CS550-TheoryHomework-2

Name-Manisha Chawla

ID No. 12210370

References:

1. Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems
2. https://www.geeksforgeeks.org/bagging-vs-boosting-in-machine-learning/
3. https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052

3. A car insurance company is building a risk assessment prediction system based on the age of the driver and the type of car. Can you help them by building a decision tree using information gain as the split point evaluation measure?
Here is the sample data:

Age of Driver	Car Type	Risk
25	Sports	L
20	Vintage	H
25	Sports	L
45	SUV	H
20	Sports	H
25	SUV	H

a. (1 mark) Entropy of the dataset is:
Hint: Entropy = H(S) = − $\sum_{c \in C} p_c log_2 p_c$

Answer:-

p_c = is the probability of getting the c value when randomly selecting one from the set C.
if we split the dataset according to a attribute Risk..

C={L,H}
From the sample data set:
Total No. of instances(SET S) = 6

Total No. of instances with Risk L= 2
Total No. of instances with Risk H = 4

$p_{Risk=L}$ = $\frac{2}{6}$=$\frac{1}{3}$

$p_{Risk=H}$ = $\frac{4}{6}$=$\frac{2}{3}$

Entropy of the dataset H(S) $= -[\frac{1}{3}log_2\frac{1}{3} + \frac{2}{3}log_2\frac{2}{3}]$

Entropy of the dataset H(S)$= -(-0.918295)$

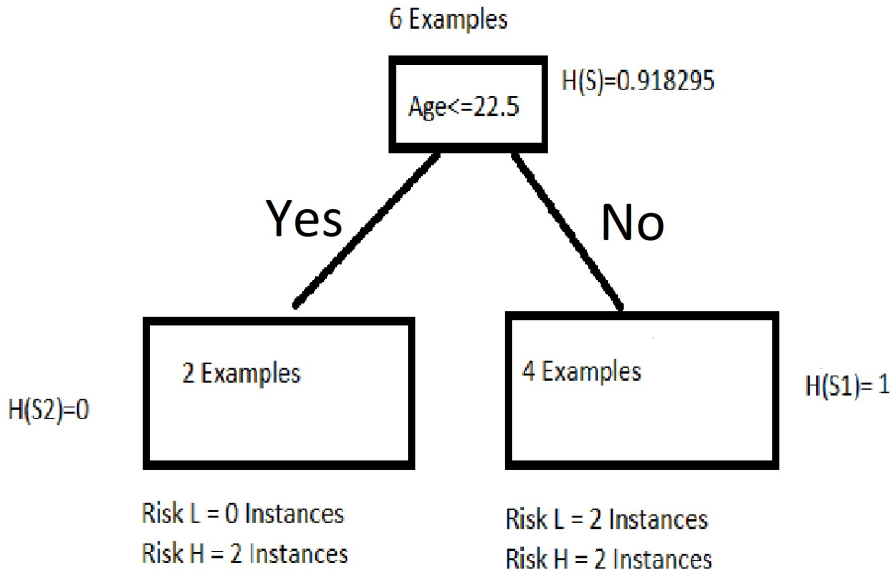
Entropy of the dataset H(S)$= 0.918295$

This makes sense – it's a good mix of both classes; so the entropy should be close to 1.

b. (1 mark) Calculate the information gain if we split the dataset by the following rule: Age<=22.5
Hint:$IG = H(S) - \frac{|S1|}{S}H(S1) - \frac{|S2|}{S}H(S2)$

Answer:-
The data set that goes down each branch of the tree has its own entropy
value. We can calculate for each possible attribute its expected entropy. This is the degree to which the entropy would change if branch on this attribute. You add the entropies of the two children, weighted by the proportion of examples from the parent node that ended up at that child.

if we split the dataset by the following rule: Age<=22.5



S2=Left child where(Age<=22.5 is Yes) has 2 Instances

S1=Right child where(Age<=22.5 is No) has 4 Instances

hence |S1|=4, |S2|=2

Entropy of the dataset H(S2) $= -[\frac{0}{2}log_2\frac{0}{2} + \frac{2}{2}log_2\frac{2}{2}] = 0$

Entropy of the dataset H(S1) $= -[\frac{2}{4}log_2\frac{2}{4} + \frac{2}{4}log_2\frac{2}{4}] = 1$

