**CSCI567 Fall 2015**

**Homework #4**

**Vaibhav Behl**

**3123054410**

**vbehl@usc.edu**

## 1. Boosting

For this binary classification task we are given $y_i \in \{+1, -1\}$ for $x_i \in R^d$, for $i = 1, \dots, n$. We are also given a set of weak learners $H = \{h_j, j = 1, \dots, M\}$. The logistic loss function is given as:

$$L(y_i, \hat{y_i}) = \log(1 + \exp(-y_i \hat{y_i}))$$

a) Gradient Calculation

$$g_i = \frac{\delta L(y_i, \hat{y_i})}{\delta \hat{y_i}}$$
$$= -\frac{y_i e^{-y_i \hat{y_i}}}{1 + e_i^{-y_i \hat{y_i}}}$$

b) Weak learner selection

$$h^* = \arg\min_{h \in H} (\min_{\gamma \in R} \sum_{i=1}^{n} \left(-g_i - \gamma h(x_i)\right)^2)$$

Here for the internal function where we want to minimize for $\gamma$, we can calculate a closed form value for it, thus $h^*$ can be derived independent of $\gamma$.

First, taking the function to be minimized:

$$f(\gamma)_i = \sum \left(-g_i - \gamma h(x_i)\right)^2$$

Now taking derivative,

$$\frac{\delta f(\gamma)}{\delta \gamma} = \sum 2\left(-g_i - \gamma h(x_i)\right).-h(x_i) = 0$$

Thus $\gamma$ can be computed as follows:

$$\gamma = -\frac{2\sum g_i h(x_i)}{\sum h(x_i)^2}$$

c) Step Size Selection

We need to calculate the optimal $\alpha^*$that minimizes the loss:

$$\alpha^* = \arg\min_{\alpha \in R} \sum_{i=1}^{n} L(y_i, \hat{y_i} + \alpha h^*(x_i))$$

Now using the equation for $L$ function,

$$L\left(y_i, \hat{y_i} + \alpha h^*(x_i)\right) = \log\left(1 + e^{-y_i\left(\hat{y_i} + \alpha h^*(x_i)\right)}\right)$$

So, taking the function to be minimized as $g$ :

$$g(\alpha) = \sum_{i=1}^{n} \log\left(1 + e^{-y_i\left(\hat{y_i} + \alpha h^*(x_i)\right)}\right)$$

Now utilizing Newton's step, and calculating gradient:

$$\nabla g(\alpha) = \sum \frac{-y_i h^*(x_i) e^{-y_i(\widehat{y_i} + \alpha h^*(x_i))}}{1 + e^{-y_i(\widehat{y_i} + \alpha h^*(x_i))}}$$

Now its value at $\alpha = 0$ ,

$$\nabla g(0) = \sum \frac{-y_i h^*(x_i) e^{-y_i \widehat{y_i}}}{1 + e^{-y_i \widehat{y_i}}}$$

Now for second order,

$$\nabla^2 g(\alpha) = \frac{\sum\left(1+e^{-y_i(\widehat{y_i}+\alpha h^*(x_i))}\right)\left(y_i^2 h^*(x_i)^2 e^{-y_i(\widehat{y_i}+\alpha h^*(x_i))}\right) - \left(y_i^2 h^*(x_i)^2 e^{-y_i(\widehat{y_i}+\alpha h^*(x_i))}\right)\left(e^{-y_i(\widehat{y_i}+\alpha h^*(x_i))}\right)}{\left(1+e^{-y_i(\widehat{y_i}+\alpha h^*(x_i))}\right)^2}$$

Now second derivative at zero,

$$\nabla^2 g(0) = \sum \frac{\left(1 + e^{-y_i \widehat{y_i}}\right)\left(y_i^2 h^*(x_i)^2 e^{-y_i \widehat{y_i}}\right) - \left(y_i^2 h^*(x_i)^2 e^{-2y_i\ \widehat{y_i}}\right)}{(1 + e^{-y_i \widehat{y_i}})^2}$$

