Machine Learning Assignment-1

Problem1:

1. (5pts) What are the `2 -norms of a1 , a2 , a3 and a4 ?
2. (5pts) Determine if a4 is a linear combination of the vectors a1; a2 and a3 .
3. (5pts) Let a1 , a2 , and a3 be the bases for a feature space. What is the coordinate of a4 in that feature space?

Answer:

1)

a. $\|a_1\|=\sqrt{1^2+1^2+0^2}=\sqrt{1+1}=\sqrt{2}$

b. $\|a_2\|=\sqrt{(-2)^2+(-1)^2+(-1)^2}=\sqrt{4+1+1}=\sqrt{6}$

c. $\|a_3\|=\sqrt{(3)^2+(-1)^2+(-3)^2}=\sqrt{9+1+9}=\sqrt{19}$

d. $\|a_4\|=\sqrt{(-5)^2+(-4)^2+(-7)^2}=\sqrt{25+16+79}=\sqrt{90}$

2) $\alpha_1 - 2\alpha_2 + \alpha_3 = -5$….(i)

$-\alpha_1 - \alpha_2 - \alpha_3 = -4$…….(ii)

$-\alpha_2 - 3\alpha_3 = -7$……(iii)

Solving the equations (i), (ii) and (iii),

We get values

$\alpha_1 = -9/11$

$\alpha_2 = 41/11$

$\alpha_3 = 12/11$

Thus, a_4 is a linear combination of a_1, a_2, a_3

3) $a_4 = \begin{bmatrix} -\frac{9}{11} \\ \frac{41}{11} \\ \frac{12}{11} \end{bmatrix}$

Thus, these are the co-ordinates of in the feature space with a_1, a_2, a_3.

Problem2:

(Linear Algebra, 15pts) For any given matrix A ∈ Rm_n , B ∈ Rn_p and C ∈ Rp_q , show that (AB)C =
A (BC) using the de_nition of matrix multiplication

Answer:

LHS= AB∈ Rm*n

=(AB)C $\in$Rn*q

RHS=BC$\in$ Rn*q

=a(BC) $\in$ Rm*q

Thus, we were able to determine that the dimensions are the same.

To show if the matrix space is associative, we check $(I,j)^{th}$ entry of (AB)C is equal to $(i,j)^{th}$ entry of A(BC).

PROOF:

$((AB)C)_{i,j} = \sum_{k=1}^{p}(AB)_{ik}\ C_{kj}$

$= \sum_{k=1}^{p}(\sum_{l=1}^{n} A_{il}\, B_{lk})C_{kj}$

$= \sum_{k=1}^{p}(\sum_{l=1}^{n} A_{il}\, B_{lk}\, C_{kj})$

$= \sum_{l=1}^{p}(\sum_{k=1}^{n} A_{il}\, B_{lk}\, C_{kj})$

$= \sum_{l=1}^{n} A_{il}(\sum_{k=1}^{n} B_{lk}\ C_{lk})$

$=\sum_{l=1}^{n} A_{il}(BC)_{lj}$

$=A(BC)_{ij}$

Problem 3:

Report for K Nearest Neighbors

The formula used in the KNN to determine distance between 2 instances:

KNN is the instance-based prediction algorithm:

Euclidian Formula:

$$\mathrm{d}(\mathbf{p}, \mathbf{q}) = \mathrm{d}(\mathbf{q}, \mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$

$$= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}.$$

KNN Classification:

The program calculates the Euclidian distance between the test instance all the training instances. Then, picks the *k* nearest neighbors. Then depending up on the majority of the class of these neighbors, a class is assigned to the test instance.

Eg. If a particular instance has the neighbors-

{[1,1,1,'a'],[2,2,2,'a'][3,3,3,'b']}

Then the test data will have class 'a', as majority of its neighbors have the class 'a'.

KNN Regression:
The program predicts the actual value of a particular attribute. It uses the Euclidian distances between the test instance and the training to data to find out the *k* nearest neighbors. Then the program calculates to average of that attribute of each of the neighbors. Then this average value is predicted as the value of attribute of test data.
Eg. [1,1,1,'3'][1,2,1,'4'][2,1,1,'4']- neighbors
Test Instance: [1,1,2,'?']
Predicted value: (3+3+4/3) = 3.33

1) The CSV module is imported to handle the data from CSV file. The random module allows the random division of data into testing and training dataset. The math module is for the mathematical operations (eg. pow) and operator module provides the easy access to operator functions.

