Assignment 5 - Machine Learning *

Anirudh Narasimhamurthy(u0941400)

April 13, 2015

1 Kernel Perceptron

1.1

We are asked to derive an expression for w in the Kernel Perceptron setting.
We know that in normal perceptron, whenever we make a mistake on an example, we update the weight vector as follows:

$$w \leftarrow w + ry(x)$$

In our case the feature representation is given by $\phi(x)$. Hence the perceptron update would be

$$w \leftarrow w + ry\phi(x).$$

Since the inital weight vector is zero and and the learning rate can be assumed as 1, the update is linear combination of the training examples. If we made 'm' mistakes while predicting, then w would be updated for all those mistakes. At the end of this, there would be m mistakes made. Hence w is now effectively

$$\boxed{w = \sum_{(x_i, y_i) \in M} y_i \, \phi(x_i)}$$

$M = \{(x_i, y_i)\}$ is the set of examples on which the learning algorithm made mistakes. And size of M is $|m|$

1.2

Classification step for perceptron is given by :

$$y = sgn\left(w^T \phi(x)\right)$$

*CS 6350 ; Spring 2015

In the given problem, if we were to replace the weight vector which we obtianed in 1.1 in the expression we would get:

$y = sgn\left(w^T \phi(x)\right)$
$y = sgn\left(\sum_{(x_i,y_i)\in M} y_i\, \phi(x_i)^T \phi(x)\right)$
$y = sgn\left(\sum_{(x_i,y_i)\in M} y_i [\phi(x_i)^T \phi(x)]\right)$

From the definition of Mercer kernel function we know that $K(u,v) = \phi(u)^T \phi(v)$. Therefore using the expression in our prediction expression, we will get

$$\boxed{y = sgn\left(\sum_{(x_i,y_i)\in M} y_i K(x_i, x)\right)}$$

The perceptron algorithm always chooses weights which are a linear combination of feature vectors. Hence the prediction function can be effectively wriiten as

$$\boxed{y = sgn\left(\sum_{(x_i,y_i)\in M} s_i K(x_i, x)\right)}$$

where $s_i = y_i$ if we make an error in predicting an example and $s_i = 0$ otherwise.

1.3

Algorithm 1 Pseudo code for Kernel Perceptron Algorithm

Input: Sequence of training examples $(x_1, y_1), (x_2, y_2), (x_3, y_3)...$ where all $x_i \in R^n, y_i \in -1, 1$

1: Initialize s=0
2: Repeat for each training example (x_i, y_i)
3: Predict y'= $sgn\left(\sum_{(x_i,y_i)\in M} s_i K(x_i, x)\right)$
4: if $(y_i \neq y')$
5: $s_i = s_i + y_i$
6: return s_i after seeing all examples

Here s_i indicates the number of times a mistake has been made with respect to x_i and this is is sync with the hint provided which asks us to store the list of examples which the algorithm made a mistake on, rather than storing the weight vector. Finally at the end of this algorithm we get s_i

2 Naïve Bayes

2.1

For the Boolean fiunction mentioned in the question, the prediction is given by :

$$\boxed{sgn(\frac{1}{3}\sum_{i=1}^{7} x_i - 1)}$$

That is whenever the feature vector contains three or more 1's the sign will be positive or the label would be '+'. Whenever we have less than three 1's the label should be '-'

Clearly the prediction function is in linear expression of x_i

Hence the function $f_{TH(3,7)}$ has a linear decision surface over the 7 dimensional Boolean cube.

2.2

Hypothesis produced by Naïve Bayes

Basically the $f_{TH(3,7)}$ produces + or positive label whenever there are more than 3 1's in the feature vector.

Let us assume k to be number of 1's in our feature vector.

For any input x we know that the prediction given by Naïve Bayes classifier is :

$argmax_y P(X = x|Y = y)P(Y = y)$

'y' in our case can either be 0 or 1. The respective probability is given by :

$P(Y = 1) = \frac{\binom{7}{3}+\binom{7}{4}+\binom{7}{5}+\binom{7}{6}+\binom{7}{7}}{2^7} = \frac{99}{128}$

i.e you will get a 1 when there are 3 or more 1's.