Now using the Newton's step:

$$\alpha^* = \alpha_0 - \frac{\nabla g(0)}{\nabla^2 g(0)}$$

Further simplifying,

$$\alpha^* = -\frac{\sum \frac{-y_i h^*(x_i) e^{-y_i \widehat{y_i}}}{1 + e^{-y_i \widehat{y_i}}}}{\sum \frac{(1 + e^{-y_i \widehat{y_i}})(y_i^2 h^*(x_i)^2 e^{-y_i \widehat{y_i}}) - (y_i^2 h^*(x_i)^2 e^{-2y_i\ \widehat{y_i}})}{(1 + e^{-y_i \widehat{y_i}})^2}}$$

$$\alpha^* = \sum -\frac{(1 + e^{-y_i \widehat{y_i}})}{y_i h^*(x_i)}$$

## 2. SVM

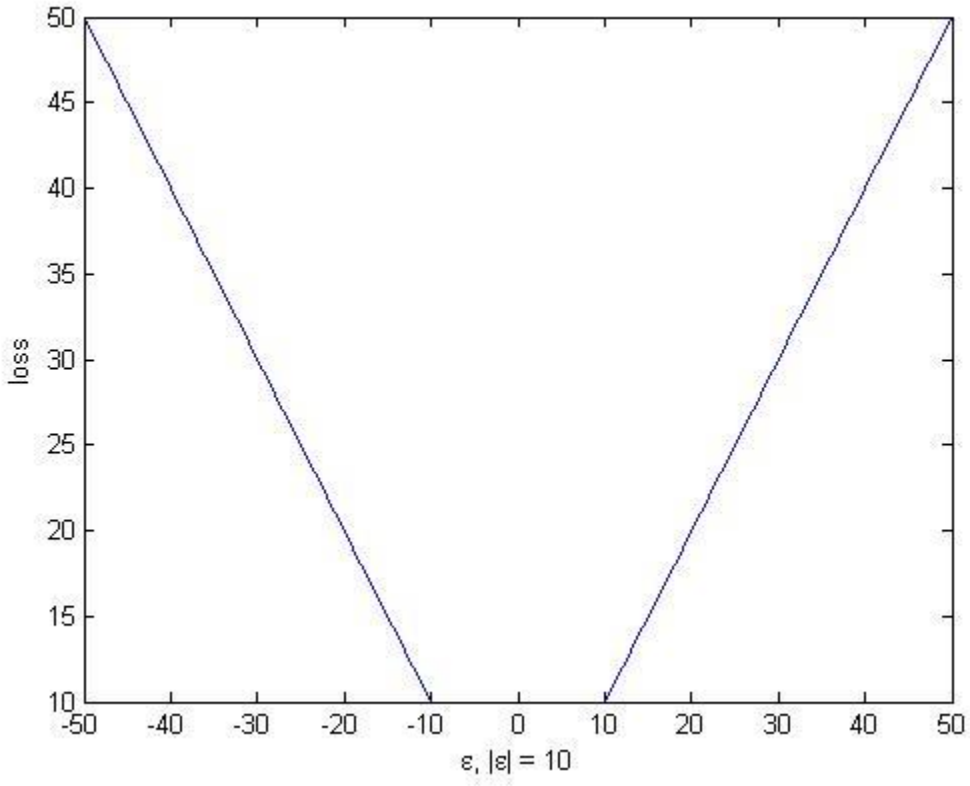
Primal Optimization problem:

$$\min_{w} \left|\left|w\right|\right|^2 ; \quad s.t.\ y_i - w_i x_i - b \leq |\epsilon|$$

Modifying optimization to incorporate the loss function with weight factor C:

$$\min_{w} \left|\left|w\right|\right|^2 + C \sum_{i=1}^{l} \epsilon_i \ ; \quad s.t.\ y_i - w_i x_i - b \leq |\epsilon| \ \ [and] \ \epsilon \geq 0$$

