Q I

I) Understand the working of the following classifier algorithms and trace the same for a sample dataset (min 10 records) which involves 2 classes (Binary Classifier) eg: 'Yes' or 'No’, 'True' or 'False'.

a) Decision Tree

Give pseudo code and trace decision tree algorithms.

Understand attribute selection measures such as Information gain, gain ratio (use anyone for the trace).

Pseudo Code

S – Samples, A – Attribute List

1. create a node N

2. If all samples are of the same class C then label N with C; Stop;

3. If A is empty then label N with the most common class C in S (majority voting); Stop;

4. Select a ∈ A, with the highest information gain; Label N with a;

5. For each value v of a:

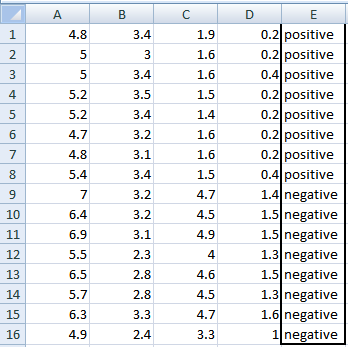
a. Grow a branch from N with condition a=y;

b. Let be the subset of samples in S with a=y;

c. If is empty then attach a leaf labelled with the most common class in S;

d. Else attach the node generated for Samples S, Attribute List A-a

Trace for Information Gain



A, B, C, D attributes can be considered as predictors

E column class labels can be considered as a target variable.

For constructing a decision tree from this data, we have to convert continuous data into categorical data.

Categorization is done by,

A: >=5 or <5

B: >=3 or <3

C: >=4.2 or <4.2

D: >=1.4 or <1.4

Entropy used for finding IG is given by,



Now calculating IG for A,

Var A has value >=5 for 12 records out of 16 and 4 records with value <5 value.

(A >= 5 and class is positive): 5/12

(A >= 5 and class is negative): 7/12

Entropy(5,7) = -1 \* ( (5/12)\*log2(5/12) + (7/12)\*log2(7/12)) = 0.9799

(A <5 and class is positive): ¾

(A <5 and class is negative): ¼

Entropy(3,1) =  -1 \* ( (3/4)\*log2(3/4) + (1/4)\*log2(1/4)) = 0.81128

Entropy(Target, A) = P(>=5) \* E(5,7) + P(<5) \* E(3,1)  
= (12/16) \* 0.9799 + (4/16) \* 0.81128 = 0.937745

Information Gain for A = E(Target) – E(Target, A) = 1 – 0.937745 = 0.062255

Similarly,

IG(A) = 0.062255

IG(B) = 0.707095

IG(C) = 0.5488

IG(D) = 0.41189

Using IG values in descending order we can construct DT as,

b) Naive Bayesian Classifier (NBC)

Read about Bayes theorem, conditional class independence, prior and posterior probabilities.

How to handle zero probability scenario (Laplacian Estimator)

Give a short pseudo code and trace.

Pseudo Code

Target Variable – Y (true or false -- binary)

Predictor Variables – X1 .. Xn

1. From the data, Estimate P(Y = true) and P(Y = false)
2. To do (1), For every value xij of each input attribute Xi,
   1. Find P(Xi = xij | Y = true)
   2. Find P(Xi = xij | Y = false)

Now, Classify a new X’ by,

(Let PROD(true) = {P(X1|Y=true). P(X2|Y=true)…P(Xn|Y=true)})

(Let PROD(false) = {P(X1|Y= false). P(X2|Y= false)…P(Xn|Y= false)})

Y’ (Predicted Class for X’) =

True if P(Y = true)PROD(true) > P(Y = false)PROD(false)

