CSE512 Fall 2018 - Machine Learning - Homework 4

Your Name: Solar ID: Astitv Nagpal

NetID: 112008011

email address: astitv.nagpal@stonybrook.edu

Names of people whom you discussed the homework with: Ayush Garg

Answer 1 : 1.1

To prove that $LOOCV_{err} \leq \frac{m}{n}$:

Concept : We choose training sets & select one of the set. We then train the classifier on the basis of all the remaining training examples. We then test the final output classifier on the initial set that was taken apart.

Also, more - importantly we can write it as:

$$LOOCV_{err} = \frac{1}{n} \sum_{i=1}^{n} L(y_i, f x_i \mid \theta_n^i \theta_0^i)$$

We have support vectors & non-support vectors in any given data set. ∴ we have

$$LOOCV_{err} = \frac{1}{n} \sum_{i=1}^{p} L(y_i, (f(x_i, \theta_p^i, \theta_0^i)) +$$

$$\sum_{i=1}^{q} L(y_i, f(x_i, \theta_q^i, \theta_0^i))$$

This is nothing but :

$$LOOCV_{err} = \frac{1}{n}\left(\text{loss due to support vector} + \text{loss due to non-support vector}\right)$$

We know that all the non-support vectors doesn't affect our final solution. But the support vectors are an integral part for deciding the line / plain separator. Therefore we can remove or exclude the loss due to non-support vectors.

Scanned by CamScanner

Hence, we can conclude that LOOCV is bounded by only the number of support vectors in a data set.

$$\therefore \boxed{LOOCV_{en} \le \frac{m}{n}}$$

$m \to$ # of support vectors

$n \to$ # total data points.

1.2 We know that in a case of a Kernel we can write out:

$$LOOCV_{en} = \frac{1}{n} \sum_{i=1}^{n} \alpha_i y_i K(x_i, x) \quad \text{——— (i)}$$

Here, when we use the kernel trick we see that for any dimensional data not linearly separable in d-1 dim can be done in dimension d.

Hence, we ultimately boils down to the same concept of a Linear ~~kernel~~ SVM by applying an appropriate Kernel.

$\therefore$ we can say that we try to chose a kernel such that we end up making a linear SVM. Hence, in our previous question

we saw that $LOOCV_{en} \le \frac{m}{n}$ ie $LOOCV \propto m$

$LOOCV \propto \frac{1}{n}$.

Hence, we can conclude that even in a general case the bound holds true for general Kernel case.

ANSWER : 2

(2.1) $\underset{\alpha}{\text{maximize}} \sum_{j=1}^{n} \alpha_j - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i \alpha_i y_j \alpha_j K(x_i, x_j)$ ——— (i)

s.t. $\sum_{j=1}^{n} y_j \alpha_j = 0$

$0 \leq \alpha_j \leq C \quad \forall j$

The quadprog form can be written as:

$$\underset{x}{\text{minimize}} \ \frac{1}{2} x^T H x + f^T x$$

s.t. $Ax \leq b$

$Cx = d$

$lb \leq x \leq ub$

Hence, we can write eqn (i) as:

$$\underset{\alpha}{\text{maximize}} \sum_{j=1}^{n} \alpha_j - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i \alpha_i y_j \alpha_j K(x_i, x_j)$$

We know a Linear Kernel is $K(x_i, x_j) = x_i \cdot x_j$

$\therefore$ we can write

$$\underset{\alpha}{\text{minimize}} \sum_{j=1}^{n} \alpha_j - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i \alpha_i y_j \alpha_j x_i x_j$$

s.t. $\sum_{j=1}^{n} y_i \alpha_j = 0$

$0 \leq \alpha_j \leq C \quad \forall j$

$\Rightarrow$ Quadprog requires the following arguments:

H, f, A, b, Aeq, beq, lb, ub

~~exa~~

$H = y * y^T .* X * X^T$

$f = -\text{ones}(1, m) = [-1, -1, -1, \ldots -1]_{m \text{ times}}$ → as we need to minimize the func".

