JAGDEESH BASAVARAJU
ASU ID : 1213004713

1. SVM

1.1 Hard-Margin SVM

Lagranges multiplier used to max ~~sep~~ separation,

$$L(w,\alpha) = \frac{1}{2}\vec{w}.\vec{w} - \sum_j \alpha_j \left[(\vec{w}.\vec{x}_j + b)y_j - 1\right]$$

$$\alpha_j \geq 0$$

This is a dual problem.

We take $\frac{\partial L}{\partial w} = 0$ to find extremum.

$$\therefore \frac{\partial L}{\partial w} = \vec{w} - \sum \alpha_i y_i x_i = 0$$

$$\therefore \vec{w}^* = \sum \alpha_i y_i x_i \quad \text{\& hence } b^* = y_K - \vec{w}\vec{x}_K$$

$$\frac{\partial L}{\partial b} = -\sum_i \alpha_i y_i = 0$$

Plug it back to L

$$L = \max_\alpha \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j x_i . x_j$$

Two support vector, hence

$$w^* = \alpha_j x_j - \alpha_K x_K$$

$$b^* = y_K - w^* x_K \qquad b^* = y_j - w^* x_j$$

~~@~~ $y_j = 1$ & $y_K = -1$ is from known ~~sta~~ assumption

$$\therefore b^* = 1 - w^* x_K \qquad b^* = -1 - w^* x_K$$

Hence

$$\underset{\alpha}{\text{Max}} \ (\alpha_j + \alpha_K) - \frac{1}{2}\left[x_j^2\alpha_j^2 + \alpha_K^2 x_K^2 - 2\alpha_j\alpha_K x_j \cdot x_K\right]$$

But $\sum_i \alpha_i y_i = 0$ from $\frac{\partial L}{\partial b}$

Hence $\alpha_j = \alpha_K = \alpha$

$$\therefore \text{Max} \ 2\alpha - \frac{1}{2}\alpha^2 [x_j - x_K]^2 \quad \text{— ①}$$

W.K.T $\underset{w,b}{\arg\max} \ d = d^+ - d^- = \underbrace{\frac{W^T}{\|W\|}}_{\text{Unit vector}} [x_j - x_K]$

$$\therefore d^+ - d^- = [x_j - x_K] \quad \text{— ②}$$

From ① & ②,

$$\underset{\alpha}{\text{Max}} \ 2\alpha - \frac{1}{2}\alpha^2 [d^+ - d^-]^2$$

where constraints are
$\alpha > 0$
$w^* = \alpha(x_j - x_K)$
$b^* = 1 - w^* x_K$
or
$b^* = -1 - w^* x_K$

1.2 Soft Margin SVM

a) ~~Since~~ In the situation where the dataset cannot be linearly separated because of a few points, we allow some amount of error in training as penalty C with slack ~~error~~ ϵ_i for each of the misclassified example. This is called Soft-Margin SVM. When $\epsilon_i = 0$ for all i, then its nothing but hard-margin SVM

Soft margin SVM is given by

$$\min_{w,b} \; w.w + C \sum_j \varepsilon_j$$

$$(w.x_j + b)\, y_j \geq 1 - \varepsilon_j \; , \; \forall j$$

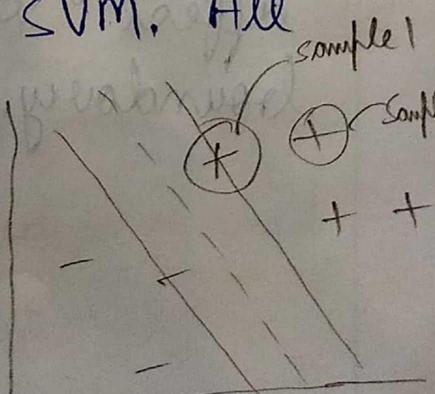
$$\varepsilon_j \geq 0 \;\; \forall j$$

b) The 3 types of SVM examples will be when

i) $\varepsilon_j = 0$

This is nothing but hard margin SVM. All samples are correctly classified.

