CSE512 Fall 2019 - Machine Learning - Homework 4

Your Name: Aveena Kottwani

Solar ID:112689816

NetID email address: akottwani@cs.stonybrook.edu, aveena.kottwani@stonybrook.edu

Names of people whom you discussed the homework with: Chaitra Hegde

1. Question 1 – Support Vector Machines

1.1. Linear case: Consider training a linear SVM on linearly separable dataset consisting of n points. Let m be the number of support vectors obtained by training on the entire set. Show that the LOOCV error is bounded above by m /n. Hint: Consider two cases: (1) removing a support vector data point and (2) removing a non-support vector data point.

1. Support Vector Machines

Linear SVM consists of n datapoints linearly separable. There are 'm' number of support vectors. The total loss using LOOCV error is given by calculating all the decision rule on the training data except a single element & testing on the removed element.

There are two cases for this:-

Case (i) Removing the single element which is a support vector data points.

let us denote number of errors in the leave one out procedure by $\mathcal{L}(x_1, y_1 \ldots x_n, y_n)$

Expected generalization error:-

$$E p_{err}^{n-1} = \frac{1}{n} E\left(\mathcal{L}(x_1, y_1 - - - x_n, y_n)\right)$$

where p_{err}^{n-1} is probability of test error for the machine trained on size$(n-1)$

⇒ Removing the single element which is a support vector would affect the LOOCV error, as the margin depends on the support vector data points, not on the non-support vectors.

⇒ If we denote f^o as the classifier obtained for all training examples present & f^i for the single element(i) removed,

$$\mathcal{L}(x_1, y_1 \ldots x_n, y_n) = \sum_{p=1}^{n} \psi\left(-y_p f^p(x_p)\right)$$

which is $\rightarrow$

$$\mathcal{L}(x_1, y_1 \ldots x_n, y_n) = \sum_{p=1}^{n} \psi\left(-y_p f^0(x_p) + y_p\left(f^0(x_p) - f^p(x_p)\right)\right)$$

Thus when support vector datapoint is removed, we get upper bound as $U_p = y_p\left(f^0(x_p) - f^p(x_p)\right)$

Hence we get,

$$\mathcal{L}(x_1, y_1 \ldots x_n, y_n) \leq \sum_{p=1}^{n} \psi(U_p - 1)$$

since for hard margin SVMs,

$y_p f^0(x_p) \geq 1$ and ψ is monotonically increasing

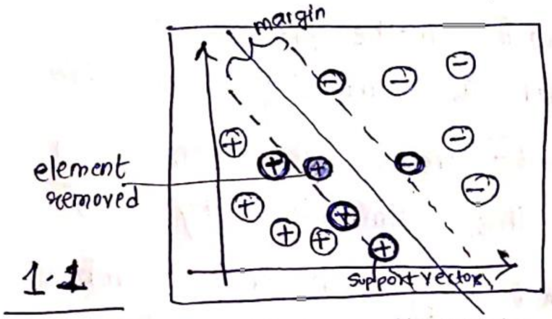
<u>Case 2:</u> When removing a single element which is a non-support vector from the training set does not change the solution computed.

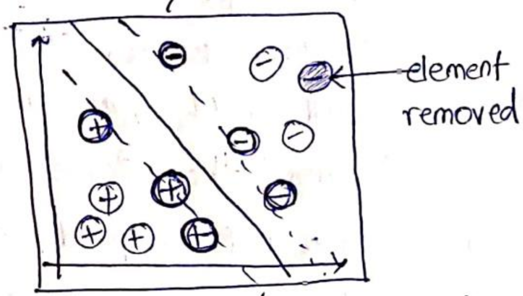
$\Rightarrow$ i.e $U_p \doteq f^0(x_p) - f^p(x_p) = 0$

for x_p non-support vector.

So we can restrict the sum to support vectors and upper bound each term in the sum by 1, which gives the following bound on the number of ~~support vectors~~ errors made by the leave one out procedure m/n

m – number of support vectors.





1.1

Case 1 : Element removed is a support vector

Margin is enlarged as the element removed was not considered during the training, and classifies

Case 2: Element removed is a non-support vector

Non-support vector does not affects the margin.