↳ m = size of H

$A = [\ ]$

$b = [\ ]$

→ as no constraints are required

$Aeq = y^T$

$beq = 0$

$lb = \text{zeros}(m, 1) = [0, 0, 0, \ldots, 0]^T_{m \text{ times}}$

$ub = C * \text{ones}(m, 1) = C[1, 1, 1, \ldots 1]^T_{m \text{ times}}$

Question 2

2.1

```
X_t = double(X_train);
X = transpose(X_t);
y = double(y_train);
y_t = transpose(y);
X_dot = X*X_t;
y_dot = y*y_t;
H = y_dot.*X_dot;
[m,n] = size(H);
f = -ones(1,m);
A = [];
b = [];
Aeq = y_t;
beq = 0;
LB = zeros(m,1);
UB = c_val*ones(m,1);
[alpha, obj] = quadprog(H,f,A,b,Aeq,beq,LB,UB);
```

2.3

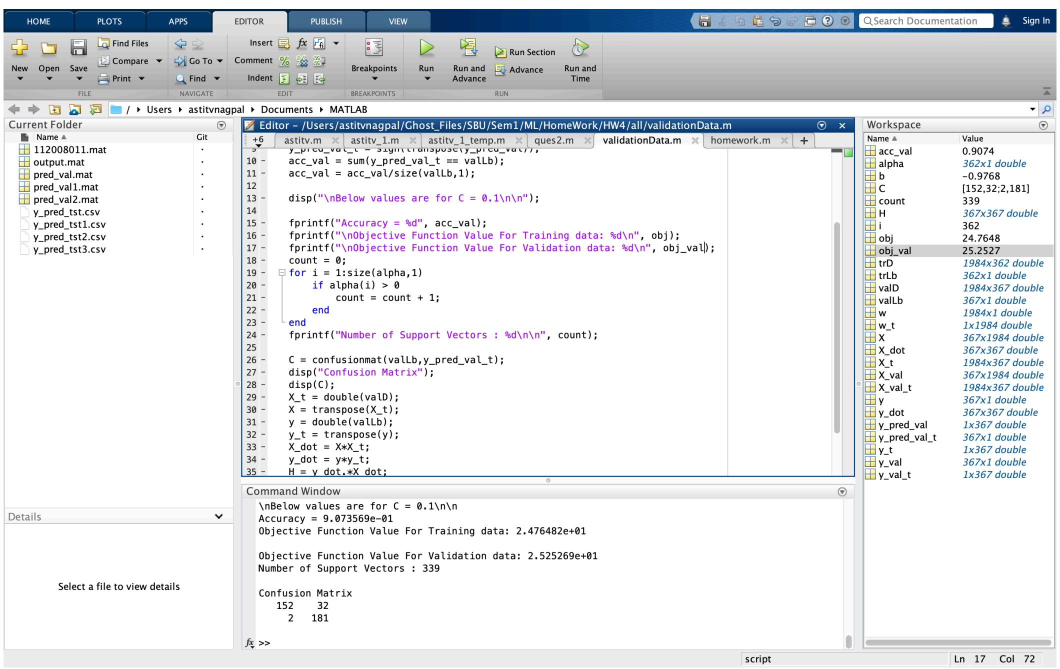
```
function [w,b, alpha, obj] = ques2(X_train, y_train, c_val)
X_t = double(X_train);
X = transpose(X_t);
y = double(y_train);
y_t = transpose(y);
X_dot = X*X_t;
y_dot = y*y_t;
H = y_dot.*X_dot;
[m,n] = size(H);
f = -ones(1,m);
A = [];
b = [];
Aeq = y_t;
beq = 0;
LB = zeros(m,1);
UB = c_val*ones(m,1);
[alpha, obj] = quadprog(H,f,A,b,Aeq,beq,LB,UB);
obj = obj*-1;
for i = 1:size(alpha,1)
    if alpha(i) <= 0.00001
        alpha(i) = 0;
    end
end
alpha_t = transpose(alpha);
temp_w = alpha_t.*X_t;
w = temp_w*y;
w_t = transpose(w);
```

```
ind = 0;
for n = 1:length(alpha)
    if alpha(n) < vpa(c_val)
        ind = n;
        break
    end
end

b = y(ind) - w_t*X_t(:, ind);

y_pred = w_t*X_t + b;
y_pred_t = sign(transpose(y_pred));
accuracy = sum(y_pred_t == y_train);
acc_train = accuracy/size(y_train,1);
disp(acc_train);
end
```