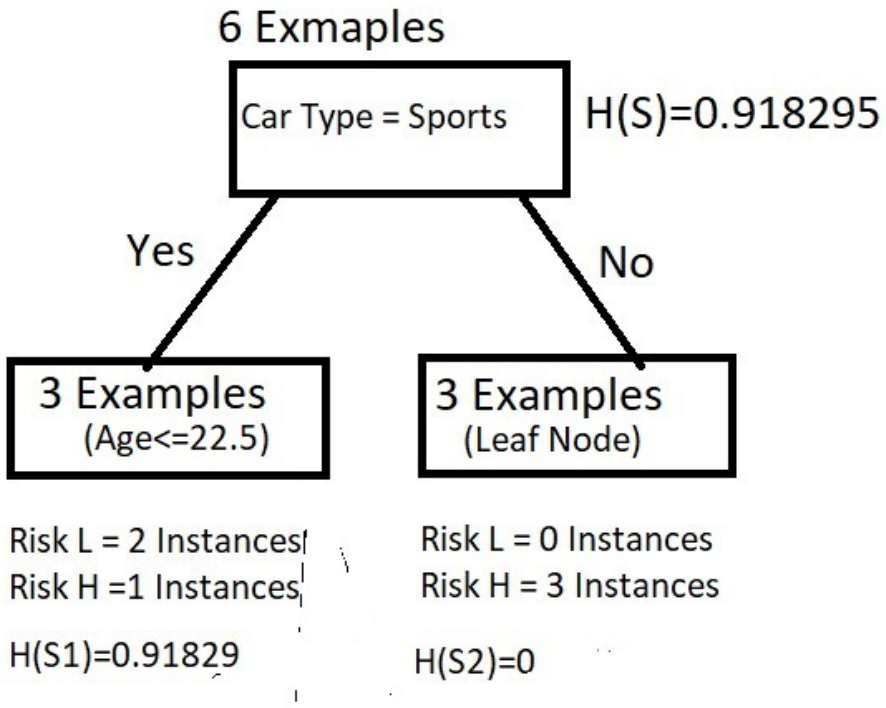
G(S,Age<=22.5) = $IG = H(S) - \frac{|S1|}{S}H(S2) - \frac{|S2|}{S}H(S1)$

$IG_1 = 0.918295 - \frac{2}{6}X0 - \frac{4}{6}X1 = 0.918295 - 0.66667 = 0.251628$

So, we have gained 0.251628 bits of information about the dataset by
choosing 'Age<=22.5' as the first branch of our decision tree.

c. (1 mark) Calculate the information gain if we split the dataset by the following rule:
Car Type = Sports
Hint: If the Car Type is not Sports, it can be either SUV or Vintage

Answer:
If we split the dataset by the following rule: Car Type



S1=Left child where(Car Type = Sports) has 3 Instances

S2=Right child where(Car Type = not sports) has 3 Instances

hence |S1|=3, |S2|=3

Entropy of the dataset H(S1) $= -[\frac{1}{3}log_2\frac{1}{3} + \frac{2}{3}log_2\frac{2}{3}] = 0.91829$

Entropy of the dataset H(S2) $= -[\frac{0}{3}log_2\frac{0}{3} + \frac{3}{3}log_2\frac{3}{3}] = 0$

G(S,Car Type=Sport) = $IG = H(S) - \frac{|S1|}{S}H(S1) - \frac{|S2|}{S}H(S2)$

$IG_2 = 0.918295 - \frac{1}{2}X0.91829 - \frac{1}{2}X0 = 0.918295 - 0.459145 = 0.45915$

So, we have gained 0.45915 bits of information about the dataset by choosing ‘Car Type=Sport’ as the first branch of our decision tree.

d. (2 mark) Which of the above two rules should we use for the root-node of the decision tree? Assuming that this is the best choice, can you complete the decision tree?