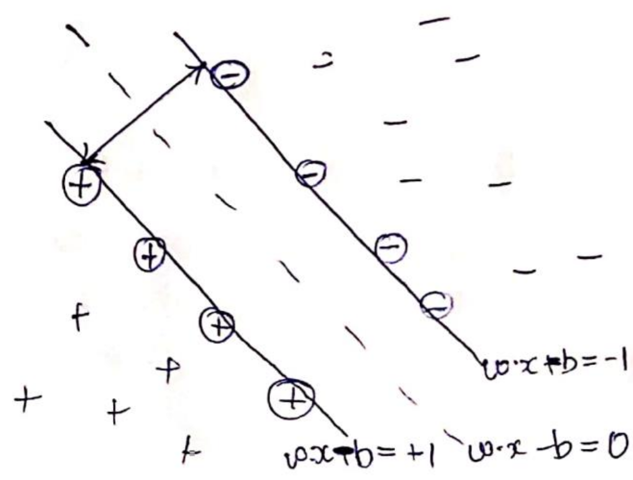
1.2. General case: Now consider the same problem as above. But instead of using a linear SVM, we will use a general kernel. Assuming that the data is linearly separable in the high dimensional feature space corresponding to the kernel, does the bound in previous section still hold? Explain why or why not.

1.2 For general case, the general kernel is considered which still gives linearly separable data in the high dimensional feature space corresponding to the kernel. The bound in the previous case still hold, as the data is linearly separable and the leave-one out cross validation error does not depend on the dimensionality of the feature space but only on the number of support vectors. Linear operation in the feature space is equivalent to non-linear operation in input space.

$$\boxed{\text{Leave-one out cross validation error} = \frac{\text{Number of support vectors } (m)}{\text{Number of training examples } (n)}}$$

The LOOCV error is still bounded by $\left(m/n\right)$,

as per ~~th~~ ~~also~~ the two cases in the previous questions



2. Question 2 – Implementation of SVMs

2.1. write down what H,f, A, b, Aeq, beq, lb, ub

2. Implementation of SVMs

2.1 Write down what H, f, A, b, Aeq, beq, lb, ub

Sol: According to the dual objective function of SVM,

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

$$\text{s.t.} \sum_{j=1}^{n} y_j \alpha_j = 0 \quad \text{— (2)}$$

$$0 \leq \alpha_j \leq C \quad \forall j \quad \text{— (3)}$$

But the function equation for quadprog is

$$\min_{x} \frac{1}{2} x^T H x + f^T x$$

$$\text{s.t} \quad Ax \leq b$$
$$Cx = d$$
$$l \leq x \leq u \quad \text{— (4)}$$

Since the quadprog gives value for minimizing the objective function, the dual equation objective function is

$$\min_{\alpha} \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} K(x_i, x_j) \alpha_i \alpha_j y_i y_j + (-1) \sum_{j=1}^{n} \alpha_j \quad \text{— (5)}$$

α corresponds to the x term in equation (4)

Hence, the values of different variables are

$k(x_i, x_j) = K = X X^T = x_i * x_j$ ← linear kernel

$H = (X X^T)(Y Y^T) = (x_i * x_j)(y_i * y_j)$

Linear Inequality constraints: $\left.\begin{array}{l} A = [\,] \\ b = [\,] \end{array}\right\}$ No linear inequality

Linear Equality constraints: $\left\{\begin{array}{l} Aeq = Y^T = (y_j) \\ beq = 0 \end{array}\right\}$ from $\sum_{j=0}^{n} y_j \alpha_j = 0$

$f = \begin{bmatrix} -1 \\ \vdots \\ -1 \end{bmatrix}$ $\updownarrow$ no. of datapoints (n) $\Big\}$ from $(-1)\sum_{j=1}^{n} \alpha_j$

(-1) is the multiplier for the term $\sum_{j=1}^{n} \alpha_j$

[n = number of data points]

For upper & lower bounds, we used the following condition,

$$0 \leq \alpha_j \leq C$$

~~lb = [0~~ $lb = 0$

~~ub =~~ ub = upperbound $- \begin{bmatrix} C \\ \vdots \\ C \end{bmatrix}$ $\updownarrow$ number of datapoints.

2.2. Use quadratic programming to optimize the dual SVM objective. In Matlab, you can use the function quadprog.

2.3. Write a program to compute w and b of the primal from α of the dual. You only need to do this for linear kernel

2.4. Set C = 0.1, train an SVM with linear kernel using trD, trLb in q2_1data.mat (in Matlab, load the data using load q2 1 data.mat). Test the obtained SVM on valD, valLb,and report the accuracy, the objective value of SVM, the number of support vectors, and the confusion matrix.

```
Accuracy
    0.9074

Objective Value
   24.7648

Number of Support Vectors
   333

Confusion Matrix
   152    32
     2   181
```

C	**Accuracy**	**Objective Value**	**Number of Support Vectors**
0.1	0.9074	24.7648	333

Confusion Matrix	Predicted Positive	Predicted Negative
Actual Positive	152	32
Actual Negative	2	181