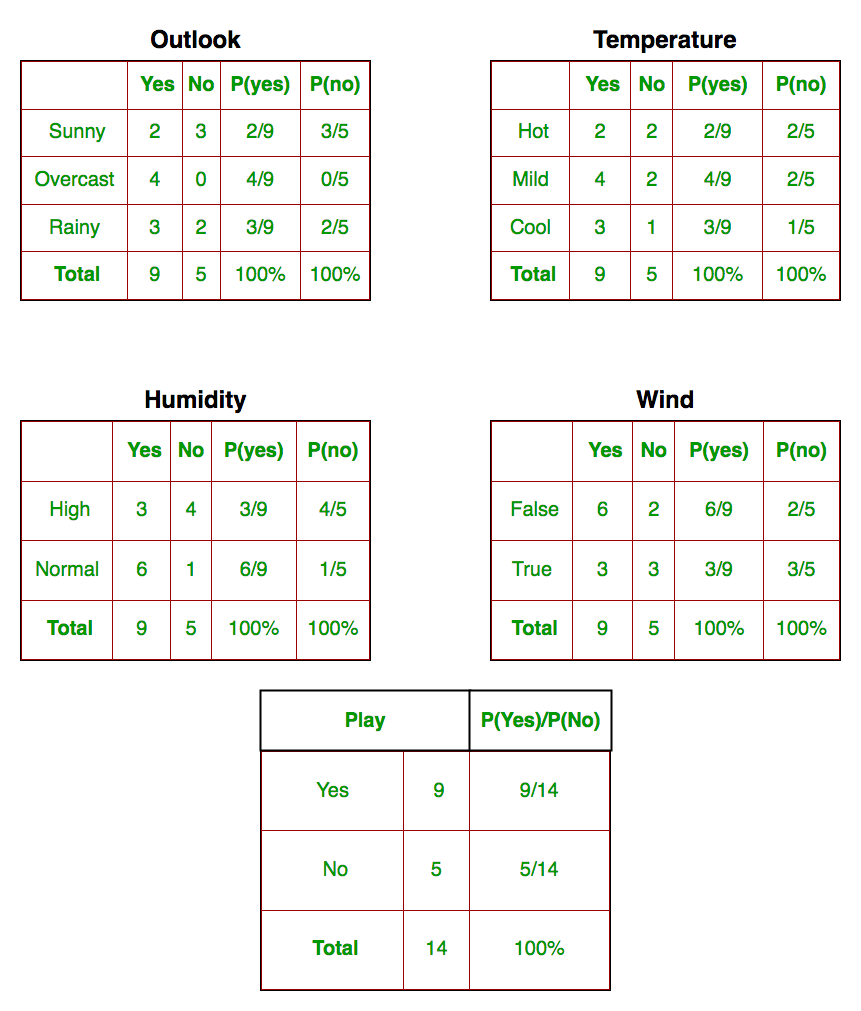
False if P(Y = true)PROD(true) <= P(Y = false)PROD(false)

Trace

| **INDEX** | **OUTLOOK** | **TEMPERATURE** | **HUMIDITY** | **WINDY** |
| --- | --- | --- | --- | --- |
| 0 | Rainy | Hot | High | False | No |
| 1 | Rainy | Hot | High | True | No |
| 2 | Overcast | Hot | High | False | Yes |
| 3 | Sunny | Mild | High | False | Yes |
| 4 | Sunny | Cool | Normal | False | Yes |
| 5 | Sunny | Cool | Normal | True | No |
| 6 | Overcast | Cool | Normal | True | Yes |
| 7 | Rainy | Mild | High | False | No |
| 8 | Rainy | Cool | Normal | False | Yes |
| 9 | Sunny | Mild | Normal | False | Yes |
| 10 | Rainy | Mild | Normal | True | Yes |
| 11 | Overcast | Mild | High | True | Yes |
| 12 | Overcast | Hot | Normal | False | Yes |
| 13 | Sunny | Mild | High | True | No |

We assume that all attributes are independent and are of equal importance to outcome.

Then we find P(Xi=xij|Y=Yes) and P(Xi=xij|Y=No)



Now that we have this data,

Suppose test data is

Today = (Sunny, Hot, Normal, False)

By Bayes Theorem,

P(Yes|today) =

= =

P(No|today) =

= =

As P(today) > 0,

Clearly P(Yes) > P(No)

So, predicted value for playing golf is YES.

For Problems with Zero Probability, use

Laplace smoothing: By applying this method, prior probability and conditional probability are, K is the number of different values in y and A is the number of different values in aj.

c) Neural Network Classifier (Back Propagation Network(BPN))

Understand basic terminologies - topology of a network, input layer, hidden layer, output layer, activation functions, weight, bias.

Strength and limitations of a neural network.

Take a sample network to learn the basics. For instance, learning the behaviour of OR gate, AND gate.

Take a network involving atleast one hidden layer and trace the BPN algorithm assuming a target class for one epoch.

Trace

Example Dataset

|  |  |  |
| --- | --- | --- |
| Class | x | y |
| class 1 (w1) | 2 | 2 |
| class 1 (w1) | -1 | 2 |
| class 1 (w1) | 1 | 3 |
| class 1 (w1) | -1 | -1 |
| class 1 (w1) | 0.5 | 0.5 |
| class 2 (w2) | -1 | -3 |
| class 2 (w2) | 0 | -1 |
| class 2 (w2) | 1 | -2 |
| class 2 (w2) | -1 | -2 |
| class 2 (w2) | 0 | -2 |

We take a neural network with 3 layers

1. 1 – input layer (2 nodes + 1 bias)
2. 1 – hidden layer (2 nodes + 1 bias)
3. 1 – output layer (2 nodes)

Hyperparameters are taken as,

1. Learning Rate = 0.5
2. Activation Function – Sigmoid
3. Epochs = 1000
4. Weights = Random

Now, for forward propagation,

Value at a node is given by,

is the i+1 th node in k th layer (it also includes the bias node).

n is the total number of nodes in i th layer.

f(.) is the activation function.

