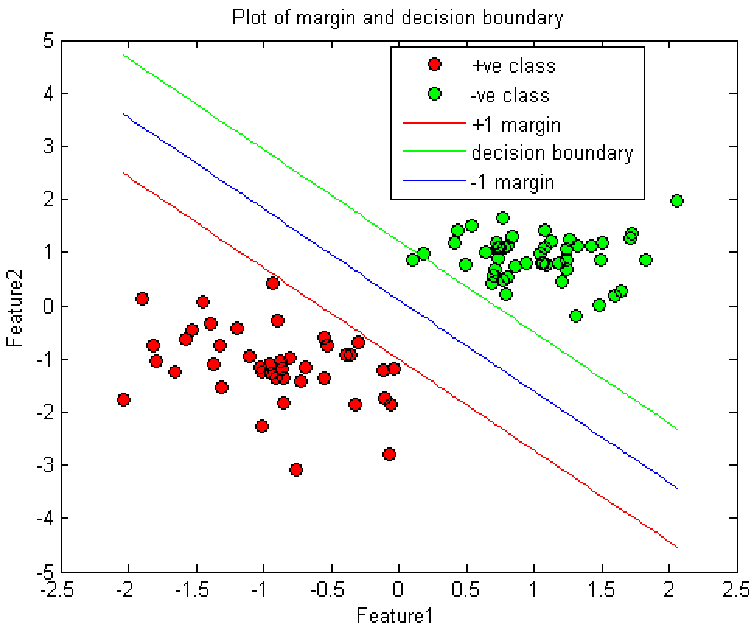
CSL407-Machine Learning

HW4(Jaskaran Singh)

Ans1)

Plot of decision boundaries and margins from the last fold

For w=(-1.5434   -0.8991) and w_0 or b=0.094361



Fold	w	b or w_0	Accuracy	Training Time
1	-1.5434 -0.8991	0.064244	100	0.132442 secs
2	-1.5541 -0.8749	0.087586	100	0.117506 secs
3	-1.5434 -0.8991	0.095092	100	0.116068 secs
4	-1.0103 -0.9956	0.021146	100	0.105223 secs
5	-1.5434 -0.8991	0.090600	100	0.118853 secs
6	-1.5434 -0.8991	0.086322	100	0.117578 secs
7	-1.5434 -0.8991	0.126809	100	0.125809 secs
8	-1.5434 -0.8991	0.084639	100	0.116253 secs
9	-1.5434 -0.8991	0.052740	100	0.119993 secs
10	-1.5434 -0.8991	0.094361	100	0.120984 secs

Ans2a)

Let the training set be $X = \{x_t, r_t\}_t$ $where\ r^t = \{+1\ if\ x^t \in C_1\ or -1\ if\ x^t \in C_2$

Find $w\ and\ w_0\ such\ that$

$$r^t(w^T x^t + w_0) \geq 1 - \varepsilon_t$$

$where\ \varepsilon_t = 0\ for\ correctly\ classified\ example\ far\ from\ the\ margin$
$0 < \varepsilon_t < 1\ for\ a\ correctly\ classified\ example\ but\ inside\ the\ margin$
$and\ \varepsilon_t \geq 1\ for\ an\ incorrectly\ classified\ example.$

Soft Error=$\sum \varepsilon_t$

We have to

$$\min \frac{1}{2}\left||w|\right|^2 + \ C\sum \varepsilon_t\ \ subject\ to\ r^t(w^T x^t + w_0) \geq 1 - \varepsilon_t\ \forall\ t$$

Where $\varepsilon_t \geq 0\ \ \forall\ t$

Using Lagrange multipliers $\alpha^t \geq 0\ and\ \beta^t \geq 0$

The primal problem is then

$$L_p = argMin_{w,w_0,\varepsilon_t} Max_{\alpha^t,\beta^t}\ \frac{1}{2}\left||w|\right|^2 + C\sum_t \varepsilon_t - \sum_t \alpha^t(r^t(w^T x^t + w_0) - 1 + \varepsilon_t) - \sum_t \beta^t \varepsilon^t$$

Applying KKT conditions,(n is the number of training examples)