2.5. Repeat the above question with C = 10

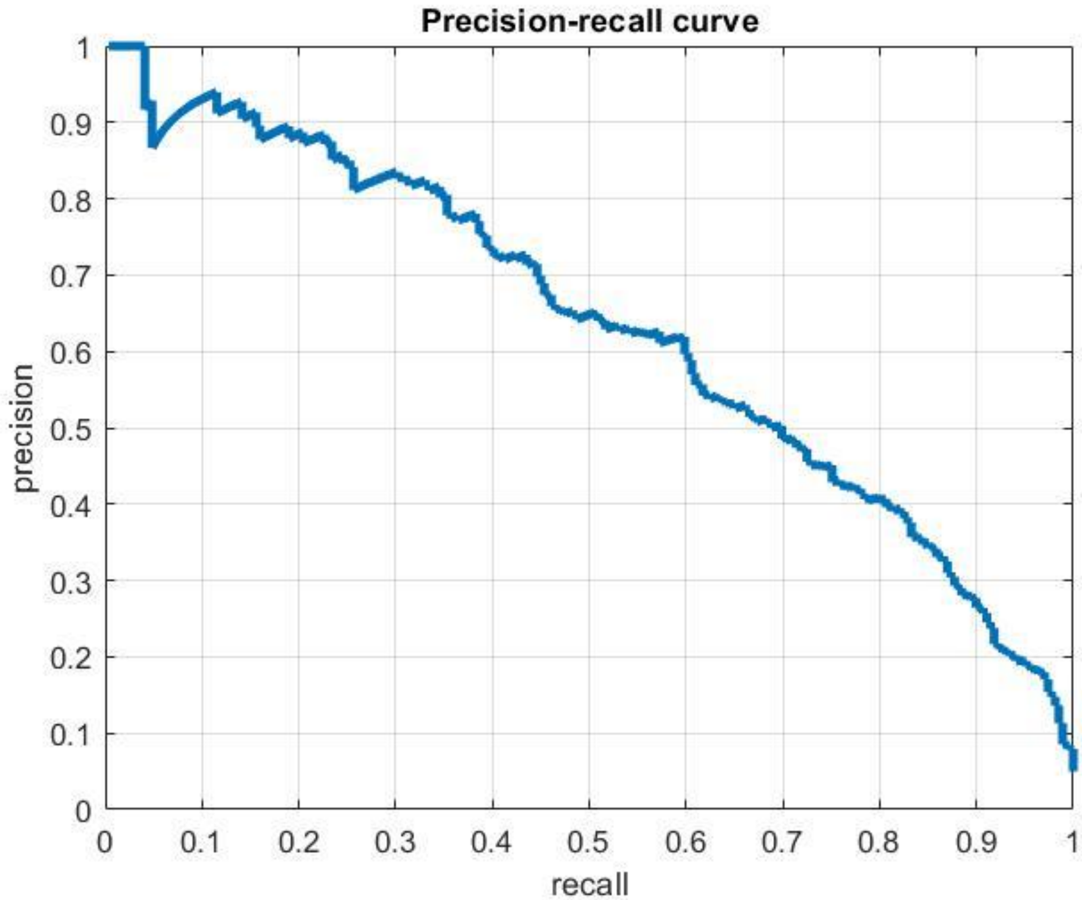
```
Accuracy
    0.9782

Objective Value
  112.1461

Number of Support Vectors
   359

Confusion Matrix
   180     4
     4   179
```

C	**Accuracy**	**Objective Value**	**Number of Support Vectors**
10	0.9782	112.1461	359

Confusion Matrix	Predicted Positive	Predicted Negative
Actual Positive	180	4
Actual Negative	4	179

3. 3.4 Question 3 – SVM for object detection

   3.1. (3.4.1)Use the training data in HW4 Utils.getPosAndRandomNeg() to train an SVM classifier. Use this classifier to generate a result file (use HW4 Utils.genRsltFile) for validation data. Use HW4 Utils.cmpAP to compute the AP and plot the precision recall curve. Submit your AP and precision recall curve (on validation data).