Plot of hinge loss function (using matlab):




Deriving the Dual form:

$$L = \frac{1}{2}\left|\left|w\right|\right|^2 + C\sum_{i=1}^{l} \epsilon - \sum_{i=1}^{l} \alpha_i(\epsilon - y_i + w_i x_i + b) - \sum_{i=1}^{l} \alpha_i'(\epsilon + y_i - w_i x_i - b)$$

Where the two values of $\alpha$ are for two constraints. Both of these are our Lagrangian multipliers, with additional constraint- $\alpha_i\, \alpha_i' \geq 0$.

Now taking partial derivatives:

$$\frac{\delta L}{\delta b} = \sum (a_i' - a_i) \ = 0$$

$$\frac{\delta L}{\delta w} = w - \sum(\alpha_i - \ \alpha_i')x_i = 0$$

$$\frac{\delta L}{\delta \epsilon} = \ C - \sum \alpha^{(\prime)} = 0$$

Substituting these values back into the dual form we get the dual optimization problem as:

$$\min -\frac{1}{2}\sum_{i,j}[(\alpha_i - \ \alpha_i')x_i].\left[\left(\alpha_j - \alpha_j'\right)x_j\right] \ + \sum y_i(a_i - a_i')$$

Subject to constraints:

$$\sum(\alpha_i - \ \alpha_i^*) = 0 \quad and \quad 0 \le (\alpha_i.\alpha_i^*) \le C$$

Also the substituted value of $w$ is:

$$w = \sum(\alpha_i - \alpha_i')x_i$$

This shows that $w$ is a linear in alpha, which are support vectors.

SV Regression (non-linear)

Modifying dual-form to have non-linear mapping:

$$\min -\frac{1}{2}\sum_{i,j}(\alpha_i - \alpha_i').\left(\alpha_j - \alpha_j'\right).\phi^T(x_i)\phi^T\left(x_j'\right) + \sum y_i(\alpha_i - \alpha_i')$$

Where $\phi^T(x)\phi^T(x') = k(x, x')$. Also, the constraints remain the same.

Now, the weight vector $w$ can be written as:

$$w = \sum(\alpha_i - \alpha_i')\phi(x_i)$$

And the prediction will be:

$$f(x) = b + \ \sum(\alpha_i - \alpha_i').k(x, x')$$

## 3. Programming

3.1 Data preprocessing: Data Processes according to the instructions given. See main Matlab function and transform_features.m.

3.2 Implement linear SVM: See Matlab files own_linear.m, trainsvm.m, testsvm.m.

3.3 Cross validation for linear SVM:

a) Cross validation result:

For C = 0.000244, Average Accuracy = 0.593961 and Average Train Time = 0.493444
For C = 0.000977, Average Accuracy = 0.905985 and Average Train Time = 0.569418
For C = 0.003906, Average Accuracy = 0.922988 and Average Train Time = 0.678002
For C = 0.015625, Average Accuracy = 0.937491 and Average Train Time = 0.724821
For C = 0.062500, Average Accuracy = 0.940994 and Average Train Time = 0.676677
For C = 0.250000, Average Accuracy = 0.940499 and Average Train Time = 0.841706
For C = 1.000000, Average Accuracy = 0.937494 and Average Train Time = 0.865914
For C = 4.000000, Average Accuracy = 0.934497 and Average Train Time = 0.753888
For C = 16.000000, Average Accuracy = 0.936496 and Average Train Time = 0.901683

C increases the Accuracy upto a certain point, after which it dips a little, but it monotonically increases the train time.

b) The best value for C is 0.062500

c) Accuracy on Test set using the Best C is 0.932500.