$P(Y = 0) = \frac{\binom{7}{0}+\binom{7}{1}+\binom{7}{2}}{2^7} = \frac{29}{128}$

Na$\acute{i}$ve Bayes classifier predicts y=1 if
$P(Y = 1)\pi P(X_i|Y) > P(Y = 0)\pi P(X_i|Y)$

In order to find them out, we would need to compute the following probabilities. All the following probabilities are computed under the assumption of conditional independence.

$P(X_i = 1|Y = 1) = \frac{\binom{6}{2}+\binom{6}{3}+\binom{6}{4}+\binom{6}{5}+\binom{6}{6}}{2^7} = \frac{57/128}{99/128} = \frac{57}{99}$

This is basically telling you given that Y=1 and one of thex_i is 1, there are basically 6 remaining positions out of which 2 or more can be equal to 1.

$P(X_i = 0|Y = 1) = \frac{\binom{6}{3}+\binom{6}{4}+\binom{6}{5}+\binom{6}{6}}{2^7} = \frac{42/128}{99/128} = \frac{42}{99}$

This case is again the other side of the previous case.

$P(X_i = 1|Y = 0) = \frac{\binom{6}{0}+\binom{6}{1}}{2^7} = \frac{42/128}{29/128} = \frac{7}{29}$

$P(X_i = 0|Y = 0) = \frac{\binom{6}{0}+\binom{6}{1}++\binom{6}{12}}{2^7} = \frac{22/128}{29/128} = \frac{22}{29}$

Based on our assumption that we have k 1's, the x_i corresponding to ones would be represented by k's and x_i representing 0's would be represented by (7-k.) We can now write the analytical equations as

$P(Y=1).\pi P(X_i|Y=1)$ as :
$\frac{99}{128} \cdot \frac{42}{99}^{7-k} \cdot \frac{57}{99}^{k}$

and

$P(Y=0).\pi P(X_i|Y=0)$ as :
$\frac{29}{128} \cdot \frac{22}{29}^{7-k} \cdot \frac{7}{29}^{k}$

The sign will be 1 if

$\frac{99}{128} \cdot \frac{42}{99}^{7-k} \cdot \frac{57}{99}^{k}$ ¿ $P(Y=0).\pi P(X_i|Y=0)$ as :
$\frac{29}{128} \cdot \frac{22}{29}^{7-k} \cdot \frac{7}{29}^{k}$

To solve this computationally simpler, we can take log on both sides and solve for k. Taking log on both sides, we get:

$log(99/128) + (7-k)log(42/99) + klog(57/99) \geq log(29/128) + (7-k)log(22/29) + klog(7/29)$

Taking the constant terms on one side and the k terms on the other end we have:

-2.7217 + 0.1322k $\geq$ -1.483 -0.4982k
-1.2387 + 0.6304k $\geq$ 0

This implies k $\geq \frac{1.2387}{0.6304}$
$\boxed{k \geq 1.964}$

2.3

The hypothesis produced by the Naive Bayes implies that it will predict 1 for an example x even if the example x contained just 2 1's in its vector. But the actual function specified by the Boolean function in the problem states that label would be 1 only when 3 or more 1's are present in the feature vector.

Hence the hypothesis produced by Naïve Bayes does not represent the original function

2.4

The Naïve Bayes assumptions are not satisfied by $f_{TH(3,7)}$ Thus the assumptions of conditional independence of the feature vectors did not produce the correct function specified the Boolean function and hence the assumption of conditional independence which we took for solving it via Naïve Bayes does not hold true.

Hence the Naïve Bayes assumptions are not satisfied by $f_{TH(3,7)}$

3 Ensembles of decision trees

3.1 Implementing SVM using SGD as training algorithm

- SVM was implemented using Stochastic Gradient Descent as training algorithm. The update to w, which I have used in my python code corresponds to the update provided in the assignment and not the one in the class slides, as I felt this was relevant for the given problem.

