

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 $LOOCV_{err} \leq \frac{w}{n}$.

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:

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

We have support vector & non-support vectors in any given data set. \therefore we have

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

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 don't affect our final solution. But the support vectors are an integral part for deciding the line / plane separator. Therefore we can remove or include the loss due to non-support vectors.

Hence, we can conclude that LOOCV is bounded by only ⁽²⁾ the number of support vectors in a data set.

$$\therefore \boxed{LOOCV_m \leq \frac{m}{n}} \quad \begin{array}{l} m \rightarrow \# \text{ of support vectors} \\ n \rightarrow \# \text{ total data points.} \end{array}$$

(1.2) We know that in a case of a kernel we can write as:

$$LOOCV_m = \frac{1}{n} \sum_{i=1}^n x_i y_i k(x_i, x) \quad \text{--- (i)}$$

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 look down to the same concept of a Linear ~~kernel~~ SVM by applying an appropriate kernel.

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

we saw that $LOOCV_m \leq \frac{m}{n}$ i.e. $LOOCV \propto m$ & $LOOCV \propto \frac{1}{n}$.

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

ANSWER : 2

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

(i)

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

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

The quadprog form can be written as :

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

$$\text{s.t.} \quad Ax \leq b$$

$$Cx = d$$

$$lb \leq x \leq ub$$

Hence, we can write eqn (i) as :

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

We know a linear kernel is $K(x_i, x_j) = x_i \cdot x_j$

\therefore we can write

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

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

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

\Rightarrow Quadprog requires the following arguments :

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

~~Q. 4~~

$$H = Y * Y^T + X * X^T$$

$$f = \text{ones}(1, m) = [-1, -1, -1, \dots, -1] \xrightarrow{\text{as we need to minimize the func.}} \substack{\text{m times} \\ \rightarrow m = \text{size of } H}$$

$$A = []$$

$$b = []$$

\rightarrow as no constraints are required

$$Aeq = Y^T$$

$$beq = 0$$

$$lb = \text{zeros}(m, 1) = [0, 0, 0, \dots, 0]^T \substack{\text{m times}$$

$$ub = c * \text{ones}(m, 1) = c[1, 1, 1, \dots, 1]^T \substack{\text{m times}$$

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);
```



```

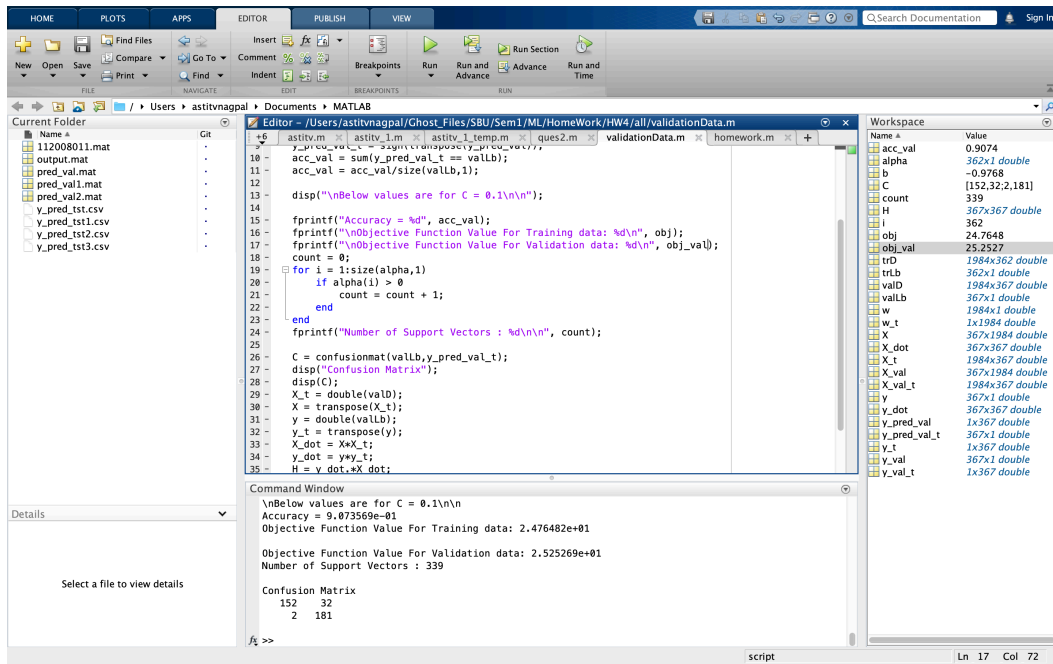
ind = 0;
for n = 1:length(alpha)
    if alpha(n) < vpa(c_val)
        ind = n;
        break
    end
end

