12. Given below is the data of five students who took a proficiency test as well as language course. S.No. X2 Y (Output) 1 95 85 2 85 95 3 80 70 4 70 65 5 60 70 a) Use the least square approximation to estimate the linear equation that best predicts language course performance, based on proficiency test scores? b) Compute the sum of squared error (SSE) using the estimated model. c) If a student scored 80 on the proficiency test, what marks would we expect her to obtain in the language course?

ChatGPT

a) We can use linear regression to estimate the linear equation that best predicts language course performance based on proficiency test scores. We can represent the data as follows:

| **S.No.** | **X2** | **Y (Output)** |
| --- | --- | --- |
| 1 | 95 | 85 |
| 2 | 85 | 95 |
| 3 | 80 | 70 |
| 4 | 70 | 65 |
| 5 | 60 | 70 |

Let's assume that the linear equation is of the form Y = b0 + b1\*X2. We want to estimate the values of b0 and b1.

We can use the following formulas to estimate b0 and b1:

b1 = Σ[(Xi2 - X̄2)\*(Yi - Ȳ)] / Σ[(Xi2 - X̄2)^2]

b0 = Ȳ - b1\*X̄2

where Xi2: proficiency test score of the i-th student X̄2: mean proficiency test score Yi: language course score of the i-th student Ȳ: mean language course score

Using the above formulas, we can calculate:

X̄2 = (95+85+80+70+60)/5 = 78 Ȳ = (85+95+70+65+70)/5 = 77

b1 = [(95-78)*(85-77) + (85-78)*(95-77) + (80-78)*(70-77) + (70-78)*(65-77) + (60-78)\*(70-77)] / [(95-78)^2 + (85-78)^2 + (80-78)^2 + (70-78)^2 + (60-78)^2] = 0.684

b0 = 77 - 0.684\*78 = 26.652

Therefore, the linear equation that best predicts language course performance based on proficiency test scores is Y = 26.652 + 0.684\*X2.

b) To compute the sum of squared error (SSE), we need to calculate the difference between the actual Y values and the predicted Y values for each student, square these differences, and add them up. Mathematically,

SSE = Σ(Yi - Ŷi)^2

where Yi: language course score of the i-th student Ŷi: predicted language course score of the i-th student

Using the linear equation we estimated in part (a), we can calculate the predicted Y values as follows:

| **S.No.** | **X2** | **Y (Output)** | **Ŷ (predicted Y)** |
| --- | --- | --- | --- |
| 1 | 95 | 85 | 89.912 |
| 2 | 85 | 95 | 84.604 |
| 3 | 80 | 70 | 81.95 |
| 4 | 70 | 65 | 75.642 |
| 5 | 60 | 70 | 69.334 |

SSE = (85-89.912)^2 + (95-84.604)^2 + (70-81.95)^2 + (65-75.642)^2 + (70-69.334)^2 = 505.215

Therefore, the sum of squared error (SSE) is 505.215.

c) If a student scored 80 on the proficiency test, we can use the linear equation we estimated in part



14. Find the linear regression coefficients using gradient descent method for the following dataset when learning rate = 0.2. Carry out the process for 2 iterations. X 0 1 2 3 4 Y 3 4 5 4 6

ChatGPT

2 / 2

We need to find the coefficients of the linear regression model:

Y = b0 + b1\*X

where Y is the dependent variable and X is the independent variable.

Let's initialize the coefficients with some random values, say b0 = 0 and b1 = 0. Then, we can use the following iterative update equations to find the optimal values of b0 and b1:

b0 = b0 - learning\_rate \* (2 \* (b0 + b1*X[i] - Y[i])) b1 = b1 - learning\_rate \* (2 \* (b0 + b1*X[i] - Y[i]) \* X[i])

where i is the index of the current data point and learning\_rate is the step size we take in each iteration.

