T-61.6040 Assignment-01/2011

Santosh Tirunagari (245577)
santosh.tirunagari@aalto.fi

September 22, 2011

AIM

Implemention of soft margin SVM algorithm discussed in the lecture using linear kernel and Gaussian kernel for classification of hand written digits (3 and 8).

$$k_{LIN} = (x_i, x_j) = < x_i, x_j > \; and \; C = \{0.0001, 0.001, 0.01, 0.1, 1\} \quad (1)$$

$$k_{GAU} = (x_i, x_j) = exp(-\|x_i x_j\|_2^2 / s^2), \; C = 1, \; and \; s = \{8, 16, 32, 64, 128\} \quad (2)$$

Datasets

The datasets for hand written digits used in the assignment are training set and test set with 500 samples and 256 features. Class labels for training and test set are denoted as $+1$ for 3 and -1 for 8. There are 250 samples each for handwritten 3s and 8s for both the training and test sets.

Implementation

Load and zscore normalize both the train and test data using zscore matlab command. Zscore normalization can also be done using the method described in the lecture.

```
X = zscore(usps_3_vs_8_train_X);
X_t = zscore(usps_3_vs_8_test_X);
```

Calculate linear kernel function.

```
K = X*X';
```

Solve SVM dual optimization problem in standard QP form using quadprog function in matlab and find alpha values.

```
H = (Y*Y').*K;          A = [];
Aeq = Y';               l = zeros(500,1);
c = -1*ones(500,1);     b = [];
beq = 0;                u = C(k)*ones(500, 1);
options = optimset('Algorithm','interior-point-convex');
alpha = quadprog(H, c, A, b, Aeq, beq, l, u,[],options);
```

Compute the near boundary coefficients and move near boundary alpha values to boundaries

```
alpha(alpha < C(k) * 0.001) = 0;
alpha(alpha > C(k)*0.999999999999) = C(k);
```

Compute value for bias parameter b

```
sv = find(alpha >0 & alpha<C(k));
sv_one = zeros(500,1);
sv_one(sv,1) = 1;
b = sv_one'*(Y-((alpha.*Y)'*K'))/sum(sv_one);
```

Compute the decision function on training set

```
Ki = X(sv,:)*X';
temp = bsxfun(@plus,Ki'*(alpha(sv,:).*Y(sv,:)),b);
```

Compute the decision function on test set

```
Ki = X(sv,:)*X_t';
res = bsxfun(@plus,Ki'*(alpha(sv,:).*Y(sv,:)),b);
```

Threshold the decisions to get proper +1 and −1

```
res(res>=0) = 1;
res(res<0) = -1;
```

Calculate the misclassification errors

```
r = sum(res~=Y_t);
r=(r/500)*100;
```

Calculate the % of support vectors utilised.

```
s = length(sv);
s=(s/500)*100;
```

Plot the graphs showing the misclassification errors.

```
hut = log10(C);
plot(hut,result_train,'k-*','LineWidth',5,'MarkerSize',10);
hold on
plot(hut,result_test,'r-o','LineWidth',5,'MarkerSize',10);
```

Plot the graph showing the % of support vectors utilised.

```
hut = log10(C);
plot(hut,support,'k-*','LineWidth',5,'MarkerSize',10);
```

The same implementation is carried out for all the values of $C = \{0.0001, 0.001, 0.01, 0.1, 1\}$ and mis-classification error and support vector count percentages are computed. These values are given in table 1.

The implentation of Gaussian kernel is the same as the implementation of linear kernel except the kernel computation. Gaussian kernel is computed as

```
temp = (S(k).^2);
K = exp(-bsxfun(@plus,sum(X.^2,2),bsxfun(@plus,sum(X.^2,2)',-2.*X*X'))./temp);
```

The decision function for Gaussian kernel is computed as follows:

```
temp = (S(k).^2);
temp1=-2.*X(sv,:)*X_t';
temp3=sum(X_t.^2,2)';
Ki = exp(-bsxfun(@plus,sum(X(sv,:).^2,2),bsxfun(@plus,temp3,temp1))./temp);
res = bsxfun(@plus,Ki'*(alpha(sv,:).*Y(sv,:)),b);
```

The Gaussian kernel implementation is carried out for all the values of $C = 1$, *and* $s = \{8, 16, 32, 64, 128\}$ and mis-classification error and support vector count percentages are computed. These values are given in table 2.