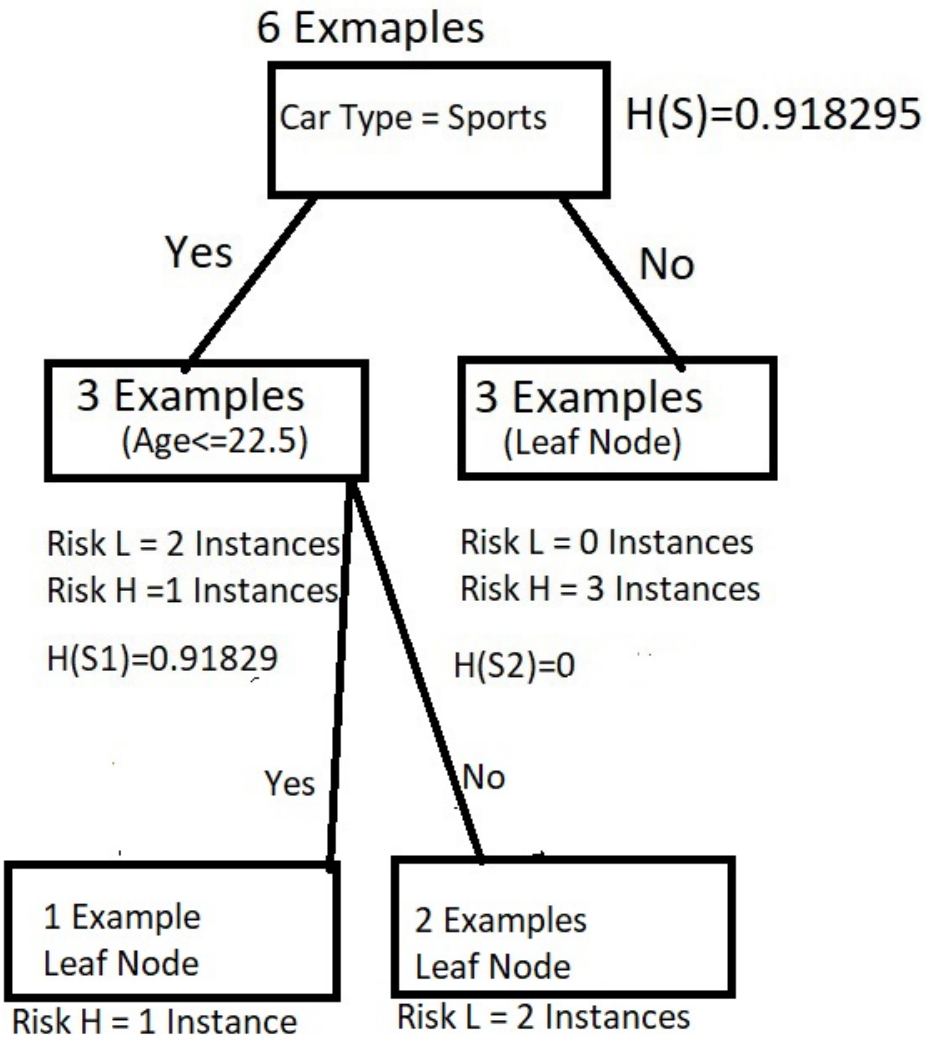
Answer:
From above two answers we have seen that information gain by choosing ‘Age<=22.5’ IG_1 and Car Type=Sport’ IG_2 as the first branch of our decision tree :

$IG_1 < IG_2$
$G(S, Age <= 22.5) < G(S, CarType = Sport)$

As Information gain or IG is a statistical property that measures how well a given attribute separates the training examples according to their target classification. Constructing a decision tree is all about finding an attribute that returns the highest information gain and the smallest entropy.

Information gain is more in case of 'Car Type=Sport' so we should choose that as the first branch of our decision tree.

Decision Tree:



2. (10 marks) (1 mark each) Objective type questions. Answer with a brief justification.

i. Explain the importance of scaling features for training Large Margin Classifiers?

Answer:
Training an Large Margin classifier includes deciding on a decision boundary between classes. This boundary is known to have the maximum distance from the nearest point on each data class.

Feature scaling is mapping the feature values of a dataset into the same range. Feature scaling is crucial for some machine learning algorithms, which consider distances between observations because the distance between two observations differs for non-scaled and scaled cases.

The decision boundary maximizes the distance to the nearest data points from different classes. Hence, the distance between data points affects the decision boundary that Large Margin Classifier chooses.

When the data is not scaled that is the scale of one feature is larger than the other one,the widest possible street/distance will ignore the feature with small scale.

In other words, training an Large Margin Classifier over the scaled and non scaled data leads to the generation of different models.

ii. Explain the effect of changing the value of C from very small to very large in a Soft margin Classifier's objective function?

Minimize: $||W||_2^2 + \sum_{i=1}^{\mathbb{N}} Cz_i$

Subject: $y_i(w^T x_i + b) \leq 1 - z_i \forall i \in \{1, 2, 3.., N\}$

Where z_i are the slacks (errors).