3.4 Use linear SVM in LIBSVM:

Cross validation result:

For C = 0.000244, 3-fold Accuracy = 56.850000 and Train Time = 0.294022
For C = 0.000977, 3-fold Accuracy = 91.100000 and Train Time = 0.249768
For C = 0.003906, 3-fold Accuracy = 92.800000 and Train Time = 0.234485
For C = 0.015625, 3-fold Accuracy = 93.550000 and Train Time = 0.117493
For C = 0.062500, 3-fold Accuracy = 94.050000 and Train Time = 0.093462
For C = 0.250000, 3-fold Accuracy = 94.400000 and Train Time = 0.080221
For C = 1.000000, 3-fold Accuracy = 94.550000 and Train Time = 0.094903
For C = 4.000000, 3-fold Accuracy = 94.550000 and Train Time = 0.116154
For C = 16.000000, 3-fold Accuracy = 94.500000 and Train Time = 0.263324

a) The CV Accuracy is better for LIBSVM, but not by much.
b) The train time has definitely improved a lot.(in some cases 5-10 times).

3.5 Use kernel SVM in LIBSVM:

a) Polynomial Kernel: Cross Validation Results:

For C = 0.015625 and Degree = 1, 3-fold Accuracy = 70.100000 and Average Train Time = 0.837131
For C = 0.015625 and Degree = 2, 3-fold Accuracy = 55.800000 and Average Train Time = 0.839559
For C = 0.015625 and Degree = 3, 3-fold Accuracy = 55.750000 and Average Train Time = 0.854591
For C = 0.062500 and Degree = 1, 3-fold Accuracy = 91.350000 and Average Train Time = 0.658277
For C = 0.062500 and Degree = 2, 3-fold Accuracy = 86.500000 and Average Train Time = 0.819151
For C = 0.062500 and Degree = 3, 3-fold Accuracy = 77.350000 and Average Train Time = 0.853352
For C = 0.250000 and Degree = 1, 3-fold Accuracy = 93.100000 and Average Train Time = 0.455626
For C = 0.250000 and Degree = 2, 3-fold Accuracy = 92.900000 and Average Train Time = 0.527413
For C = 0.250000 and Degree = 3, 3-fold Accuracy = 91.400000 and Average Train Time = 0.638415
For C = 1.000000 and Degree = 1, 3-fold Accuracy = 93.850000 and Average Train Time = 0.302414
For C = 1.000000 and Degree = 2, 3-fold Accuracy = 94.600000 and Average Train Time = 0.383742
For C = 1.000000 and Degree = 3, 3-fold Accuracy = 94.450000 and Average Train Time = 0.488695
For C = 4.000000 and Degree = 1, 3-fold Accuracy = 94.300000 and Average Train Time = 0.242460
For C = 4.000000 and Degree = 2, 3-fold Accuracy = 95.050000 and Average Train Time = 0.248002
For C = 4.000000 and Degree = 3, 3-fold Accuracy = 95.950000 and Average Train Time = 0.297524
For C = 16.000000 and Degree = 1, 3-fold Accuracy = 94.300000 and Average Train Time = 0.241409
For C = 16.000000 and Degree = 2, 3-fold Accuracy = 96.200000 and Average Train Time = 0.267664
For C = 16.000000 and Degree = 3, 3-fold Accuracy = 96.300000 and Average Train Time = 0.266684
For C = 64.000000 and Degree = 1, 3-fold Accuracy = 94.400000 and Average Train Time = 0.270427
For C = 64.000000 and Degree = 2, 3-fold Accuracy = 96.050000 and Average Train Time = 0.229228
For C = 64.000000 and Degree = 3, 3-fold Accuracy = 96.350000 and Average Train Time = 0.240087
For C = 256.000000 and Degree = 1, 3-fold Accuracy = 94.450000 and Average Train Time = 0.370391
For C = 256.000000 and Degree = 2, 3-fold Accuracy = 95.900000 and Average Train Time = 0.248245
For C = 256.000000 and Degree = 3, 3-fold Accuracy = 96.100000 and Average Train Time = 0.255359
For C = 1024.000000 and Degree = 1, 3-fold Accuracy = 94.500000 and Average Train Time = 0.949745
For C = 1024.000000 and Degree = 2, 3-fold Accuracy = 95.900000 and Average Train Time = 0.241281
For C = 1024.000000 and Degree = 3, 3-fold Accuracy = 96.000000 and Average Train Time = 0.229090
For C = 4096.000000 and Degree = 1, 3-fold Accuracy = 94.550000 and Average Train Time = 5.588515
For C = 4096.000000 and Degree = 2, 3-fold Accuracy = 95.550000 and Average Train Time = 0.243851
For C = 4096.000000 and Degree = 3, 3-fold Accuracy = 96.000000 and Average Train Time = 0.235520
For C = 16384.000000 and Degree = 1, 3-fold Accuracy = 94.550000 and Average Train Time = 13.976358
For C = 16384.000000 and Degree = 2, 3-fold Accuracy = 95.600000 and Average Train Time = 0.390146
For C = 16384.000000 and Degree = 3, 3-fold Accuracy = 96.000000 and Average Train Time = 0.410122