2) Read_data(): this function reads the data from the csv file then randomly divides the data into training and test dataset. This split happens with 66:34 ratio.

3) Calc_euclidian(): it calculates the Euclidian distance between the instances based on the feature values.

4) myknnClassify(): this function uses calc_euclidian() to calculate the distances and find the 5 closest neighbors. Then these neighbors and their classes are saved in a dictionary.

5) Calc_classvotes(): it takes the dictionary generated at my classify and gets the vote form all the *k* nearest neighbors and assigns the class to the testing instance.

6) Myknnregressor(): Uses the calc_euclidian to calculate the distance between each instance and testing instance. Then uses the k nearest neighbors to calculate the average of the values of the feature that needs the prediction for the test instance. This average is then presented as a predicted value.

Output Format:

```
Train set:  138
Test set:  75
Enter the value for K:4
Enter the value of the attribute you want to predict:3
[[2.0, 1.51761, 13.89, 3.6, 1.36, 72.73, 0.48, 7.83, 0.0, 0.0, 1.0], [1.0,
1.52101, 13.64, 4.49, 1.1, 71.78, 0.06, 8.75, 0.0, 0.0, 1.0], [6.0, 1.5159
6, 12.79, 3.61, 1.62, 72.97, 0.64, 8.07, 0.0, 0.26, 1.0], [7.0, 1.51743, 1
3.3, 3.6, 1.14, 73.09, 0.58, 8.17, 0.0, 0.0, 1.0]]
Regressor AVG: 13.405000000000001
Actual: 13.53
[1.0]
> predicted= 1.0 , actual= 1.0
[[6.0, 1.51596, 12.79, 3.61, 1.62, 72.97, 0.64, 8.07, 0.0, 0.26, 1.0], [2.
0, 1.51761, 13.89, 3.6, 1.36, 72.73, 0.48, 7.83, 0.0, 0.0, 1.0], [7.0, 1.5
```

```
1743, 13.3, 3.6, 1.14, 73.09, 0.58, 8.17, 0.0, 0.0, 1.0], [1.0, 1.52101, 1
3.64, 4.49, 1.1, 71.78, 0.06, 8.75, 0.0, 0.0, 1.0]]
Regressor AVG: 13.405000000000001
Actual: 13.21
[1.0]
> predicted= 1.0 , actual= 1.0
.
.
.
> predicted= 7.0 , actual= 7.0
[[213.0, 1.51651, 14.38, 0.0, 1.94, 73.61, 0.0, 8.48, 1.57, 0.0, 7.0], [20
9.0, 1.5164, 14.37, 0.0, 2.74, 72.85, 0.0, 9.45, 0.54, 0.0, 7.0], [207.0,
1.51645, 14.94, 0.0, 1.87, 73.11, 0.0, 8.67, 1.38, 0.0, 7.0], [206.0, 1.51
732, 14.95, 0.0, 1.8, 72.99, 0.0, 8.61, 1.55, 0.0, 7.0]]
Regressor AVG: 14.66
Actual: 14.36
[7.0]
> predicted= 7.0 , actual= 7.0
Accuracy:  98.66666666666667%
```

The dataset given is the glass dataset, which has 213 instances, 138 of which are used as training instances and 75 are testing instances. The program asks for the user input for k and the feature-index to be predicted with the regressor. Then the program prints the 4 nearest neighbors, the average of the feature to be predicted, the actual value of the feature and the class predicted with myknnclassify() function.

The accuracy depends upon the number of *k*-s we choose. A too small value for *k* will provide less evidence to predict the correct class or the feature value. A too large value will provide widely spread data with possible outliers for that instance, thus affecting the prediction.