Answer:

$$L = ||W||_2^2 + \sum_{i=1}^{\mathbb{N}} Cz_i$$

The new slack variables ζ_i add flexibility for misclassifications of the model.
In a SVM we are searching for two things:

- A hyperplane with the largest minimum margin,
- and a hyperplane that correctly separates as many instances as possible.

The problem is that we will not always be able to get both things. The C parameter determines the penalty for error.

Here, C is a hyperparameter that decides the trade-off between maximizing the margin and minimizing the mistakes. When C is small, classification mistakes are given less importance(low penalty) and focus is more on maximizing the margin, whereas when C is large(large penalty), the focus is more on avoiding misclassification at the expense of keeping the margin small.

The C parameter tells the SVM optimization how much you want to avoid misclassifying each training example. A very small value of C will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points. When we change C from very small to large values the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly. This may lead to overfitting of the model.

iii.Consider the 2-D array, arr. What is the output of np.sum(arr, axis=1)?

Answer:-

```
import numpy as np
A= np.array([1, 2, 3, 4, 5, 6, 7, 8,9])
arr = arr.reshape(3, 3)
```

print(arr)
[[1 2 3]
[4 5 6]
[7 8 9]]
np.sum(arr, axis=1)
array([6, 15, 24])
numpy.sum(arr, axis, dtype, out) : This function returns the sum of array elements over the specified axis.Here axis = 1 means working along the row.

iv. Which of the following are methods for indexing into a DataFrame?
a. Use the loc and iloc functions
b. Directly index columns similar to a Python dictionary
c. Using slices to retrieve a set of rows
d. All of the above

Answer:
d. All of the above

v. What is an indicator feature?
a. A feature that indicates whether or not the data has been pre-processed
b. A feature that represents categorical data, using 1's and 0's to denote which categories are present
c. An indicator of whether or not a category is quantitative or categorical
d. None of the above

Answer:
b. A feature that represents categorical data, using 1's and 0's to denote which categories are present.

vi. What is the main purpose of standardizing data?
a. It lets us view data in terms of standard deviations from the average case (i.e. mean)
b. It lets us use much smaller data values, which makes computation much quicker
c. It makes it easier to calculate the mean of each column in the data
d. It removes outliers from the dataset
e. All of the above

Answer:
a. It lets us view data in terms of standard deviations from the average case (i.e. mean)

vii. For the given confusion matrix, please calculate accuracy, precision, recall and F1 score.

	Actual	
Predicted	Yes	No
Yes	12	3

	Actual	
No	1	9

Answer:

	Actual	
Predicted	Yes	No
Yes	TP=12	FP=3
No	FN=1	TN=9

TP=True Positive

FP=False Positive

FN= False Negative

TN=True Negative

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} = \frac{12 + 9}{12 + 3 + 1 + 9} = 0.84$$

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{12}{12 + 3} = 0.8$$

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{12}{12 + 1} = 0.9231$$

$$\text{F1 Score} = \frac{TP}{TP + \frac{FN+FP}{2}} = \frac{12}{12 + \frac{1+3}{2}} = 0.8571$$

viii. What is the difference between bagging and boosting? Which technique would be suitable when the data has high variance?

Answer:-

SN	Bagging	Boosting
1	Various training data subsets are randomly drawn with replacement from the whole training dataset.	Each new subset contains the components that were misclassified by previous models.
2	Every model receives an equal weight.	Models are weighted by their performance.
3	Every model is constructed independently.	New models are affected by the performance of the previously developed model.
4	Example: The Random forest model	The AdaBoost uses Boosting techniques
5	If the classifier is unstable (high variance), then apply bagging.	If the classifier is stable and simple (high bias) the apply boosting.

Bootstrap Aggregating, also known as bagging decreases the variance and helps to avoid overfitting.It improve the stability and accuracy of machine learning algorithms used in statistical classification and regression.

Beacuse Bagging is based on Central limit theorem, It says that
$X \sim N(\mu_x, \frac{\sigma_x}{\sqrt{n}})$

Here,
Mean is the same mean as the original distribution and a variance that equals the original variance divided by, the sample size. Hence it shrinks the variance by $\sqrt{n}$

ix. Justify the cost function $L(w)$ used by logistic regression for binary classification and compute its gradient $\frac{\partial L}{\partial W}$ with respect to parameters w.