Let's carry out the process for 2 iterations:

Iteration 1: For i = 0: b0 = 0 - 0.2 \* (2 \* (0 + 0*0 - 3)) = 0.6 b1 = 0 - 0.2 \* (2 \* (0 + 0*0 - 3) \* 0) = 0

For i = 1: b0 = 0.6 - 0.2 \* (2 \* (0.6 + 0*1 - 4)) = 0.96 b1 = 0 - 0.2 \* (2 \* (0.6 + 0*1 - 4) \* 1) = 0.8

For i = 2: b0 = 0.96 - 0.2 \* (2 \* (0.96 + 0.8*2 - 5)) = 1.216 b1 = 0.8 - 0.2 \* (2 \* (0.96 + 0.8*2 - 5) \* 2) = 0.64

For i = 3: b0 = 1.216 - 0.2 \* (2 \* (1.216 + 0.64*3 - 4)) = 1.0592 b1 = 0.64 - 0.2 \* (2 \* (1.216 + 0.64*3 - 4) \* 3) = 0.2688

For i = 4: b0 = 1.0592 - 0.2 \* (2 \* (1.0592 + 0.2688*4 - 6)) = 0.92736 b1 = 0.2688 - 0.2 \* (2 \* (1.0592 + 0.2688*4 - 6) \* 4) = -0.4032

Iteration 2: For i = 0: b0 = 0.92736 - 0.2 \* (2 \* (0.92736 - 0.4032*0 - 3)) = 1.100928 b1 = -0.4032 - 0.2 \* (2 \* (0.92736 - 0.4032*0 - 3) \* 0) = -0.4032

For i = 1: b0 = 1.100928 - 0.2 \* (2 \* (1.100928 - 0.4032*1 - 4)) = 1.2987232 b1 = -0.4032 - 0.2 \* (2 \* (1.100928 - 0.4032*1 - 4) \* 1) = -0.



15. Consider the dataset given below, which categorizes an article either as Technical (Class 1) or Non-Technical (Class 0) based on the time spent in reading (in Hours) and the number of sentences (in multiples of 1000) in that article. Time (Hours) Sentences (in multiple of 1000) Y (Output) 2.7 2.5 0 3 3 0 5.9 2.2 1 7.7 3.5 1 a) Using the above data, build a logistic regression model to predict the class of an article using gradient descent method. Assume 𝑙𝑒𝑎𝑟𝑛𝑖𝑛𝑔 𝑟𝑎𝑡𝑒=0.3. Further, in the first iteration the value of the coefficients is 0, and the bias is 1. Use two iterations of the gradient descent process to learn the model parameters. b) Compute the error in prediction. c) Use the above model to predict the article type of an article which requires 6.2 hours of reading time and contains 3100 sentences.

ChatGPT

2 / 2

a) We need to first define the logistic regression model as:

$y = \frac{1}{1+e^{-(\beta\_0 + \beta\_1x\_1 + \beta\_2x\_2)}}$

where $y$ is the predicted output, $x\_1$ and $x\_2$ are the input variables (time and sentences), and $\beta\_0$, $\beta\_1$, and $\beta\_2$ are the coefficients that we need to learn.

The cost function for logistic regression is given as:

$J(\beta) = -\frac{1}{m}\sum\_{i=1}^{m}[y^{(i)}\log(h\_{\beta}(x^{(i)})) + (1-y^{(i)})\log(1-h\_{\beta}(x^{(i)}))]$

where $m$ is the number of training examples, $y^{(i)}$ is the actual output for the $i^{th}$ example, and $h\_{\beta}(x^{(i)})$ is the predicted output for the $i^{th}$ example.

We can update the coefficients using the gradient descent method as:

$\beta\_j := \beta\_j - \alpha \frac{\partial J(\beta)}{\partial \beta\_j}$

where $\alpha$ is the learning rate.

For our dataset, we can define the logistic regression model as:

$y = \frac{1}{1+e^{-(\beta\_0 + \beta\_1x\_1 + \beta\_2x\_2)}}$

where $x\_1$ is the time, $x\_2$ is the sentences, and $y$ is the output. We can use the given initial values of the coefficients as $\beta\_0=1$, $\beta\_1=0$, and $\beta\_2=0$.