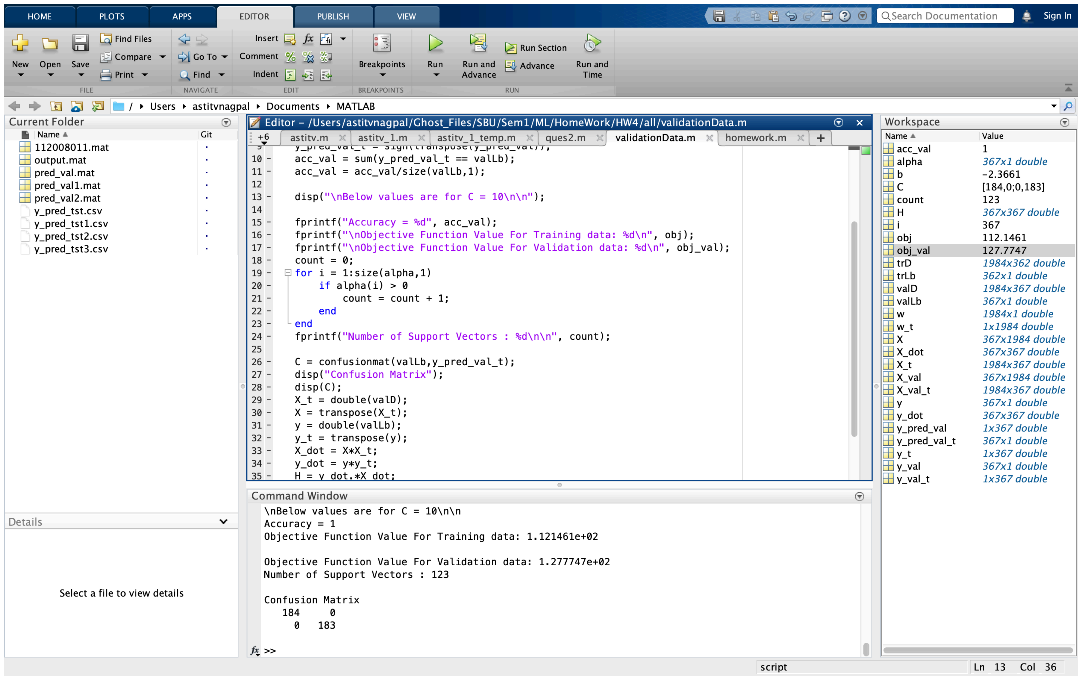
2.4



C = 0.1
Accuracy = 0.90735
Obj Train = 24.76482
Obj Val = 25.25269
No of support vectors = 339
Confusion Matrix

152	32
2	181

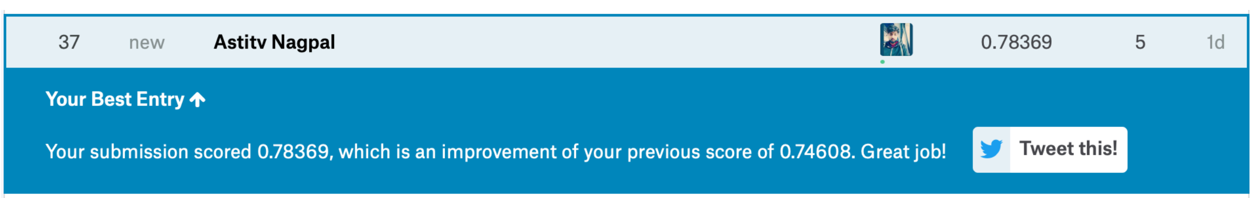
2.5



C = 10
Accuracy = 1.0
Obj Train = 112.1461
Obj Val = 17.7747
No of support vectors = 123
Confusion Matrix