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

```

% Editor - /Users/astitvnagpal/Chost_Files/SBU/Sem1/ML/HomeWork/HW4/all/validationData.m
% astitv.m astitv_1.m astitv_1_temp.m ques2.m validationData.m homework.m
% y_pred_val_t = sigmoid(support_vec_pred_val_t);
% acc_val = sum(y_pred_val_t == valLb);
% acc_val = acc_val/size(valLb,1);
% disp("\nBelow values are for C = 10\n\n");
% fprintf("Accuracy = %d", acc_val);
% fprintf("\nObjective Function Value For Training data: %d\n", obj);
% fprintf("\nObjective Function Value For Validation data: %d\n", obj_val);
% count = 0;
% for i = 1:size(alpha,1)
%     if alpha(i) > 0
%         count = count + 1;
%     end
% end
% fprintf("Number of Support Vectors : %d\n\n", count);
% C = confusionmat(valLb,y_pred_val_t);
% disp("Confusion Matrix");
% disp(C);
% X_t = double(valD);
% X = transpose(X_t);
% y = double(valLb);
% y_t = transpose(y);
% X_dot = X*X_t;
% y_dot = y*y_t;
% H = y_dot.*X_dot;

```

Command Window

```

\nBelow values are for C = 10\n\n
Accuracy = 1
Objective Function Value For Training data: 1.121461e+02
Objective Function Value For Validation data: 1.277747e+02
Number of Support Vectors : 123

Confusion Matrix
184    0
  0   183

```

Workspace

Name	Value
acc_val	1
alpha	367x1 double
b	-2.3661
C	[184 0; 0 183]
count	123
H	367x367 double
i	367
obj	112.1461
obj_val	127.7747
trD	1984x362 double
trLb	362x1 double
valD	1984x367 double
valLb	367x1 double
w	1984x1 double
w_t	1x1984 double
X	367x1984 double
X_dot	367x367 double
X_t	1984x367 double
X_val	367x1984 double
X_val_t	1984x367 double
y	367x1 double
y_dot	367x367 double
y_pred_val	1x367 double
y_pred_val_t	367x1 double
y_t	1x367 double
y_val	367x1 double
y_val_t	1x367 double