**Best accuracy for Polynomial Kernel SVM with LIBSVM = 96.35, for C = 64, Degree = 3**

b) RBF Kernel: Cross Validation Results:

For C = 0.015625 and gamma = 6.103516e-05, 3-fold Accuracy = 55.750000 and Train Time = 0.936789
For C = 0.015625 and gamma = 2.441406e-04, 3-fold Accuracy = 55.750000 and Train Time = 0.920487
For C = 0.015625 and gamma = 9.765625e-04, 3-fold Accuracy = 55.750000 and Train Time = 0.919824
For C = 0.015625 and gamma = 3.906250e-03, 3-fold Accuracy = 55.750000 and Train Time = 0.895921
For C = 0.015625 and gamma = 1.562500e-02, 3-fold Accuracy = 65.150000 and Train Time = 0.904128
For C = 0.015625 and gamma = 6.250000e-02, 3-fold Accuracy = 72.250000 and Train Time = 0.922098
For C = 0.015625 and gamma = 2.500000e-01, 3-fold Accuracy = 55.750000 and Train Time = 0.950777
For C = 0.062500 and gamma = 6.103516e-05, 3-fold Accuracy = 55.750000 and Train Time = 0.889093
For C = 0.062500 and gamma = 2.441406e-04, 3-fold Accuracy = 55.750000 and Train Time = 0.893234
For C = 0.062500 and gamma = 9.765625e-04, 3-fold Accuracy = 55.750000 and Train Time = 0.958606
For C = 0.062500 and gamma = 3.906250e-03, 3-fold Accuracy = 83.100000 and Train Time = 0.988097
For C = 0.062500 and gamma = 1.562500e-02, 3-fold Accuracy = 91.800000 and Train Time = 0.732198
For C = 0.062500 and gamma = 6.250000e-02, 3-fold Accuracy = 90.650000 and Train Time = 0.743581
For C = 0.062500 and gamma = 2.500000e-01, 3-fold Accuracy = 63.250000 and Train Time = 0.938567
For C = 0.250000 and gamma = 6.103516e-05, 3-fold Accuracy = 55.750000 and Train Time = 0.906074
For C = 0.250000 and gamma = 2.441406e-04, 3-fold Accuracy = 55.750000 and Train Time = 0.941010
For C = 0.250000 and gamma = 9.765625e-04, 3-fold Accuracy = 86.150000 and Train Time = 0.958306
For C = 0.250000 and gamma = 3.906250e-03, 3-fold Accuracy = 91.850000 and Train Time = 0.675867
For C = 0.250000 and gamma = 1.562500e-02, 3-fold Accuracy = 93.350000 and Train Time = 0.493650
For C = 0.250000 and gamma = 6.250000e-02, 3-fold Accuracy = 94.550000 and Train Time = 0.491870
For C = 0.250000 and gamma = 2.500000e-01, 3-fold Accuracy = 92.100000 and Train Time = 0.800518
For C = 1.000000 and gamma = 6.103516e-05, 3-fold Accuracy = 55.750000 and Train Time = 0.949444
For C = 1.000000 and gamma = 2.441406e-04, 3-fold Accuracy = 86.350000 and Train Time = 0.944976
For C = 1.000000 and gamma = 9.765625e-04, 3-fold Accuracy = 91.750000 and Train Time = 0.658937