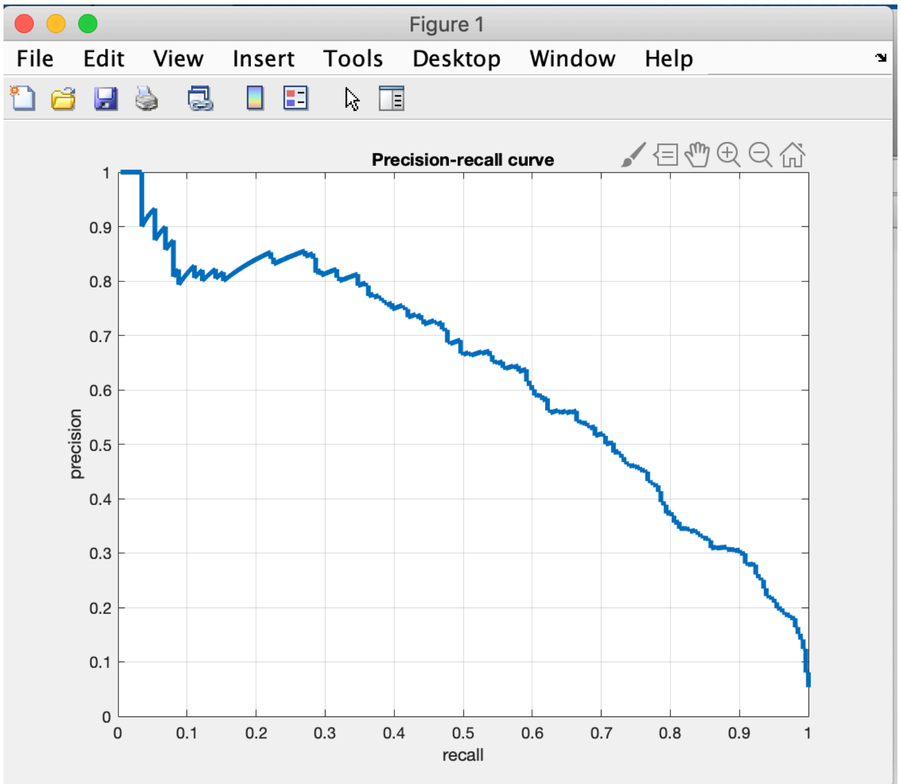
184	0
0	183

2.6



Answer 3.4

1

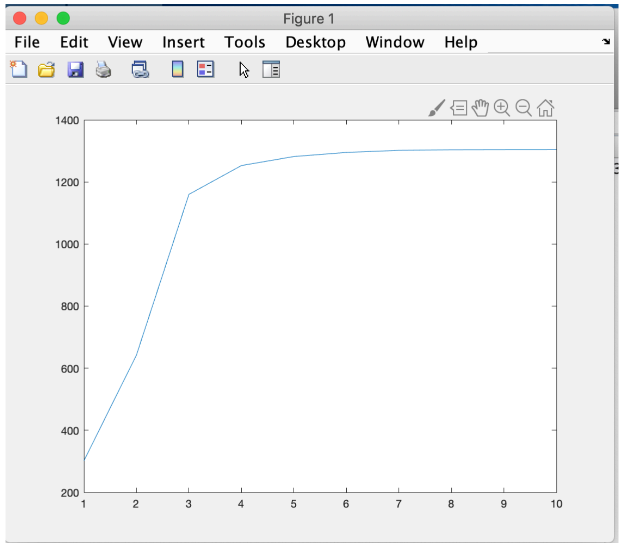



```
Ub detection 92/92 (100.00%), elapse time:    59.6s
results have been saved to output_val.mat
    0.6362
```

~~

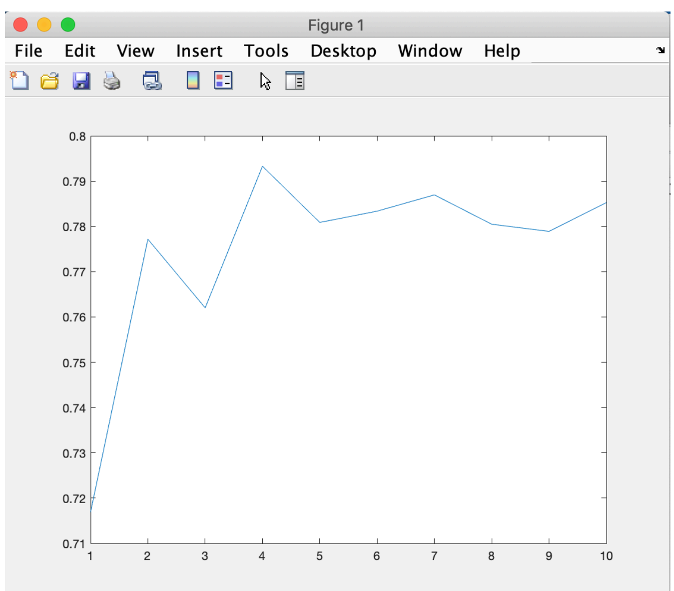
ap = 0.6362

3



Obj Function Curve:

Values Obtained:
301.698029236053 641.935143196331 1159.89595281663 1252.72746518837 1281.81780693147 1295.02202226784 1301.68114348140 1303.63498978279 1304.18797922127 1304.46197579301



Average Precision Curve:

Values Obtained:
0.716901445318868 0.777178936242526 0.762018475808314
0.793288180853029 0.780879773674237 0.783365530654037
0.786963682699916 0.780480472484640 0.778899599757491
0.785245496107708

4

Leaderboard Rank and score

36	Astitv Nagpal	112008011	0.749524	0.733819