Now, we can calculate the predicted output using the logistic regression equation for each example:

For the first example, $(x\_1,x\_2,y) = (2.7,2.5,0)$:

$y\_{pred} = \frac{1}{1+e^{-(1 + 0*2.7 + 0*2.5)}} = 0.731$

For the second example, $(x\_1,x\_2,y) = (3,3,0)$:

$y\_{pred} = \frac{1}{1+e^{-(1 + 0*3 + 0*3)}} = 0.731$

For the third example, $(x\_1,x\_2,y) = (5.9,2.2,1)$:

$y\_{pred} = \frac{1}{1+e^{-(1 + 0*5.9 + 0*2.2)}} = 0.731$

For the fourth example, $(x\_1,x\_2,y) = (7.7,3.5,1)$:

$y\_{pred} = \frac{1}{1+e^{-(1 + 0*7.7 + 0*3.5)}} = 0.731$

Now we can calculate the cost function for the given dataset as:

$J(\beta) = -\frac{1}{4}[(0\log(0.731) + (1-0)\log(1-0.731)) + (0\log(0.731) + (1-0)\log(1-0.731)) + (1\log(0.731) + (1-1)\log(1-0.731)) + (1\log(0.731)



17. Given the set of values X = (3, 9, 11, 5, 2)T and Y = (1, 8, 11, 4, 3)T. Evaluate the regression coefficients using ordinary least square method.

ChatGPT

To find the regression coefficients using the ordinary least square method, we need to solve the following equation:

β = (XTX)-1XTY

where, X is the matrix of independent variables (features) Y is the dependent variable (target)

Here, X = [1, 3 1, 9 1, 11 1, 5 1, 2]

Y = [1 8 11 4 3]

We need to find the value of β such that the error term ε = Y - Xβ is minimized.

Using the above equation, we can calculate the β as follows:

XTX = [5 30 30 650]

(XTX)-1 = [0.454545 -0.072727 -0.072727 0.012121]

XTY = [27 352]

β = [0.5455 0.5091]

Therefore, the regression equation is:

Y = 0.5455 + 0.5091X

This means that for every unit increase in X, the predicted value of Y increases by 0.5091.

Thus, the regression coefficients using ordinary least square method are 0.5455 and 0.5091.



19. Find the least square regression line for the given dataset using normal equation method. Show computation at each step. X1 X2 Y (Output) 1 9 14 2 1 7 3 2 12 4 3 16 5 4 20

ChatGPT

The least square regression line using normal equation method can be found by solving the equation β = (XT X)-1XT Y, where X is the design matrix, Y is the output vector and β is the vector of regression coefficients.

First, we need to create the design matrix X and output vector Y:

X = [1 1 2 3 4; 9 1 1 3 4;] Y = [14; 7; 12; 16; 20;]

Note that the first column of X is all ones, which is required for the intercept term. Now, we can compute the regression coefficients using the normal equation:

β = (XT X)-1XT Y

XT X = [5 15; 15 86] XT Y = [69; 301]

Now, we need to compute the inverse of XT X:

XT X-1 = 1/35 \* [86 -15; -15 5]

Substituting the values:

β = (1/35 \* [86 -15; -15 5]) \* [69; 301] β = [1.6; 2.8; 2.2]

Therefore, the least square regression line is given by:

Y = 1.6 + 2.8 X1 + 2.2 X2

This means that for every unit increase in X1, the predicted value of Y increases by 2.8, and for every unit increase in X2, the predicted value of Y increases by 2.2.



20. Consider the dataset given below having two input variables x1, x2 and one output variable y. Update the coefficients b0, b1 and b2 using gradient descent for the Logistic Regression model. Assume the learning rate = 0.3 and initial values of coefficients as b0 = -0.5, b1= 1 and b2= -1. Perform one iteration of gradient descent. x1 1 3 8 6 x2 2 5 3 2 y 0 0 1 1

ChatGPT

We can use the following logistic regression model to predict the output variable:

y = 1 / (1 + exp(-z))

where z = b0 + b1*x1 + b2*x2

Let's calculate the gradient of the cost function with respect to each coefficient:

dJ/db0 = (1/m) \* sum(y\_hat - y) dJ/db1 = (1/m) \* sum((y\_hat - y) \* x1) dJ/db2 = (1/m) \* sum((y\_hat - y) \* x2)

where y\_hat is the predicted output given by the logistic regression model and m is the number of training examples.

