we want to flip the script with the following optimization problem:

To convert this to a minimization problem, we convert it to its dual:

Subject to:

Thus, for the *quadprog* function, the parameters are as follows:

H NxN where H[i, j] = yi \* yj \* xi \* xjT

F Nx1 = [-1, -1, …, -1]T

Aeq Nx1 = yT

beq 1x1 = 0

lb Nx1 (lower bound) = 0

ub Nx1 (upper bound) = [C, C, …, C]T

*A* and *b* are not utilized since the first constraint does not apply to our problem

Once plugged into *quadprog*, the alpha array α is returned. From the notes, we can compute the weight vector and bias for our SVM’s **wTx + b** with the following formulas:

With both the weights and bias, given a d-dimensional test feature vector x, we can compute the final class prediction with the following formula in MATLAB:

Thus, our SVM successfully solves a quadratic optimization problem to produce a hyperplane of maximum margin across varying values of C, which is a parameter to penalize misclassified samples.

**Results**

Below are the results of the SVM’s accuracy with C = {10-4, 10-3, 10-2, 10-1, 100}. Accuracies are plotted for each of the values, as they each correspond to a bar on the graph. There is a significant boost in accuracy from C = 10-4 to C = 10-3; about 10%. However, higher values of C do not showcase a significant impact on the accuracy of the SVM. The C parameter in our SVM’s optimization limits the range of the alpha α parameters (i.e. 0 ≤ αi ≤ C). It can also be viewed as a misclassification cost parameter. As such, as C increases, misclassified samples will be penalized further in the quadratic optimization problem, resulting in a hyperplane that is very strictly between the two classes. It is surprising the accuracy does not increase further by a couple percent as C approaches 1, but this may be because the data is already fairly linearly separable after C = 10-4, so increasing C does not produce much of an effect.

Given the results, a C value of 10-3 or 10-2 would be best because a higher C should result in a closer overfitting to the training dataset, since it specifically focuses on minimizing those misclassified samples, but does not produce a significant increase in accuracy in the long term. Therefore, a lower value of C, which obtains a satisfactory accuracy, should generalize better to test data.