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

37 new **Astitv Nagpal** 0.78369 5 1d

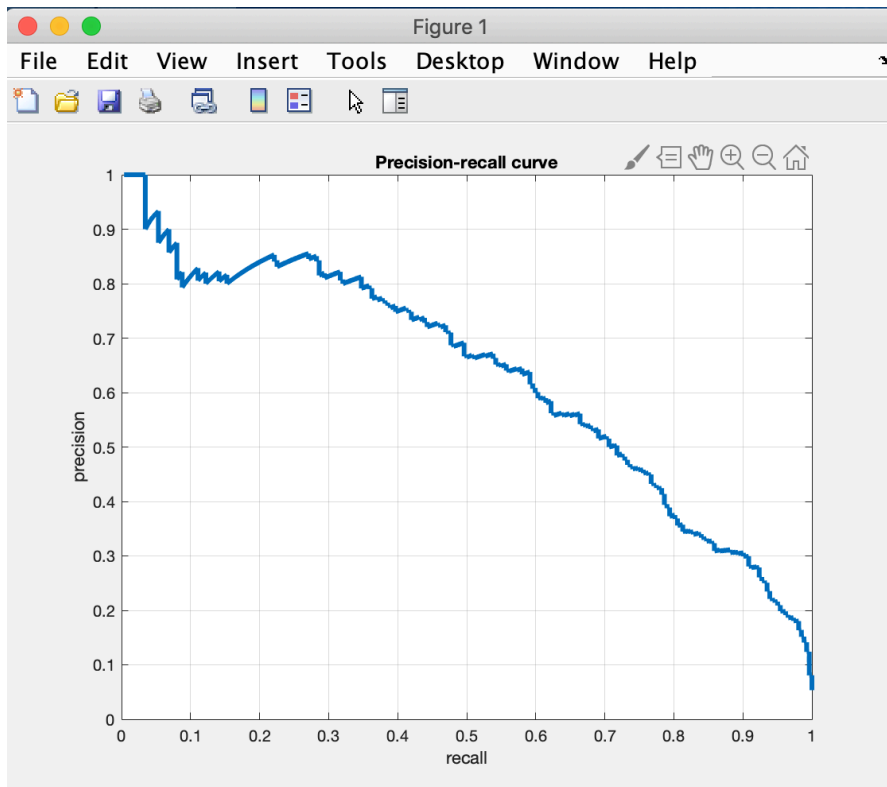
Your Best Entry

Your submission scored 0.78369, which is an improvement of your previous score of 0.74608. Great job!

Tweet this!

Answer 3.4

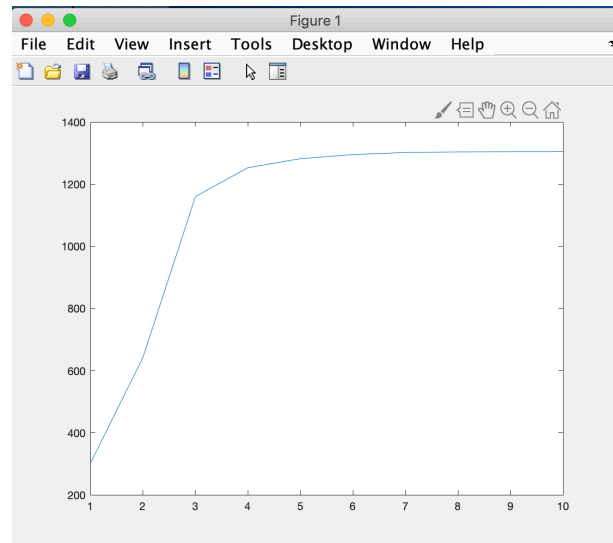
1



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

ap = 0.6362

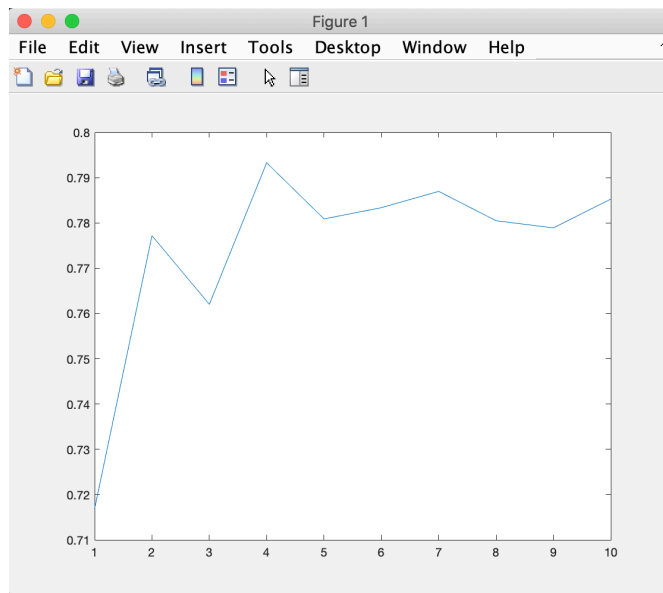
3



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
----	---------------	-----------	----------	----------