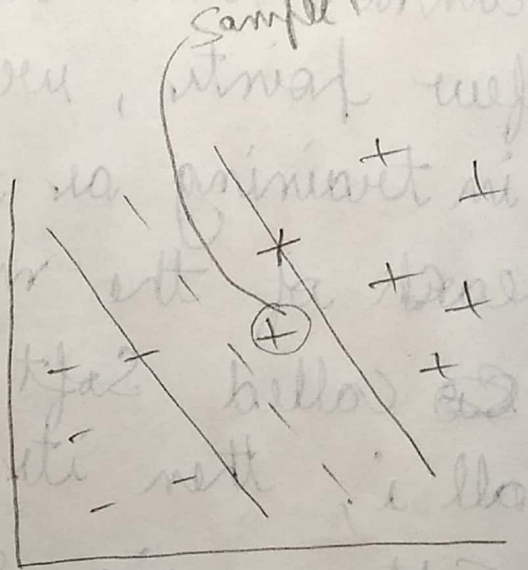
If sample 1 is removed, then decision boundary changes as its on the support vector.



But if sample 2 is removed, then the decision boundary won't change as it's not on the support vector.

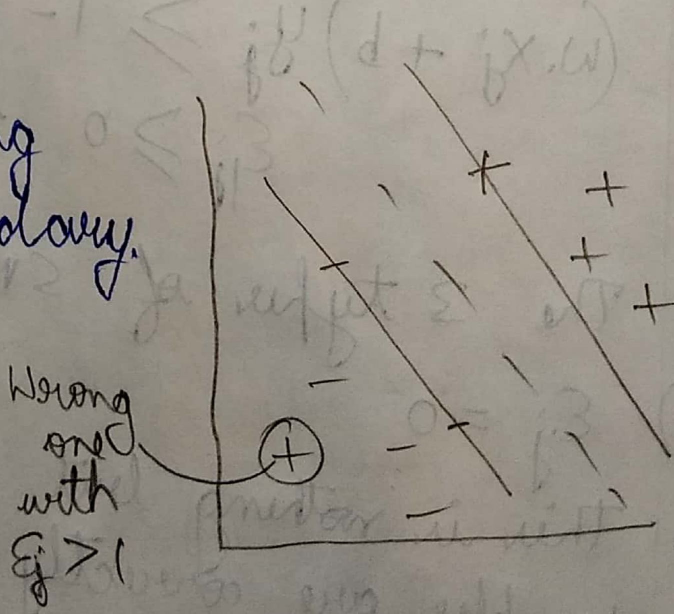
ii) $0 < \varepsilon_j \leq 1$

This is a soft margin SVM where we allow some slack.
Here Sample 1 is classified properly. Removing sample 1 won't have any effect on decision boundary.



iii) $\varepsilon_j > 1$

Sample will be on wrong side of decision boundary.
Removing it wont affect decision boundary.



2 AdaBoost

2.1 If weak classifier has 50% accuracy, then error $\epsilon = 0.5$. Confidence $\alpha = \frac{1}{2} \ln\left(\frac{1-\epsilon}{\epsilon}\right) = \frac{1}{2}\ln(1)$

$$\alpha = 0$$

So confidence will be 0.
The weights of sample also won't change even if we carry out any number of iterations.
$H_{final} = sign\left(\sum_{t=1} \alpha_t h_t\right) = sign(0)$ which has no significance and hence cannot infer anything from it. So we will stop the iteration.

2.2 If accuracy is less than 50%, let's say 45%, then error, $\epsilon = 0.55$ and hence
$\alpha_t = \frac{1}{2} \ln\left(\frac{1-0.55}{0.55}\right) = -0.1003$. So we have to do the opposite of what each classifier says
$H_{final} = sign(\alpha_t h_t(x))$. If $h_t(x)$ predicts $+^{ve}$, $\sum \alpha_t$ will be $-ve$, hence h_{final} will predict $-^{ve}$
When $h_t(x)$ predicts $-ve$ & α_t will $-ve$, hence H_{final} will predict $+^{ve}$. So ~~wet~~ we will reverse the predictions of weak classifier.

2.3

x	0	1	2	3	4	5	6	7	8	9
y	1	1	1	-1	-1	-1	1	1	1	-1

Consider $\theta = 2.5$

$$C(x) = \begin{cases} +1 & x < 2.5 \\ -1 & x \geq 2.5 \end{cases}$$

Correctly classified, $x = \{0, 1, 2, 3, 4, 5, 9\}$

Incorrectly classified, $x = \{6, 7, 8\}$

$D_1(i) = \frac{1}{10} \quad \forall\, i = 0, 1, 2, \ldots, 9$

Error, $\epsilon_t = \frac{1}{\sum D_t(i)} \sum_i D_t(i)\, \delta(h_t(x_i) \neq y_i)$