$p = \sigma(W^T x_i) = \frac{1}{1 + e^{-W^T x_i}}$ **and** $L(W) = \sum_{i=1}^{\mathbb{N}} [y_i log(p_i) + (1 - y_i) log(1 - p_i)]$

Answer:As we know the hypothesis of linear regression is:

$$h_w(x) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 .... + w_i x_i = W^T X$$

For logistic regression, focusing on binary classification here, we have class 0 and class 1 for a given x.To compare with the target, we want to constrain predictions to some values between 0 and 1. That's why **Sigmoid Function** is applied on the raw model output and provides the ability to predict with probability.

As we know $SigmoidFunction = \sigma(z) = \frac{1}{1+e^{-z}}$

So our hypothesis function will become,

$$p = HypothesisFunction = \sigma(W^T x) = \frac{1}{1+e^{(-W^T x)}}$$

What hypothesis function returns is the probability that y = 1, given x, parameterized by W, written as: p = P(y = 1|x; W). Decision boundary can be described as:
Predict 1, if $W^T x \geq 0 \rightarrow p(x) \geq 0.5$;
Predict 0, if $W^T x < 0 \rightarrow p(x) < 0.5$.

$$p(X) = p(Y = 1|X) = \frac{1}{1+e^{(-W^T x)}} = \frac{e^{(W^T x)}}{1+e^{(W^T x)}}$$

$$1 - p(X) = p(Y = 1|X) = 1 - \frac{1}{1+e^{(-W^T x)}} = \frac{1}{1+e^{(W^T x)}}$$

$$\frac{p(X)}{1-p(X)} = e^{W^T x}$$

taking log both sides,

$log(\frac{p(X)}{1-p(X)}) = W^T x$

This is $logit$ of $P(X)$

Given samples $\{x_i, y_i\} \in \mathbb{R}^p X \{0, 1\} \forall i = 1, 2, 3..n$

from $logit$ of $P(x_i)$

$log(\frac{p(x_i)}{1-p(x_i)}) = W^T x_i$

We need to estimate $\{w_0, w_1, w_2, w_3 .... w_i = \hat{W}\}$

Cost Function:
Intuitively, we want to assign more punishment when predicting 1 while the actual is 0 and when predict 0 while the actual is 1. The loss function of logistic regression is doing this exactly which is called Logistic Loss. See as below. If y = 1, looking at the plot below on left, when prediction = 1, the cost = 0, when prediction = 0, the learning algorithm is punished by a very large cost. Similarly, if y = 0, the plot on right shows, predicting 0 has no punishment but predicting 1 has a large value of cost.

we use Maximum Likelyhood Estimation of that, in MLE we take all the data and break it into otwo groups based on their labels.

MLE can be given as:
For smaple labeled "1" estimate W such that $p(x)$ is as close to 1 as possible. i.e. $\pi_{i,y_i=1} p(x_i)$

For smaple labeled "0" estimate W such that $1 - p(x)$ is as close to 1 as possible. i.e. $\pi_{i,y_i=0} (1 - p(x_i))$

We can write it as

$L(W) = \pi_{i,y_i=1} p(x_i) . \pi_{i,y_i=0} (1 - p(x_i))$

In a generalised form,

$L(W) = \pi_{i=1}^{n} p(x_i)^{y_i} . (1 - p(x_i))^{1-y_i}$

Our objective is to maximize this L(W).

Objective : $\max_w L(W)$

or

$- \min_w L(W)$

Alll the $p(x_i)$ and $(1 - p(x_i))$ are the probabilities and we have to maximize it.
So to minimizing this is equivalent to minimizing the log of that function. Because this function is always

positive.

log likelihood:

$$l(W) = -logL(W) = -\sum_{i=1}^{n} y_i logp(x_i) + (1-y_i)log(1-p(x_i))$$

For minimizing $l(W)$,we need to do compute its gradient $\frac{\partial l}{\partial W}$