For C = 1.000000 and gamma = 3.906250e-03, 3-fold Accuracy = 93.400000 and Train Time = 0.505063
For C = 1.000000 and gamma = 1.562500e-02, 3-fold Accuracy = 94.600000 and Train Time = 0.313912
For C = 1.000000 and gamma = 6.250000e-02, 3-fold Accuracy = 96.200000 and Train Time = 0.341084
For C = 1.000000 and gamma = 2.500000e-01, 3-fold Accuracy = 95.800000 and Train Time = 0.647358
For C = 4.000000 and gamma = 6.103516e-05, 3-fold Accuracy = 86.400000 and Train Time = 0.932301
For C = 4.000000 and gamma = 2.441406e-04, 3-fold Accuracy = 91.850000 and Train Time = 0.680780
For C = 4.000000 and gamma = 9.765625e-04, 3-fold Accuracy = 93.300000 and Train Time = 0.442477
For C = 4.000000 and gamma = 3.906250e-03, 3-fold Accuracy = 94.300000 and Train Time = 0.312263
For C = 4.000000 and gamma = 1.562500e-02, 3-fold Accuracy = 95.550000 and Train Time = 0.254437
For C = 4.000000 and gamma = 6.250000e-02, 3-fold Accuracy = 96.450000 and Train Time = 0.264493
For C = 4.000000 and gamma = 2.500000e-01, 3-fold Accuracy = 95.400000 and Train Time = 0.625955
For C = 16.000000 and gamma = 6.103516e-05, 3-fold Accuracy = 91.850000 and Train Time = 0.637470
For C = 16.000000 and gamma = 2.441406e-04, 3-fold Accuracy = 93.200000 and Train Time = 0.422562
For C = 16.000000 and gamma = 9.765625e-04, 3-fold Accuracy = 94.050000 and Train Time = 0.363507
For C = 16.000000 and gamma = 3.906250e-03, 3-fold Accuracy = 95.050000 and Train Time = 0.318384
For C = 16.000000 and gamma = 1.562500e-02, 3-fold Accuracy = 96.350000 and Train Time = 0.224164
For C = 16.000000 and gamma = 6.250000e-02, 3-fold Accuracy = 96.900000 and Train Time = 0.289823
For C = 16.000000 and gamma = 2.500000e-01, 3-fold Accuracy = 95.300000 and Train Time = 0.644644
For C = 64.000000 and gamma = 6.103516e-05, 3-fold Accuracy = 93.300000 and Train Time = 0.432363
For C = 64.000000 and gamma = 2.441406e-04, 3-fold Accuracy = 93.800000 and Train Time = 0.299685
For C = 64.000000 and gamma = 9.765625e-04, 3-fold Accuracy = 94.500000 and Train Time = 0.250897
For C = 64.000000 and gamma = 3.906250e-03, 3-fold Accuracy = 95.450000 and Train Time = 0.250998
For C = 64.000000 and gamma = 1.562500e-02, 3-fold Accuracy = 96.150000 and Train Time = 0.210252
For C = 64.000000 and gamma = 6.250000e-02, 3-fold Accuracy = 96.650000 and Train Time = 0.276124
For C = 64.000000 and gamma = 2.500000e-01, 3-fold Accuracy = 95.300000 and Train Time = 0.647230