To update the coefficients, we can use the following update rule:

b\_new = b\_old - learning\_rate \* dJ/db

where b\_old is the old value of the coefficient and learning\_rate is the learning rate.

Let's calculate the predicted output and the gradients for the given dataset:

x1 1 3 8 6 x2 2 5 3 2 y 0 0 1 1

b0 = -0.5, b1 = 1, b2 = -1

z = -0.5 + 1*x1 - 1*x2 y\_hat = 1 / (1 + exp(-z))

y\_hat = [0.62245933, 0.73105858, 0.05464917, 0.11920292]

dJ/db0 = (1/4) \* sum(y\_hat - y) = -0.19124846 dJ/db1 = (1/4) \* sum((y\_hat - y) \* x1) = -0.16938966 dJ/db2 = (1/4) \* sum((y\_hat - y) \* x2) = 0.20515822

Now, we can update the coefficients using the update rule:

b0\_new = -0.5 - 0.3\*(-0.19124846) = -0.4481255378 b1\_new = 1 - 0.3\*(-0.16938966) = 1.050817898 b2\_new = -1 - 0.3\*(0.20515822) = -1.061547466

Therefore, after one iteration of gradient descent, the updated coefficients are:

b0 = -0.4481255378, b1 = 1.050817898, b2 = -1.061547466.



23. The sales of a company (in million rupees) for each year are shown in the table below: x (year) y (sales) 2005 12 2006 19 2007 29 2008 37 2009 45 2010 49 (a) Find the least square regression line y = ax+b (b) Use the least squares regression line as a model to estimate the sales of the company in 2013.

ChatGPT

To find the least square regression line y = ax+b, we need to calculate the values of a and b using the following formulas:

b = (Σy \* Σx^2 - Σx \* Σxy) / (n \* Σx^2 - (Σx)^2) a = (n \* Σxy - Σx \* Σy) / (n \* Σx^2 - (Σx)^2)

where Σ represents the sum of, n is the number of data points, x and y are the data points.

We have the following values:

x = [2005, 2006, 2007, 2008, 2009, 2010] y = [12, 19, 29, 37, 45, 49] n = 6

Σx = 12035 Σy = 191 Σx^2 = 24184185 Σxy = 384477

Using the above formulas, we get:

b = (191 \* 24184185 - 12035 \* 384477) / (6 \* 24184185 - (12035)^2) = -1044.5 a = (6 \* 384477 - 12035 \* 191) / (6 \* 24184185 - (12035)^2) = 19.85

Therefore, the least squares regression line is y = 19.85x - 1044.5

To estimate the sales of the company in 2013, we can substitute x = 2013 in the above equation:

y = 19.85 \* 2013 - 1044.5 = 42.28

Therefore, the estimated sales of the company in 2013 is 42.28 million rupees.



31. Consider the following 10 training instances. No. Color Type Mileage Tested 1 Blue Sports Average Yes 2 Blue SUV Average No 3 Blue Sports High Yes 4 Pink Sports High No 5 Pink Sports Average Yes 6 Pink SUV Average No 7 Blue SUV High Yes 8 Red SUV High No 9 Red SUV Average Yes 10 Red Sports Average Yes Compute the following probabilities. a. P(Blue|Yes) b. P(Sports|No) c. P(Red|Yes) d. P(Pink|Yes)

ChatGPT

To compute the probabilities, we need to use Bayes' theorem, which states:

P(A|B) = P(B|A) \* P(A) / P(B)

where A and B are events.

a. P(Blue|Yes) P(Blue|Yes) = P(Yes|Blue) \* P(Blue) / P(Yes) We count the number of instances where the car is Blue and Tested (Yes):

* P(Yes|Blue) = 2/3
* P(Blue) = 4/10
* P(Yes) = 6/10 P(Blue|Yes) = (2/3) \* (4/10) / (6/10) = 0.2667

b. P(Sports|No) P(Sports|No) = P(No|Sports) \* P(Sports) / P(No) We count the number of instances where the car is Sports and not Tested (No):