  $w = w - r_t \nabla E(w, x_i, y_i)$

- The gradient definition was also taken up from the notes provided in the experiment section

$$\nabla E(w, x, y) = \begin{cases} w - Cyx & \text{if } yw^T x \leq 1 \\ w & \text{otherwise} \end{cases}$$

- The hyper parameters C and ρ_0 were determined after performing cross validation.

- The experiments were carried out by running it for several epochs.

- To predict the accuracy the same expression which was used in perceptron was used to check if a mistake has been made. That is if $y_i.w^T.x_i \leq 0$. If so the mistake counter was updated and the accuracy at the end was correctly calculated using the updated value of mistakes.

3.2 Cross validation Accuracy

- The experiments were performed by running a 10-fold cross validation. The training data set had 220 entries and hence 10 sets each comprising of 22 entries was formed and each one of those 10 sets was used as the held out set and the accuracy of the cross validation was tested by running the weight vector obtained on these examples.

- For each of the held-out set combination, a total of 10 epochs were run and care was taken to make sure data was shuffled during each epoch.

- The average of the accuracy returned by the 10 fold cross validation was stored in a separate list.

- To determine the best combination of hyper-parameters, I had an outermost loop in the program which taking different combinations of C and ρ_0.

- I had run my experiments for the 20 possible combinations provided in the assignment, i.e $\rho_0 \in \{0.001, 0.01, 0.1, 1\}$ and $C \in \{0.1, 1, 10, 100, 1000\}$

- The best results for accuracy obtained via cross validation are tabulated below:

ρ_0	**C**	**Cross-validation Accuracy**
0.001	100	94.5454545455 %
0.01	10	91.8181818182 %
0.1	1	91.8181818182 %
1	0.1	91.8181818182 %
0.001	10	82.2727272727 %

Table 1: Cross validation accuracy for different hyper paramters for SGD for SVM

- **The best set of hyper parameters obtained from cross validation was $\rho_0 = 0.001$ and C=100**

- Using the best value of rho_0 and C from the above table, I ran the SVM code with 30 epochs and the performance of the classifier on the test set is shown below:

- **Performance values of classifier for best hyper parameter values obtained via 10-fold cross validation**

  Since shuffling is performed, the accuracy varies during every run and I have reported few of the values. Additionally I have also reported the accuracy on the training data for the new weight vector and hyper parameters chosen.

Fraction of examples correctly predicted on test data	**Test data accuracy**
59/74	79.729%
58/74	78.3783%
57/74	77.027%
56/74	75.675%

Table 2: Performance values for best classifier for SGD for SVM

ID3 Algorithm on decision trees of different depths

- I made use of the base code of ID3 algorithm created for Assignment 1 and added a depth parameter, which makes sure the tree stops growing beyond the specified depth.
- As suggested half the training data was taken randomly everytime and a total of 100 decision trees were constructed. The test examples and the training examples were run on the decision trees and the accuracy were calculated. The mean and variance of the accuracy of these trees on the test set are given below:

Decision tree depth	**Mean**	**Variance**
4	24.999999999999996	284.4594594594596
8	17.459459459459456	93.8889700511322
20	16.945945945945944	22.988312636961286

Table 3: Mean and variance values for decision trees of depths 4, 8 and 20

- The accuracy values were slightly surprising as with greater depth the average accuracy or mean sort of went down. Again it could come down to the features provided in the training data and test set couldn't have been the best ones or close to the good ones which we had used for the badges data in Assignment 1.