1)$\frac{\delta L_p}{\delta w} = 0$

$$w = \sum_{t=1}^{n} \alpha^t r^t x^t$$

2)$\frac{\delta L_p}{\delta w_0} = 0$

$$\sum_{t=1}^{n} \alpha^t r^t = 0$$

3)$\frac{\delta L_p}{\delta \varepsilon_t} = 0$

$$C = \alpha^t + \beta^t$$

Substituting 1),2),3) in Lp we get the dual problem

$$\begin{aligned} L_d &= -\frac{1}{2}\sum_{\substack{1\leq t\leq n\\1\leq s\leq n}} \alpha^t\alpha^s r^t r^s (x^t)^T x^s + \sum_{t=1}^{n} \alpha^t + (C - \alpha^t - \beta^t)\sum_{t=1}^{n} \varepsilon^t \\ &= -\frac{1}{2}\sum_{\substack{1\leq t\leq n\\1\leq s\leq n}} \alpha^t\alpha^s r^t r^s (x^t)^T x^s + \sum_{t=1}^{n} \alpha^t \end{aligned}$$

Where

$$\sum_{t=1}^{n} \alpha^t r^t = 0$$

Since $\beta^t = C - \alpha^t$ and $\beta^t \geq 0$

Thus, $C - \alpha^t \geq 0$

Thus,$0 \leq \alpha^t \leq C \;\; for\, all\, 1 \leq t \leq n$

Since,Φ is the feature transformation function that transforms x^t into a high dimensional data point.

The dual problem in the transformed feature space is

$$L_d = -\frac{1}{2}\sum_{\substack{1\leq t\leq n\\ 1\leq s\leq n}} \alpha^t\alpha^s r^t r^s (\Phi(x^t))^T \Phi(x^s) + \sum_{t=1}^{n} \alpha^t$$

The inner product $(\Phi(x^t))^T \Phi(x^s)$ is equal to the kernel function K(x^t, x^s).

Thus,

$$L_d = -\frac{1}{2}\sum_{\substack{1\leq t\leq n\\ 1\leq s\leq n}} \alpha^t\alpha^s r^t r^s K(x^t, x^s) + \sum_{t=1}^{n} \alpha^t$$

Where

$$\sum_{t=1}^{n} \alpha^t r^t = 0$$

Ans2

c)

Classification accuracy using linear kernel=51.899749

Classification accuracy using polynomial kernel=99.50

Time taken for training linear kernel=0.217371secs(keeps changing even when running with the same seed)

Time taken for training polynomial kernel=0.225488secs(keeps changing even when running with the same seed)

In general, time taken for linear kernel is less than a polynomial kernel.

Ans3)

One needs to uncomment the implementation to test the code. Instructions are given in Ans3.m

Observations on *dataset2*

<u>My SVM(QP) with linear kernel</u>

Fold	Accuracy(%)	Training Time(secs)
1	52	218.943328

2	50.5	222.010305
3	49	208.935682
4	52.5	213.452922
5	47	221.278326
6	53	226.710451
7	58	210.091532
8	51.5	210.601275
9	56	217.314476
10	54.5	210.9946

Average accuracy per fold=52.4%

Average training time=216.033secs

My SVM(QP) with quadratic kernel

Fold	Accuracy(%)	Training Time(secs)
1	100	258.126325
2	99	264.000210
3	100	284.128869
4	100	265.374409
5	99.5	312.497133
6	100	273.380957
7	100	241.684927
8	100	234.838762
9	100	229.988886
10	100	226.114451

Average accuracy per fold=99.85

Average training time=259.0134 secs

Matlab SVM(SMO) with linear kernel

Fold	Accuracy(%)	Training Time(secs)
1	56.5	0.596029
2	55.5	0.588618
3	55.5	0.580925
4	56.5	0.572787
5	52.5	0.576702
6	56.5	0.547072
7	62	0.527391
8	52.5	0.574007
9	57.000000	0.570307
10	62.000000	0.608593

Average accuracy per fold=56.65

Average training time=0.57 secs

Matlab SVM(SMO) with quadratic kernel

Fold	Accuracy(%)	Training Time(secs)
1	100	0.173717
2	100	0.085969
3	100	0.095104
4	100	0.087932
5	99.5	0.471089
6	100	0.114536
7	100	0.115224
8	100	0.091879
9	100	0.218507
10	100	0.134842

Average accuracy per fold=99.95

Average training time=0.158880

Conclusions:

We conclude that

1. *dataset2* has a non-linear distribution of data since the linear model doesn't fit well and gives an accuracy of only 50% whereas the non-linear model gives an accuracy of close to 99%.
2. SMO is much faster than QP.