Therefore, if we use the above formula, we get,

(H1) = f(W111 \* x + W121 \* y + W131 \* 1)

(H2) = f(W112 \* x + W122 \* y + W132 \* 1)

(O1) = f(W211 \* H1 + W221 \* H2 + W231 \* 1)

(O2) = f(W212 \* H1 + W222 \* H2 + W232 \* 1)

When we plug in the values for the first value of the dataset i.e. (2, 2), We get,

H1 = f(0.13436424411240122 \* 2 + 0.8474337369372327 \* 2 + 0.763774618976614 \* 1) =f(2.7273705810758817) = 0.9386225302230806

H2 = f(0.2550690257394217 \* 2 + 0.49543508709194095 \* 2 + 0.4494910647887381 \* 1) = f(1.9504992904514635) = 0.8755010741482664

O1 = f(0.651592972722763 \* 0.9386225302230806 + 0.7887233511355132 \* 0.8755010741482664 + 0.0938595867742349 \* 1) = f(1.3959875726318156) = 0.8015464048450159 29 B ANIRUDH

O2 = f(0.02834747652200631 \* 0.9386225302230806 + 0.8357651039198697 \* 0.8755010741482664 + 0.43276706790505337 \* 1) = f(1.1910878942610617) = 0.7669355767878618

Next is backpropagation of the error,

1. For a unit j in the output layer, the error Errj is computed by,

Outj (1 - Outj ) is the derivative of the logistic (sigmoid) function.

Tj is the target value of the Oj node.

1. In hidden layer,

is the weight of the connection from unit j to a unit k in the next higher layer.

is the is the error of unit k.

The weights and biases are updated to reflect the propagated errors. Weights are updated by the following equations, where ∆ is the change in weight :

(l – learning rate)

So, the error for Err1 for the output layer is,

Erri = 0.8015464048450159 \* (1 - 0.8015464048450159) \* (1 - 0.8015464048450159) = 0.03156796688859639

The change in weight for Hidden Layer is,

∆ W11 = (0.5) \* (0.03156796688859639) \* (0.9386225302230806) = 0.014815202477486385

W11 = 0.651592972722763 + 0.014815202477486385 = 0.6664081752002493

Similarly, the error for Err1 for the Hidden Layer is,

Erri = 0.9386225302230806 \* (1 - 0.9386225302230806) \* (0.03156796688859639 \* 0.651592972722763 + (0.7669355767878618) \* (1 - 0.7669355767878618) \* (0 - 0.7669355767878618) \* 0.02834747652200631) = 0.0009611362816286504

The change in weight for Hidden Layer is,

∆ W11 = (0.5) \* (0.0009611362816286504) \* (2) = 0.0009611362816286504

W11 = 0.13436424411240122 + 0.0009611362816286504 = 0.13532538039402986

Now, again forward propagate and backpropagate and update weights and repeat.

d) SVM or Nearest Neighbour Classifier (Anyone)

Understand the algorithm/working and give the pseudocode and the trace

Pseudo Code

k-Nearest Neighbour

Classify (X, Y, x) where X: training data, Y: class labels of X, x: test sample

For i = 1 to m do

Computer distance d(Xi, x)

End for

Computer set I containing indices for the k smallest distances d(Xi, x)

Return majority label for {Yi where i ∈ I}

Trace

Consider binary Y with attributes X

**K = 3**

|  |  |  |
| --- | --- | --- |
| X1 | X2 | Y |
| 1 | 2 | False |
| 2 | 3 | True |
| 3 | 1 | True |
| 4 | 3 | False |
| 5 | 5 | False |
| 6 | 1 | False |
| 7 | 2 | True |

Now for a test x = (2.5, 3.5),

D(X, x) =>

|  |  |  |
| --- | --- | --- |
| X1 | X2 | D(X, x) |
| 1 | 2 | 2.12 |
| 2 | 3 | 0.71 |
| 3 | 1 | 2.55 |
| 4 | 3 | 1.58 |
| 5 | 5 | 2.91 |
| 6 | 1 | 4.30 |
| 7 | 2 | 4.74 |

As K = 3, minimum 3 distances is for (2, 3), (4, 3), (1, 2)

And their Y is True, False, False

SO, majority is FALSE = y for x