* P(No|Sports) = 1/3
* P(Sports) = 4/10
* P(No) = 4/10 P(Sports|No) = (1/3) \* (4/10) / (4/10) = 0.3333

c. P(Red|Yes) P(Red|Yes) = P(Yes|Red) \* P(Red) / P(Yes) We count the number of instances where the car is Red and Tested (Yes):

* P(Yes|Red) = 1/2
* P(Red) = 3/10
* P(Yes) = 6/10 P(Red|Yes) = (1/2) \* (3/10) / (6/10) = 0.25

d. P(Pink|Yes) P(Pink|Yes) = P(Yes|Pink) \* P(Pink) / P(Yes) We count the number of instances where the car is Pink and Tested (Yes):

* P(Yes|Pink) = 2/3
* P(Pink) = 3/10
* P(Yes) = 6/10 P(Pink|Yes) = (2/3) \* (3/10) / (6/10) = 0.1



34. Use Naïve Bayes classification rule for the following training data, predict whether an old student having high income and excellent credit rating will buy a computer or not. Id Age Income Student Credit Rating Buys Computer 1 Young High No Fair No 2 Young High No Excellent No 3 Middle High No Fair Yes 4 Old Medium No Fair Yes 5 Old Low Yes Fair Yes 6 Old Low Yes Excellent No 7 Middle Low Yes Excellent Yes 8 Young Medium No Fair No 9 Young Low Yes Fair Yes 10 Old Medium Yes Fair Yes 11 Young Medium Yes Excellent Yes 12 Middle Medium No Excellent Yes 13 Middle High Yes Fair Yes 14 Old Medium No Excellent No

ChatGPT

To apply Naïve Bayes classification rule, we need to calculate the conditional probabilities of buying and not buying a computer given the values of the attributes (age, income, student, and credit rating) for the training data.

First, let's calculate the prior probabilities of buying and not buying a computer.

P(Buys Computer) = 9/14 P(Does not buy computer) = 5/14

Next, let's calculate the conditional probabilities for each attribute given the class (buys computer or does not buy computer).

Conditional Probabilities of Buys Computer: P(Young|Buys Computer) = 2/9 P(Middle|Buys Computer) = 2/9 P(Old|Buys Computer) = 5/9 P(High|Buys Computer) = 3/9 P(Medium|Buys Computer) = 4/9 P(Low|Buys Computer) = 5/9 P(No|Buys Computer) = 3/9 P(Yes|Buys Computer) = 6/9 P(Fair|Buys Computer) = 6/9 P(Excellent|Buys Computer) = 3/9

Conditional Probabilities of Does not buy computer: P(Young|Does not buy computer) = 3/5 P(Middle|Does not buy computer) = 0/5 P(Old|Does not buy computer) = 2/5 P(High|Does not buy computer) = 2/5 P(Medium|Does not buy computer) = 2/5 P(Low|Does not buy computer) = 3/5 P(No|Does not buy computer) = 4/5 P(Yes|Does not buy computer) = 1/5 P(Fair|Does not buy computer) = 3/5 P(Excellent|Does not buy computer) = 2/5

Now, let's calculate the posterior probabilities of buying and not buying a computer given the attributes for the new instance:

P(Buys Computer|Old, High, Excellent) = (P(Old|Buys Computer) \* P(High|Buys Computer) \* P(Excellent|Buys Computer) \* P(Buys Computer)) / (P(Old) \* P(High) \* P(Excellent))

= (5/9 \* 3/9 \* 3/9 \* 9/14) / ((9/14) \* (7/14) \* (6/14))

= 0.175

P(Does not buy computer|Old, High, Excellent) = (P(Old|Does not buy computer) \* P(High|Does not buy computer) \* P(Excellent|Does not buy computer) \* P(Does not buy computer)) / (P(Old) \* P(High) \* P(Excellent))

= (2/5 \* 2/5 \* 2/5 \* 5/14) / ((9/14) \* (7/14) \* (6/14))

= 0.0238

Since P(Buys Computer|Old, High, Excellent) > P(Does not buy computer|Old, High, Excellent), we predict that the old student with high income and excellent credit rating will buy a computer.