As we know that differentiation of sigmoid function will give :
$\frac{d}{dz}\sigma(z) = \sigma(z)(1-\sigma(z))$

$$\frac{\partial l}{\partial W} = -\sum_{i=1}^{n} y_i \frac{\partial}{\partial W} logp(x_i) + (1-y_i)\frac{\partial}{\partial W} log(1-p(x_i))$$

$$\frac{\partial l}{\partial W} = -\sum_{i=1}^{n} y_i \frac{\partial}{\partial W} log\sigma(W^T x_i) + (1-y_i)\frac{\partial}{\partial W} log(1-\sigma(W^T x_i))$$

$$\frac{\partial l}{\partial W} = -\sum_{i=1}^{n} y_i \frac{1}{\sigma(W^T x_i)}\frac{\partial}{\partial W}\sigma(W^T x_i) + (1-y_i)\frac{1}{1-\sigma(W^T x_i)}\frac{\partial}{\partial W}(1-\sigma(W^T x_i))$$

$$\frac{\partial l}{\partial W} = -\sum_{i=1}^{n} y_i \frac{1}{\sigma(W^T x_i)}\sigma(W^T x_i)(1-\sigma(W^T x_i))\frac{\partial}{\partial W}(W^T x_i)$$

$$-(1-y_i)\frac{1}{1-\sigma(W^T x_i)}\sigma(W^T x_i)(1-\sigma(W^T x_i))\frac{\partial}{\partial W}(W^T x_i)$$

$$\frac{\partial l}{\partial W} = -\sum_{i=1}^{n} y_i(1-\sigma(W^T x_i))(x_i) - (1-y_i)\sigma(W^T x_i)(1-\sigma(W^T x_i))(x_i)$$

$$\frac{\partial l}{\partial W} = -\sum_{i=1}^{n} (y_i - \sigma(W^T x_i))(x_i)$$

$$\frac{\partial l}{\partial W} = -\sum_{i=1}^{n} (y_i - p(x_i))(x_i)$$

$$\frac{\partial l}{\partial W} = -\sum_{i=1}^{n} (y_i - \hat{y}_i)(x_i)$$

Gradient = $\frac{\partial l}{\partial W} = \sum_{i=1}^{n} (\hat{y}_i - y_i)(x_i)$

We update W as

$$W^{t+1} = W^t - \eta\frac{\partial l}{\partial W}$$

where t = iteration .

x.It takes 10 minutes to train a SVM algorithm on a dataset with 100K data points. How much time do you think it might take to train the same algorithm on 1 M data points?

Answer:SVM training can be arbitrary long, this depends on dozens of parameters: C parameter - greater the missclassification penalty, slower the process. kernel - more complicated the kernel, slower the process (rbf is the most complex from the predefined ones) data size/dimensionality etc.

From Hands-On Machine Learning Book:

Class	Time Complexity
Linear SVC	$O(mXn)$
SVC	$O(m^2Xn)$ to $O(m^3Xn)$

m= No. of training instances
n=No. of features

Case1 Linear SVC:
Given in case of 100K =10^5 data points time required to train =10 minutes
Assume that number of features are constant.
Time required to train 1M =10^6 is $\frac{10X10^6}{10^5} = 100$ minutes.

It scales almost linearly with the number of training instances and the number of features.

Case2 SVC: Assume the minimum time complexity $O(m^2Xn)$
Given in case of 100K =10^5 data points time required to train =10 minutes
Assume that number of features are constant.
Time required to train 1M =10^6 is $\frac{10X(10^6)^2}{(10^5)^2} = 1000$ minutes.

The training time complexity is usually between $O(m^2Xn)$ and $O(m^3Xn)$. Unfortunately, this means that it gets dreadfully slow when the number of training instances gets large.

1. (10 marks) Deriving Decision Boundary for Learning with Prototypes LwP Classifier
These are some training examples of +ve class: (2,3,4); (1,1,2); (0,2,0) These are some training examples of -ve class: (4,3,2); (2,1,1); (3,5,3)

+ve class(p)	-ve class(n)
p1=(2,3,4)	n1=(4,3,2)
p2=(1,1,2)	n2=(2,1,1)
p3=(0,2,0)	n3=(3,5,3)

(1 mark) Compute the average feature vectors of +ve and -ve class examples. Let us call them μ+ and $\mu-$

Average Feature Vector for +ve class=$\mu^+ = (\frac{2+1+0}{3}, \frac{3+1+2}{3}, \frac{4+2+0}{3}) = (1,2,2)$

Average Feature Vector for -ve class=$\mu^- = (\frac{4+2+3}{3}, \frac{3+1+5}{3}, \frac{2+1+3}{3}) = (3,3,2)$