$\therefore \epsilon_1 = \frac{1}{1}(0.1 + 0.1 + 0.1) = 0.3$

$$\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon}{\epsilon}\right)$$

$$\therefore \alpha_1 = \frac{1}{2} \ln\left(\frac{1-0.3}{0.3}\right) = 0.4236$$

For correctly predicted

$D_{t+1}(i) = \frac{D_t(i)}{Z_t} e^{\{-\alpha_t\}}$

$D_2(i) = \frac{(0.1)}{Z_1} e^{\{-0.4236\}}$

$i = 0, 1, 2, 3, 4, 5, 9$

$= \frac{0.0655}{Z_1}$

For incorrectly predicted

$D_{t+1}(i) = \frac{D_t(i)}{Z_t} e^{\{\alpha_t\}}$

$D_2(i) = \frac{(0.1)}{Z_1} e^{\{0.4236\}}$

$i = 6, 7, 8$

$= \frac{0.1527}{Z_1}$

Scanned by CamScanner

$$Z_t = \underbrace{\sum_i D_t(i)\, e^{\{-\alpha_t\}}}_{\text{Correctly classified}} + \underbrace{\sum_i D_t(i)\, e^{\{\alpha_t\}}}_{\text{Incorrectly classified}}$$

$\therefore Z_1 = 0.0655 \times 7 + 0.1527 \times 3 = 0.9166$

For correctly classified

$$D_2(i) = \frac{0.0655}{Z_1} = \frac{0.0655}{0.9166} = 0.0715$$

for $i = 0, 1, 2, 3, 4, 5, 9$

For incorrectly classified

$$D_2(i) = \frac{0.1527}{Z_1} = \frac{0.1527}{0.9166} = 0.1666$$

for $i = 6, 7, 8$

Hence

x	0	1	2	3	4	5	6	7	8	9
y	1	1	1	-1	-1	-1	1	1	1	-1
D_2	0.0715	0.0715	0.0715	0.0715	0.0715	0.0715	0.1666	0.1666	0.1666	0.0715

2.4 Iteration 2

Consider $\theta = 8.5$, So $C(x) = \begin{cases} +1 & x < 8.5 \\ -1 & x \geq 8.5 \end{cases}$

Correctly classified, $x = \{0, 1, 2, 6, 7, 8, 9\}$

Incorrectly classified, $x = \{3, 4, 5\}$

Error, $\epsilon_2 = \frac{1}{1}[0.0715 + 0.0715 + 0.0715] = 0.2145$

$$\alpha_2 = \frac{1}{2}\ln\left(\frac{1-\epsilon_2}{\epsilon_2}\right) = \frac{1}{2}\ln\left(\frac{1-0.2145}{0.2145}\right) = 0.649$$

For correctly classified

$$D_3(0) = \frac{(0.0715)\, e^{\{-0.649\}}}{Z_2} = \frac{0.0374}{Z_2} = \frac{0.0374}{0.8213} = 0.0455$$

For incorrectly classified

$$D_3(3) = \frac{(0.0715)\, e^{\{0.649\}}}{Z_2} = \frac{0.1368}{Z_2} = 0.1666$$

$D_3(0) = D_3(1) = D_3(2) = D_3(9)$ $\qquad$ $D_3(3) = D_3(4) = D_3(5)$

$$D_3(6) = \frac{(0.1666)\, e^{\{-0.649\}}}{Z_2} = \frac{0.0871}{Z_2} = 0.1061$$

$$Z_2 = 0.0374 \times 4 + 0.0871 \times 3 + 0.1368 \times 3 = 0.8213$$

$D_3(6) = D_3(7) = D_3(8)$

$\therefore$

x	0	1	2	3	4	5	6	7	8	9
y	1	1	1	-1	-1	-1	1	1	1	-1
D_3	0.0455	0.0455	0.0455	0.1666	0.1666	0.1666	0.1061	0.1061	0.1061	0.0455

Iteration 3 :- We choose $\theta = 5.5$ & consider the below classifier which results in least error.