35. Identify the first splitting attribute for decision tree with the following dataset using ID3 algorithm: Field Experience Hired IT Coding No IT Coding No IT Administration Yes IT Administration Yes Business Coding Yes Business Coding Yes Business Administration No Business Administration No

ChatGPT

To identify the first splitting attribute, we need to calculate the information gain for each attribute. The attribute with the highest information gain will be selected as the first splitting attribute.

First, we calculate the entropy of the entire dataset:

* Total number of instances (n) = 8
* Number of positive instances (p) = 4
* Number of negative instances (n-p) = 4
* Entropy of the dataset (H(S)) = -(p/n)log2(p/n) - ((n-p)/n)log2((n-p)/n) = -0.5log2(0.5) - 0.5log2(0.5) = 1

Next, we calculate the information gain for each attribute:

1. Field:

* There are 2 possible values: IT and Business
* For IT, there are 2 positive instances and 2 negative instances
* For Business, there are 2 positive instances and 2 negative instances
* Entropy of IT (H(S|IT)) = -(2/4)log2(2/4) - (2/4)log2(2/4) = 1
* Entropy of Business (H(S|Business)) = -(2/4)log2(2/4) - (2/4)log2(2/4) = 1
* Information gain (IG) of Field = H(S) - ((4/8)H(S|IT) + (4/8)H(S|Business)) = 1 - 1 - 1 = 0

1. Experience:

* There are 2 possible values: Coding and Administration
* For Coding, there are 3 positive instances and 1 negative instance
* For Administration, there is 1 positive instance and 3 negative instances
* Entropy of Coding (H(S|Coding)) = -(3/4)log2(3/4) - (1/4)log2(1/4) = 0.81
* Entropy of Administration (H(S|Administration)) = -(1/4)log2(1/4) - (3/4)log2(3/4) = 0.81
* Information gain (IG) of Experience = H(S) - ((4/8)H(S|Coding) + (4/8)H(S|Administration)) = 1 - 0.81 - 0.81 = 0.38

1. Hired:

* There are 2 possible values: Yes and No
* For Yes, there are 4 positive instances and 0 negative instances
* For No, there are 0 positive instances and 4 negative instances
* Entropy of Yes (H(S|Yes)) = 0
* Entropy of No (H(S|No)) = 0
* Information gain (IG) of Hired = H(S) - ((4/8)H(S|Yes) + (4/8)H(S|No)) = 1 - 0 - 0 = 1

Therefore, the first splitting attribute for the decision tree is Hired, as it has the highest information gain of 1.



36. Suppose that the probability of five events are P(first) = 0.5, P(second) = P(third) = P(fourth) = P(fifth) = 0.125. Calculate its entropy.

ChatGPT

The entropy of a probability distribution with events E1, E2, ..., En and probabilities P(E1), P(E2), ..., P(En) is defined as:

H(E) = - Σ P(Ei) \* log2(P(Ei))

where log2 is the binary logarithm.

Using this formula, we can calculate the entropy of the given probability distribution as:

H = - (0.5 \* log2(0.5) + 0.125 \* log2(0.125) + 0.125 \* log2(0.125) + 0.125 \* log2(0.125) + 0.125 \* log2(0.125))

H = - (0.5 \* -1 + 0.125 \* -3 + 0.125 \* -3 + 0.125 \* -3 + 0.125 \* -3)

H = - (-0.5 + 0.375 + 0.375 + 0.375 + 0.375)

H = - 1.5

H = 1.5

Therefore, the entropy of the given probability distribution is 1.5.



37. Consider the training data in the following table where Play is a class attribute. Humidity Sunny Windy Play L N S N H N W Y H N W N H Y S Y H N W Y L Y S N Build a Naïve Bayes Classifier using the above data. Estimate the class Label for day where (Humidity = L, Sunny = N, Wind = W) using the above classifier.

ChatGPT

To build a Naïve Bayes Classifier, we need to calculate the prior probabilities and conditional probabilities.