Problem 4:

1. (5pts) Variance of f (x) is de_ned by var [f] = E [(f (x) □ E [f (x)])2]. Show that it can be also written
as var [f] = E [f (x)2] □ [f (x)]2 .
2. (5pts) In the lecture note 05 Statistics review, from page 23 to page 26, there is an example describing
Bayes rule. In the same setting, let's assume that you picked an apple with your eyes closed. What is
the probability that the bag where you picked the apple is red?
3. (5pts) Pattern Recognition and Machine Learning, Bishop, Exercise 1.6
4. (5pts) Pattern Recognition and Machine Learning, Bishop, Exercise 1.11

Answer:
1)
$Var[f]=E[(f(x)-E[f(x)])^2]$---------- (Given)

$=E[f(x)-2f(x)e[f(x)]+ E(f(x))^2]$
$=E[f(x)^2]-2E[f(x)]E[f(x)]+ E[f(x)]^2$
$= E[f(x)^2]-2E[f(x)]^2+ E[f(x)]^2$
$= E[f(x)^2]-E[f(x)]^2$

2)
To find: P(R|a)=?
Solution:
P(F=a) = p(F=a|B=r)p(B=r)+p(F=a|B=b)P(B=b)

=(1/4)*(4/10)+(3/4)(6/10)

=(11/20)----------(1)

P(F=a|B=r)=1/4----------(2)

$P(B=r|F=a)= \frac{P(F=r|B=R)\,P(B=r)}{P(F=a)}$

$= \frac{\left(\frac{1}{4}\right) * \left(\frac{4}{10}\right)}{11/20}$

= (1/10) * (20/11)

=2/11

3)To prove: x and y independent variables, then their covariance is zero:
Solution:
Formula of covariance-
$cov[x,y]= E_{x,y}[(x-E(x))(y-E[y])]$

$= E_{x,y}[xy]-E[x]E[y]$

Proof:
When x and y are independent, E[x,y]=E[x]E[y]------(1)
Now, $cov(x,y)= E[(x-\mu_x)(y-\mu_y)]$
Since, x and y are independent $x-\mu_x$ and $y-\mu_y$ are also independent of each other.
Therefore, $cov(xy)= E[(x-\mu_x)]E[(y-\mu_y)]$
$= [E(x)-\mu_x)E(y)-\mu_y)]$
$= 0$
Since, $E[x]= \mu_x$ and $E[y]= \mu_y$

4) By setting the derivatives of the log likelihood function (1.54) with respect to μ and σ 2 equal to zero, verify the results (1.55) and (1.56). (Please find all the equations in Pattern Recognition and Machine Learning, Bishop)

$$\ln p\left(\mathbf{x}|\mu,\sigma^2\right) = -\frac{1}{2\sigma^2}\sum_{n=1}^{N}(x_n-\mu)^2 - \frac{N}{2}\ln\sigma^2 - \frac{N}{2}\ln(2\pi).$$

The first part of the problem say that if we equate the derivative of log likelihood function with respect to μ and σ^2, we will be able to prove the following :

1. $\mu = \frac{1}{N}\sum_{n=1}^{N} x_n$
2. $\sigma^2 = \frac{1}{N}\sum_{n=1}^{N}(x_n-\mu)^2$

First, lets differentiate the log likelihood function with respect to μ, we get

$$\frac{d}{d\mu}\left(lnp(x/\mu,\sigma^2)\right) = \sum_{n=1}^{N}(-2)x_n + 2\mu$$

Now equating this equation to 0, we get

$$N\mu = \sum_{n=1}^{N} x_n$$

$$\mu = \frac{1}{N}\sum_{n=1}^{N} x_n$$

Now, lets differentiate the log likelihood function with respect to σ^2, we get

$$\frac{d}{d\sigma^2}\left(lnp(x/\mu,\sigma^2)\right) = \sum_{n=1}^{N}(x_n-\mu)^2 - N\sigma^2$$

Now equating this equation to 0, we get

$$N\sigma^2 = \sum_{n=1}^{N}(x_n-\mu)^2$$

$$\sigma^2 = \frac{1}{N}\sum_{n=1}^{N}(x_n-\mu)^2$$

Problem 5:

Report for Naïve Bayes Classifier

The basic formula used in Naïve Bayes:

$$P(H|E) = \frac{P(E|H) * P(H)}{P(E)}$$

Where,
P(H)- probability of hypothesis H being true
P(E)- probability of evidence
P(E|H)-probability of evidence given the hypothesis is true.
P(H|E)- probability of hypothesis given the evidence.