For C = 256.000000 and gamma = 6.103516e-05, 3-fold Accuracy = 93.750000 and Train Time = 0.305683
For C = 256.000000 and gamma = 2.441406e-04, 3-fold Accuracy = 94.200000 and Train Time = 0.243909
For C = 256.000000 and gamma = 9.765625e-04, 3-fold Accuracy = 95.000000 and Train Time = 0.216083
For C = 256.000000 and gamma = 3.906250e-03, 3-fold Accuracy = 96.150000 and Train Time = 0.221138
For C = 256.000000 and gamma = 1.562500e-02, 3-fold Accuracy = 96.200000 and Train Time = 0.208795
For C = 256.000000 and gamma = 6.250000e-02, 3-fold Accuracy = 96.300000 and Train Time = 0.260941
For C = 256.000000 and gamma = 2.500000e-01, 3-fold Accuracy = 95.300000 and Train Time = 0.609597
For C = 1024.000000 and gamma = 6.103516e-05, 3-fold Accuracy = 94.100000 and Train Time = 0.240323
For C = 1024.000000 and gamma = 2.441406e-04, 3-fold Accuracy = 94.450000 and Train Time = 0.224981
For C = 1024.000000 and gamma = 9.765625e-04, 3-fold Accuracy = 95.200000 and Train Time = 0.258187
For C = 1024.000000 and gamma = 3.906250e-03, 3-fold Accuracy = 96.450000 and Train Time = 0.241610
For C = 1024.000000 and gamma = 1.562500e-02, 3-fold Accuracy = 96.250000 and Train Time = 0.239906
For C = 1024.000000 and gamma = 6.250000e-02, 3-fold Accuracy = 96.300000 and Train Time = 0.268707
For C = 1024.000000 and gamma = 2.500000e-01, 3-fold Accuracy = 95.300000 and Train Time = 0.651654
For C = 4096.000000 and gamma = 6.103516e-05, 3-fold Accuracy = 94.250000 and Train Time = 0.222766
For C = 4096.000000 and gamma = 2.441406e-04, 3-fold Accuracy = 95.000000 and Train Time = 0.250513
For C = 4096.000000 and gamma = 9.765625e-04, 3-fold Accuracy = 96.250000 and Train Time = 0.322425
For C = 4096.000000 and gamma = 3.906250e-03, 3-fold Accuracy = 96.300000 and Train Time = 0.311550
For C = 4096.000000 and gamma = 1.562500e-02, 3-fold Accuracy = 96.150000 and Train Time = 0.226796
For C = 4096.000000 and gamma = 6.250000e-02, 3-fold Accuracy = 96.300000 and Train Time = 0.266909
For C = 4096.000000 and gamma = 2.500000e-01, 3-fold Accuracy = 95.300000 and Train Time = 0.686163
For C = 16384.000000 and gamma = 6.103516e-05, 3-fold Accuracy = 94.700000 and Train Time = 0.290691
For C = 16384.000000 and gamma = 2.441406e-04, 3-fold Accuracy = 95.200000 and Train Time = 0.370916
For C = 16384.000000 and gamma = 9.765625e-04, 3-fold Accuracy = 96.200000 and Train Time = 0.435764
For C = 16384.000000 and gamma = 3.906250e-03, 3-fold Accuracy = 96.300000 and Train Time = 0.386759
For C = 16384.000000 and gamma = 1.562500e-02, 3-fold Accuracy = 96.150000 and Train Time = 0.226120
For C = 16384.000000 and gamma = 6.250000e-02, 3-fold Accuracy = 96.300000 and Train Time = 0.273910
For C = 16384.000000 and gamma = 2.500000e-01, 3-fold Accuracy = 95.300000 and Train Time = 0.692664

**Best accuracy for RBF Kernel SVM with LIBSVM = 96.9, for C = 16, gamma = 0.0625**

RBF kernel performs better. See script libsvm.m which uses it and the optimum parameters. It reports an Accuracy = 96.9% (1938/2000) (classification), on the test set.