The prior probability of Play = Yes and Play = No are: P(Play = Yes) = 3/6 = 0.5 P(Play = No) = 3/6 = 0.5

Now, let's calculate the conditional probabilities for each attribute:

* Humidity: P(Humidity = L | Play = Yes) = 1/3 = 0.333 P(Humidity = H | Play = Yes) = 2/3 = 0.667 P(Humidity = L | Play = No) = 2/3 = 0.667 P(Humidity = H | Play = No) = 1/3 = 0.333
* Sunny: P(Sunny = N | Play = Yes) = 2/3 = 0.667 P(Sunny = Y | Play = Yes) = 1/3 = 0.333 P(Sunny = N | Play = No) = 2/3 = 0.667 P(Sunny = Y | Play = No) = 1/3 = 0.333
* Windy: P(Windy = N | Play = Yes) = 1/3 = 0.333 P(Windy = Y | Play = Yes) = 2/3 = 0.667 P(Windy = N | Play = No) = 2/3 = 0.667 P(Windy = Y | Play = No) = 1/3 = 0.333

Now, let's calculate the probabilities for the new instance (Humidity = L, Sunny = N, Windy = W) for both classes:

P(Humidity = L | Play = Yes) \* P(Sunny = N | Play = Yes) \* P(Windy = W | Play = Yes) \* P(Play = Yes) = 0.333 \* 0.667 \* 0.667 \* 0.5 = 0.074

P(Humidity = L | Play = No) \* P(Sunny = N | Play = No) \* P(Windy = W | Play = No) \* P(Play = No) = 0.667 \* 0.667 \* 0.333 \* 0.5 = 0.111

Since the probability is higher for Play = No, the predicted class label for the new instance is No.



44. Train a neural network for the following data with X1 and X2 as inputs and Y as output. X1 X2 Y (Output) 0 0 1 0 1 0 1 0 0 1 1 0

ChatGPT

We will train a neural network with one hidden layer and one output layer. We will use the sigmoid activation function for both layers.

First, we need to initialize the weights and biases. We will use random values between -1 and 1.

Let's define the sigmoid function:

sigmoid(x) = 1 / (1 + exp(-x))

where exp() is the exponential function.

Now, we will define the neural network architecture and implement the forward propagation algorithm to calculate the output of the network for a given input.

Let's denote the input layer as X, the hidden layer as H, and the output layer as Y.

We will have 2 input neurons, 2 hidden neurons, and 1 output neuron.

We will denote the weights between the input layer and the hidden layer as W1, and the biases as b1. Similarly, we will denote the weights between the hidden layer and the output layer as W2, and the biases as b2.

The output of the hidden layer is calculated as:

H = sigmoid(X \* W1 + b1)

where \* denotes matrix multiplication.

The output of the output layer is calculated as:

Y = sigmoid(H \* W2 + b2)

Let's initialize the weights and biases:

import numpy as np

np.random.seed(0)

W1 = np.random.uniform(-1, 1, (2, 2)) b1 = np.random.uniform(-1, 1, (1, 2)) W2 = np.random.uniform(-1, 1, (2, 1)) b2 = np.random.uniform(-1, 1, (1, 1))

Now, let's define the sigmoid function:

def sigmoid(x): return 1 / (1 + np.exp(-x))

Let's implement the forward propagation algorithm:

def forward\_propagation(X, W1, b1, W2, b2): H = sigmoid(np.dot(X, W1) + b1) Y = sigmoid(np.dot(H, W2) + b2) return Y

Now, let's train the neural network using backpropagation.

We will use the mean squared error as the loss function and the gradient descent algorithm for optimization.

The steps for backpropagation are as follows:

1. Calculate the error between the predicted output and the actual output.
2. Calculate the gradients of the weights and biases in the output layer.
3. Calculate the gradients of the weights and biases in the hidden layer.
4. Update the weights and biases using the gradients and the learning rate.

We will repeat these steps for a certain number of epochs.