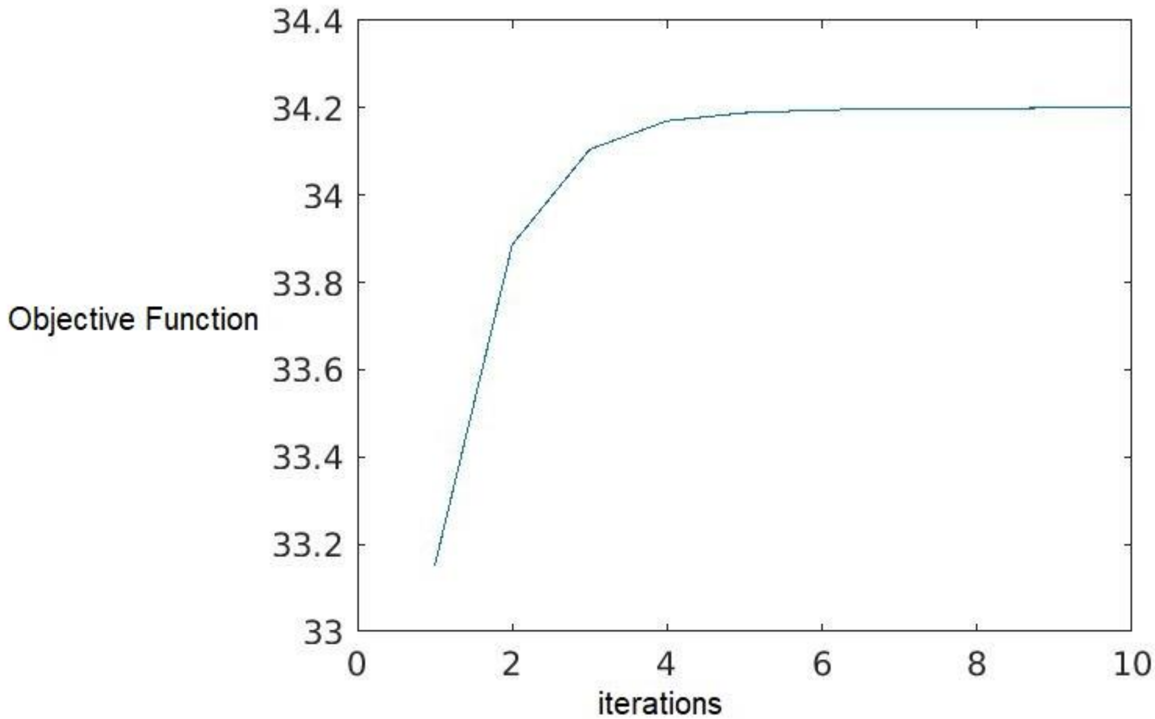


Average precision : 0.635120640788209

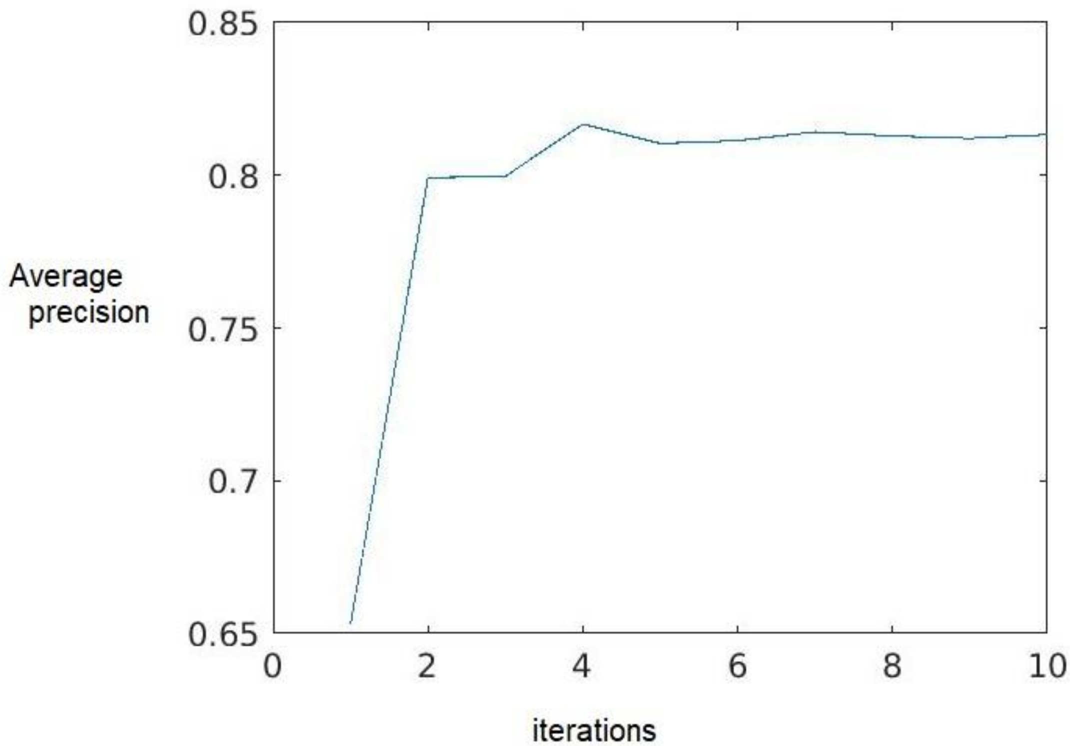
3.2. Hardmining algorithm

3.3. (15 points) Run the negative mining for 10 iterations. Assume your computer is not so powerful and so you cannot add more than 1000 new negative training examples at each iteration. Record the objective values (on train data) and the APs (on validation data) through the iterations. Plot the objective values. Plot the APs.

Objective Function plot:



AP plot:



3.4.4 Submitted result file for test data using the function HW4 Utils.genRsltFile

Average precision on the leaderboard= 0.**7277**