(1 marks) Determine the vector w which joins μ+ and $\mu-$: $w = \mu+ - \mu-$

The vector w which joins μ^+ and μ^-
$W = \mu^+ - \mu^- = (1,2,2) - (3,3,2) = (1-3, 2-3, 2-2) = (-2,-1,0)$

(2 marks) Equation of the decision boundary can be written as $W^T x + b = 0$. We know that the decision boundary passes through the mid-point of the prototypes.
Mid-point is given by $\frac{1}{2}(\mu^+ + \mu^-)$
Compute the value of b

Mid Point of the prototypes=x_{mid}=$\frac{1}{2}(\mu^+ + \mu^-) = \frac{1}{2}(1+3, 2+3, 2+2) = (2, 2.5, 2)$
We know that the decision boundary passes through the mid-point of the prototypes. hence must satisfy the decision boundary equation.
$W^T x + b = 0$

$$\begin{bmatrix} -2 & -1 & 0 \end{bmatrix} \quad . \begin{bmatrix} 2 \\ 2.5 \\ 2 \end{bmatrix} \quad + b = 0$$

we will get

$-4 - 2.5 + 0 + b = 0$
$b = 6.5$

(2 marks) Compute the distances of the training examples from the decision boundary. Are there any errors in classification using the LwP classifier?

Answer: Since given training examples are in 3-dimensional space. The decision hyperplane would be a 2 dimensional surface.

The equation of hyperplane can be writyen as:

$$\begin{bmatrix} -2 & -1 & 0 \end{bmatrix} \quad . \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad + 6.5 = 0$$

we will get
$-2x - y + 0 + 6.5 = 0$

Distance calculation using below formula:

$$\text{distance}(ax + by + c = 0, (x_0, y_0)) = \frac{|ax_0 + by_0 + c|}{\sqrt{a^2 + b^2}}.$$

Distance from decision boundary:
p1=(2,3,4): (- 4 - 3 + 6.5)/√5 = - 0.5/√5
p2=(1,1,2): (-2 - 1 + 6.5)/√5 = 3.5/√5
p3=(0,2,0) : (-2 + 6.5)/√5 = 4.5/√5
The point (2,3,4) is misclassified.

Distance of - ve class points from decision boundary are:
n1=(4,3,2):(-8 -3 + 6.5)/√5 = -4.5/√5

n2=(2,1,1):(-4 -1 + 6.5)/√5 = 1.5/√5
n3=(3,5,3):(-6 - 5 +6.5)/√5 = -4.5/√5
The point (2,1,1) is misclassified.

+ve class(p)	-ve class(n)
d(D,p1)=-ve(misclassified)	d(D,n1)=-ve
d(D,p2)=+ve	d(D,n2)=+ve(misclassified)
d(D,p3)=+ve	d(D,n3)=-ve

Hence there are errors in classification using LWP classifier

(4 marks) Find the equation of a linear classifier which separates the data without any error? If no such classifier exists, give a justification.
Answer: We can find the linear classifier which can separate the data without any error.
Using perceptron algorithm:

> if $y_i(W^T x_i + b) < 0$ then
>
> > While ($y_i(W^T x_i + b) < 0$):
> >
> > > $w^* = w + y_i * x_i$
> > > $b^* = b + y_i$

where $y_i = 1$ for positive class and $y_i = -1$ for negative class.
Program to find the linear classifier using perceptron algorithm:

```
import numpy as np
w=np.array([-2,-1,0])
b=6.5
data_points=[(-1,[2,1,1]),(1,[2,3,4]),(1,[1,1,2]),(1,[0,2,0]),(-1,[4,3,2]), (-1,[3,5,3])]
for i,j in data_points:
    if i*(w.T@np.array(j))+b<0:
        t=0
      #iterations
        while(i*(w.T@np.array(j))+b<0) and t<100 :
            w=w+i*np.array(j)
            b=b+i
```

Output:-
w=array([-4, -1, 2])
b=6.5

```
data_points=[(1,[2,3,4]),(1,[1,1,2]),(1,[0,2,0]),(-1,[4,3,2]),(-1,[2,1,1]),(-1,[3,5,3])]
for i,j in data_points:
    if i*(w.T@np.array(j))+b>0:
        print("Correctly Classified!!!")
    else:
        print("Incorrectly Classified!!!")
```

Output:-

Correctly Classified!!!
Correctly Classified!!!
Correctly Classified!!!
Correctly Classified!!!
Correctly Classified!!!
Correctly Classified!!!