Let's define the backpropagation algorithm:

def backpropagation(X, Y, W1, b1, W2, b2, learning\_rate, epochs): for i in range(epochs): # Forward propagation H = sigmoid(np.dot(X, W1) + b1) Y\_pred = sigmoid(np.dot(H, W2) + b2)

pythonCopy code

# Backward propagation error = Y - Y\_pred d\_Y\_pred = error \* Y\_pred \* (1 - Y\_pred) d\_H = np.dot(d\_Y\_pred, W2.T) \* H \* (1 - H) d\_W2 = np.dot(H.T, d\_Y\_pred) d\_b2 = np.sum(d\_Y\_pred, axis=0, keepdims=True) d\_W1 = np.dot(X.T, d\_H) d\_b1 = np.sum(d\_H, axis=0, keepdims=True) # Update weights and biases W2 += learning\_rate \* d\_W2 b2 += learning\_rate \* d\_b2 W1 += learning\_rate \* d\_W1



49. Calculate the output y of a three inputs neuron with bias. The input feature vector is (𝑥1,𝑥2,𝑥3) = (0.8,0.6,0.4) and weight values are [𝑤1,𝑤2,𝑤3,𝑏]=[0.2,0.1,−0.3,0.35]. Use binary Sigmoid function as activation function.

ChatGPT

The output of a neuron with bias and sigmoid activation function can be calculated as follows:

z = w1*x1 + w2*x2 + w3\*x3 + b

y = sigmoid(z)

where w1, w2, w3 are the weights, b is the bias, x1, x2, x3 are the input values, sigmoid is the activation function and y is the output value.

Given x1 = 0.8, x2 = 0.6, x3 = 0.4, w1 = 0.2, w2 = 0.1, w3 = -0.3, and b = 0.35, we have

z = w1*x1 + w2*x2 + w3*x3 + b = 0.2*0.8 + 0.1*0.6 - 0.3*0.4 + 0.35 = 0.31

y = sigmoid(z) = 1 / (1 + exp(-z)) = 1 / (1 + exp(-0.31)) = 0.5772

Therefore, the output of the neuron with the given input and weight values is y = 0.5772.



60. Use K-means clustering to cluster the following data into two groups: {2, 4, 10, 12, 3, 20, 30, 11, 25} Assume cluster centroid are m1 = 2 and m2 =4. The distance function used is Euclidean distance.

ChatGPT

Initial centroids: m1=2, m2=4

Iteration 1: Cluster 1: {2, 3} Cluster 2: {4, 10, 11, 12, 20, 25, 30} New centroids: m1 = 2.5, m2 = 16.2857

Iteration 2: Cluster 1: {2, 3, 4, 10, 11, 12} Cluster 2: {20, 25, 30} New centroids: m1 = 7.0, m2 = 25.0

Iteration 3: Cluster 1: {2, 3, 4, 10, 11, 12} Cluster 2: {20, 25, 30} New centroids: m1 = 7.0, m2 = 25.0

Final clusters: Cluster 1: {2, 3, 4, 10, 11, 12} Cluster 2: {20, 25, 30}

Therefore, the two clusters are: Cluster 1: {2, 3, 4, 10, 11, 12} Cluster 2: {20, 25, 30}



64. Assume a total of 1000 patients are tested for influenza; 900 are found to be healthy, while 100 are found to be sick. A test resulted in 60 being positive and 40 being negative for the sick persons. The same test was positive for 120 and negative for 780 in healthy adults. Construct a confusion matrix for the data and determine precision and recall.

ChatGPT

A confusion matrix for the given data is as follows:

|  | **Actual Positive** | **Actual Negative** |
| --- | --- | --- |
| Predicted Positive | 60 | 120 |
| Predicted Negative | 40 | 780 |

Precision is the proportion of correctly predicted positive cases out of total predicted positive cases. It is calculated as TP / (TP + FP), where TP is the true positive and FP is the false positive. In this case, precision is:

Precision = 60 / (60 + 120) = 0.333

Recall is the proportion of correctly predicted positive cases out of actual positive cases. It is calculated as TP / (TP + FN), where FN is the false negative. In this case, recall is:

Recall = 60 / (60 + 40) = 0.6