Ans4)

a)

Box Constraint Parameter C	Gaussian Kernel function width g	Average Cross Validation Accuracy(in %)
c=0.001000	g=0.001000	cv=69.860000
c=0.001000	g=0.010000	cv=75.320000
c=0.001000	g=0.100000	cv=75.460000
c=0.001000	g=1.000000	cv=13.140000
c=0.001000	g=10.000000	cv=40.020000
c=0.001000	g=100.000000	cv=44.460000
c=0.001000	g=1000.000000	cv=10.820000
c=0.010000	g=0.001000	cv=69.860000
c=0.010000	g=0.010000	cv=75.340000
c=0.010000	g=0.100000	cv=75.460000
c=0.010000	g=1.000000	cv=13.140000
c=0.010000	g=10.000000	cv=15.180000
c=0.010000	g=100.000000	cv=44.600000
c=0.010000	g=1000.000000	cv=10.900000
c=0.100000	g=0.001000	cv=73.000000

c=0.100000	g=0.010000	cv=89.320000
c=0.100000	g=0.100000	cv=88.760000
c=0.100000	g=1.000000	cv=13.140000
c=0.100000	g=10.000000	cv=14.180000
c=0.100000	g=100.000000	cv=44.800000
c=0.100000	g=1000.000000	cv=10.940000
c=1.000000	g=0.001000	cv=88.820000
c=1.000000	g=0.010000	cv=93.800000
c=1.000000	g=0.100000	cv=96.540000
c=1.000000	g=1.000000	cv=38.040000
c=1.000000	g=10.000000	cv=11.880000
c=1.000000	g=100.000000	cv=35.340000
c=1.000000	g=1000.000000	cv=10.960000
c=10.000000	g=0.001000	cv=92.620000
c=10.000000	g=0.010000	cv=95.500000
c=10.000000	g=0.100000	cv=96.620000
c=10.000000	g=1.000000	cv=41.420000
c=10.000000	g=10.000000	cv=12.260000
c=10.000000	g=100.000000	cv=35.420000
c=10.000000	g=1000.000000	cv=10.960000
c=100.000000	g=0.001000	cv=93.760000
c=100.000000	g=0.010000	cv=95.320000
c=100.000000	g=0.100000	cv=96.620000
c=100.000000	g=1.000000	cv=41.420000
c=100.000000	g=10.000000	cv=12.260000
c=100.000000	g=100.000000	cv=35.420000
c=100.000000	g=1000.000000	cv=10.960000
c=1000.000000	g=0.001000	cv=92.740000
c=1000.000000	g=0.010000	cv=95.320000
c=1000.000000	g=0.100000	cv=96.620000
c=1000.000000	g=1.000000	cv=41.420000
c=1000.000000	g=10.000000	cv=12.260000
c=1000.000000	g=100.000000	cv=35.420000
c=1000.000000	g=1000.000000	cv=10.960000

The models giving the highest cross validation accuracy for the *mnist* dataset are

c=10.000000,g=0.100000,cv=96.62%(Taking this model for comparison with neural network from hw3)

c=100.000000,g=0.100000,cv=96.62%

c=1000.000000,g=0.100000,cv=96.62%

b)

One can use Micro averaged F-measure to compare between the best models of ANN and SVM.

One having the higher F-measure is a better classifier.

In case of SVM, we take the best model to be c=10,g=0.1

In case of ANN, we take the best model to be H=500.

On the same validation set:

Fsvm=.962376

Fmlp=.929703

Thus, SVM performs better than MLP on the validation set.