The Mushroom data set has multiple features on which the probability of each mushroom being edible or poisonous depends up on. Explaining in the terms of above formula, each feature/attribute is the evidence and the poisonous or edible is the class. As the probability of each mushroom belonging to a class(poisonous/edible) depends upon each of the feature, above formula changes to-

P(H|Multiple Evidences) = P(E1| H)* P(E2|H) ……*P(En|H) * P(H) / P(Multiple Evidences)

Code implementation:

The csv package allows the '.csv' files to be accessible to the python code.

1) Load_datasets () function: This function retrieves the data from CSV file and loads it into 2 separate arrays.

1. Training array
2. Testing array

The 1st 4000 records are placed in the training array and the rest are put into test array.
The instance having the '?' value for the 11th feature is dropped.

2)Attributes() functions: This function loads the attributes with the values they might take are loaded into attribute list.

3) Attribute Lists() function: The attributes of both the edible and poisonous mushrooms are loaded in the attribute_edible and attribute_poisonous lists respectively. Thus making the frequency count easier when needed in the formula.

4) Naïve Bayes() function: naïve bayes cprobability calculation is done in this function for both the probability of mushroom being poisonous and probability of mushroom being edible given the features. Then both the probabilities are compared and the test data is assigned the higher probability.
Eg. If a mushroom has p(edible| features)=0.67 and p(poisonous|features)=0.44 then the mushroom is assigned to the edible class.

5) Main() function: Function calls the above functions and prints the accuracy according to the output by comparing the predicted and actual class. And gives the accuracy in the percentage.

Output format:

```
training dataset size: 4000
testing dataset size: 1644
total data: 5644
Tables uplaoded successfully!!!
total edible in training data: 3309
total poisonous in training data: 691
actual: p classified: p
actual: e classified: e
actual: e classified: e
actual: p classified: p
actual: p classified: p
actual: e classified: e
actual: p classified: p
actual: e classified: e
actual: e classified: e
actual: e classified: e
actual: p classified: p
actual: e classified: e
actual: p classified: p
actual: p classified: p
actual: p classified: p
actual: p classified: p
actual: p classified: p
.
.
.
total edible: 179
total poison: 1465
Percent correct: 80.778589
```

Explanation:

-The original mushroom data set contains 8124 records. There are some instances that consisting '?' in the feature 11, these instances are removed and the total data size becomes 5644. The out of these 4000 are used as an training data and rest are used as testing data.
The data is divided into ratio 7:3 ratio.

-the data set explanation file explains that the main 6 features that determine if the mushroom are poisonous are:

Disjunctive rules for poisonous mushrooms, from most general
to most specific:

P_1) odor=NOT(almond.OR.anise.OR.none)
120 poisonous cases missed, 98.52% accuracy

P_2) spore-print-color=green
48 cases missed, 99.41% accuracy

P_3) odor=none.AND.stalk-surface-below-ring=scaly.AND.
(stalk-color-above-ring=NOT.brown)
8 cases missed, 99.90% accuracy

P_4) habitat=leaves.AND.cap-color=white
100% accuracy

Since, some these values only occur in the test dataset, the accuracy degrades. If the train data is randomly selected, then the accuracy factor may increase.

Eg. The feature Habitat has the value 'Leaves' only in the last few of the records. Thus, making its probability very low, and thus affecting the prediction of the mushroom class.

Reference:
[1] http://archive.ics.uci.edu/ml/machine-learning-databases/mushroom/agaricus-lepiota.names
[2]https://machinelearningmastery.com/naive-bayes-classifier-scratch-python/
[3] https://machinelearningmastery.com/tutorial-to-implement-k-nearest-neighbors-in-python-from-scratch/
[4] https://www.youtube.com/watch?v=G275SvYjg2o
[5] https://www.youtube.com/watch?v=XkU09vE56Sg&t=3s
[6] https://www.youtube.com/watch?v=8pTlCJX59Do&t=578s
[7] Pattern Recognition and Machine Learning- Christopher M. Bishop