SVM over ensemble of decision trees of different depths

- After I constructed the 100 decision trees on the training data, I created the new feature vector which consists of 100 dimensions and value of the i^{th} feature is the value of the i^{th} decision tree. Based on this, I constructed the new training data features.
- I had also similarly constructed the new feature vectors for the test data. Both these processes was done for decision trees of depth 4, 8 and 20.
- I then ran the 10-fold cross validation on the training data containing the new 100 dimensional feature vector for 20 different combinations of ρ_0 and C. The 5-best hyper parameters and the cross validation accuracy for different decision trees are shown below:

Decision trees of depth 4

Best choice of hyper parameters : $rho_0 = 0.001$ **and C=10**

ρ_0	C	Cross-validation Accuracy
0.001	10	58.6363636364%
0.01	1	55.4545454545 %
0.1	0.1	55.4545454545 %
0.1	1	52.7272727273 %
0.001	100	52.7272727273 %

Table 4: Cross validation accuracy for Ensemble decision trees of depth=4

ρ_0	C	Cross-validation Accuracy
0.001	100	65.4545454545%
0.01	10	65.4545454545 %
0.1	1	60.4545454545 %
0.001	10	60.4545454545 %
0.01	1	58.1818181818 %

Table 5: Cross validation accuracy for Ensemble decision trees of depth=8

Decision trees of depth 8

Best choice of hyper parameters : $rho_0 = 0.001$ **and C=100**

Decision trees of depth 20

ρ_0	C	Cross-validation Accuracy
0.001	100	74.0909090909%
0.01	10	74.0909090909 %
0.1	1	74.0909090909 %
1	0.1	74.0909090909 %
0.01	1	68.6363636364 %

Table 6: Cross validation accuracy for Ensemble decision trees of depth=20

Best choice of hyper parameters : $rho_0 = 0.001$ **and C=100**

3.1 Evaluation and Report

Comparison of accuracy for the seven different models for which experiments were performed are tabulated below:

- To rank the following seven models in terms of their performance and cross validation accuracy for obtaining the hyper parameters for our given test and training examples, the order would be as follows:

  SGD for SVM > SGD for SVM over ensemble of decision trees > Accuracy of decision tree

  More specifically SGD for SVM > SGD over ensemble of decison trees of depth=20 > SGD over ensemble of decison trees of depth=8 > SGD over ensemble of decison trees of depth=4 > Accuracy of decison trees of depth=20 > Accuracy of decison trees of depth=8 > Accuracy of decison trees of depth=4

Model	Accuracy	Mean	Variance
SGD for SVM on the original feature set	79.72%	-	-
Decision tree of depth 4	-	24.99	284.45
Decision tree of depth 8	-	17.45	93.88
Decision tree of depth 20	-	16.94	22.98
SVM over ensemeble of decision trees of depth 4	43.24%	-	-
SVM over ensemeble of decision trees of depth 8	48.64 %	-	-
SVM over ensemeble of decision trees of depth 20	58.10 %	-	-

Table 7: Performance Evaluation of different models

- In SGD for SVM over ensemble trees, the accuracy is better as the depth of the decision tree increases. Again since we are doing a random sub sampling and then generating the feature vectors, these might not always reflect in the accuracy values. But this was something which I observed and have reported.
- Accuracy of the stand alone decision trees of different depths 4, 8 and 20 is lower than the SVM accuracy. This can be attributed to the fact that the given training and test data set might be linearly separable. Also since we are not running over ensemble or cross validation, it could be lower. Otherwise, we would have expected the decision trees to give us a better accuracy on the examples provided.
- The lower accuracy of decision trees could also be due to the fact that the features provided might not have encoded the most important feature and there could have been large number of irrelevant features thereby bringing down the accuracy.
- In conclusion the SGD for SVM on the original data set had a much better accuracy than the rest of the models which we experimented with.

  PS: I have provided the commands in shell script named "run.sh" which runs the SVM, Cross Validation, ID3 for constructing decision trees of depths 4, 8 and 20, Cross Validation for the new feature vectors and SGD for SVM for th ensemble decision trees of different depths. I have provided the best found hyper parameters as parameters to the python program and the so the result it produces would be for the best chosen hyper parameters. Also all the programs have randomization involved and running the script multiple times might produce different outputs each time.