$$C(x) = \begin{cases} -1 & x < 5.5 \\ +1 & x \geq 5.5 \end{cases}$$

Correctly classified, $x = \{3, 4, 5, 6, 7, 8\}$

Incorrectly classified, $x = \{0, 1, 2, 9\}$

Error $\epsilon_3 = \frac{1}{1}(0.0455 + 0.0455 + 0.0455 + 0.0455)$

$= 0.182$

$$\alpha_3 = \frac{1}{2} \ln\left(\frac{1-0.182}{0.182}\right) = 0.7514$$

For correctly classified

$$D_4(3) = \frac{(0.1666)\, e^{\{-0.7514\}}}{Z_3}$$

$$= \frac{0.0786}{Z_3} = 0.1018$$

$D_4(3) = D_4(4) = D_4(5)$

$$D_4(6) = \frac{(0.1061)\, e^{\{-0.7514\}}}{Z_3}$$

$$= \frac{0.05}{Z_3} = 0.0648$$

$D_4(6) = D_4(7) = D_4(8)$

For incorrectly classified

$$D_4(0) = \frac{(0.0455)\, e^{\{0.7514\}}}{Z_3}$$

$$= \frac{0.0965}{Z_3} = 0.125$$

$$Z_3 = 0.0786 \times 3 + 0.05 \times 3 + 0.0965 \times 4$$

$$= 0.7718$$

x	0	1	2	3	4	5	6	7	8	9
y	1	1	1	-1	-1	-1	1	1	1	-1
D_4	0.125	0.125	0.125	0.1018	0.1018	0.1018	0.0648	0.0648	0.0648	0.125

2.5 Iteration 4

Consider $\theta = 2.5$ & consider the below classifier to get minimum error.

$$c(x) = \begin{cases} +1 & x < 2.5 \\ -1 & x \geq 2.5 \end{cases}$$

Correctly classified, $x = \{0, 1, 2, 3, 4, 5, 9\}$

Incorrectly classified, $x = \{6, 7, 8\}$

Error, $\epsilon_4 = \frac{1}{1}(0.0648 + 0.0648 + 0.0648) = 0.1944$

Here we can see that, this classifier is the same as $C_1(x)$. & error is more than previous iteration. We can stop here.

3. K-Nearest Neighbor Classifier

3.1 Lazy Classifier

a. When a new training example becomes available, among SVM, Naive Bayes and KNN, which classifier(s) have to be re-trained from scratch?

SVM has to be re-trained from scratch. For KNN, just add the new data to the training set and then it will be available for prediction. Nothing else has to be done. For Naïve Bayes also, just the count of data points will vary and accordingly probability values can be adjusted and hence it can be updated easily. But for SVM, the new data might change the support vectors entirely and hence has to re-trained from scratch.

b. When a new test example becomes available, among SVM, Naive Bayes and KNN, which classifier needs the most computation to infer the class label for this example, and what is the time complexity for this inference, assuming that we have n training examples, and the number of features is significantly smaller than n?

KNN

In KNN, firstly we need to calculate distance from the new test sample to all of the n training samples. Since number of features are negligible, it will take O(1) time to calculate distance from the test sample to 1 training sample. Therefore O(n) time is required to calculate distance from test sample to n training samples.

Now to select k closest points from the sample, need O(n Log k), assuming max-heap is used to select k closest points.

Selecting label based on majority vote will take O(k).

Hence complexity will be O(n)+O(n Log k)+O(k). If k is negligible compared to n, then O(n).

3.2 Implementation of KNN Classifier

a. Pseudocode

1. Download 'mnist_train.csv' and 'mnist_test.csv' files from the site mentioned.
2. Load the first 6000 samples from training set to X_train (Samples) and y_train(Labels).
3. Load the last 1000 samples from test set to X_test (Samples) and y_test(Labels).
4. Calculate Euclidean Distance from each of the test sample to all the training samples and store it in a matrix of 1000*6000 dimension.
5. Predict label for the test set using the distance matrix using KNN Classifier algorithm with different values of k and calculate error.
6. Plot the graph of error vs the value of k.

```
Function calculate_distance_matrix
        For i=0 to NumberOfTestSamples-1
                difference = X_test^i – X_train
                squared = difference^2
                summed = ∑_j (squared_j)
                squareRooted = √ summed
                distance_matrix[i] = squareRooted
        return distance_matrix

Function predict
        For i=0 to NumberOfTestSamples-1
                distance_from_i = distance_matrix[i]
```

```
            sort(distance_from_i)
            select k closest points from distance_from_i
            obtain classes of those k points from y_train
            y_pred[i] = majority label among k values
        accuracy = (# y_pred == y_test) / (# y_pred)
        error = 1-accuracy
        return error
```

b. Curve of Error vs Value of K