Running of code for linear kernel in matlab: svm_kernel
Running of code for Gaussian kernel in matlab: svm_guass

Results

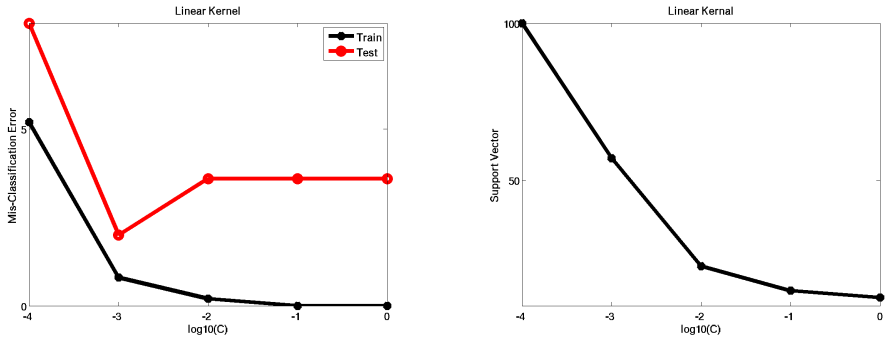
Table 1 and figure 1 shows that the support vectors are decreasing as the value of C increases. The mis-classification error for the test data is minimum at $C = 0.001$.

Table 1: Mis-classification errors and % of support vectors utilized in implementing linear kernel

C	Mis-Classification Error on Training set	Mis-Classification Error on Test set	% of Support vectors
0.0001	5.200	8.000	100.000
0.001	0.800	2.000	57.000
0.01	0.200	3.600	22.600
0.1	0.000	3.600	14.800
1	0.000	3.600	12.600

Table 2 and figure 2 shows that the increase or decrease in the count of support

Figure 1: Plot of mis-classification errors and % of support vectors utilized in implementing linear kernel.

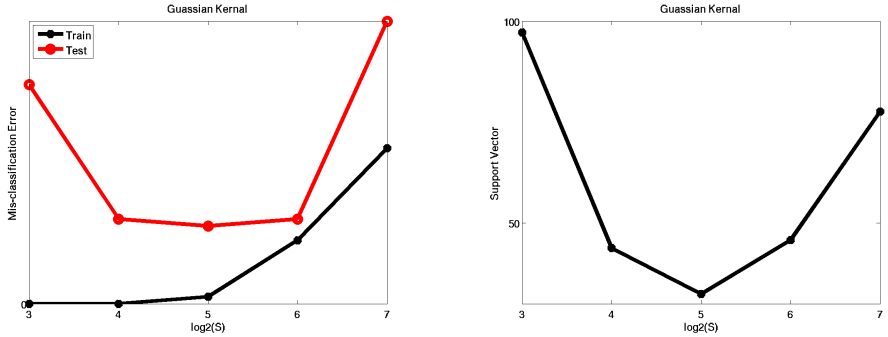


vectors utilised are not dependent on the value of S. The mis-classification error for the test data is minimum at $S = 32$.

Table 2: Mis-classification errors and % of support vectors utilized in implementing Gaussian kernel

S	Mis-Classification Error on Training set	Mis-Classification Error on Test set	% of Support vectors
8	0.0000	0.0620	97.2000
16	0.0000	0.0240	43.8000
32	0.0020	0.0220	32.4000
64	0.0180	0.0240	45.8000
128	0.0440	0.0800	77.6000

Figure 2: Plot of mis-classification errors and % of support vectors utilized in implementing Gaussian kernel.



References

[1] Scholkopf and Smola (2002)
[2] http://en.wikipedia.org/wiki/Support_vector_machine
[3] http://www.mathworks.se/help/toolbox/optim/ug/quadprog.html

Attachments

Matlab code svm_kernal.m and svm_guass.m and datasets have been archived along with this report.