Feature selection and accuracy analysis for the Mnist dataset

For the mnist dataset classification we have utilized three feature extraction techniques. These are:

1. **Raw pixel intensity feature:** extraction technique where each pixel value of the image is used as a feature.
2. **Histogram of Oriented Gradients(HOG) feature:** HOG feature extraction method was selected for the MNIST image classification task because it is a simple and effective way to capture important visual information in an image. HOG works by computing the orientation of image gradients and grouping them into histograms, which can capture information about the edges and shapes in an image, making it particularly effective at object recognition tasks. Additionally, HOG is relatively invariant to changes in lighting and contrast, which makes it a robust feature extraction method that can be used in a wide range of lighting conditions and environments. Finally, HOG is computationally efficient and can be computed quickly, even for large images or datasets, making it a practical choice for image classification task.
3. **Principal Component Analysis (PCA).** PCA helps in reducing the dimensionality of image data while preserving the most important information. PCA involves converting the images into a numerical representation, normalizing the data, computing the covariance matrix, performing eigenvalue decomposition or SVD, selecting the top eigenvectors (principal components), and transforming the original data into a lower-dimensional feature representation. These extracted features can then be used as input for various machine learning tasks such as image classification or object recognition. PCA allows for efficient representation of images, reducing noise and redundancy while capturing the most significant variations in the data.

Accuracy analysis on different smoothing values and each feature extraction method on mnist data.

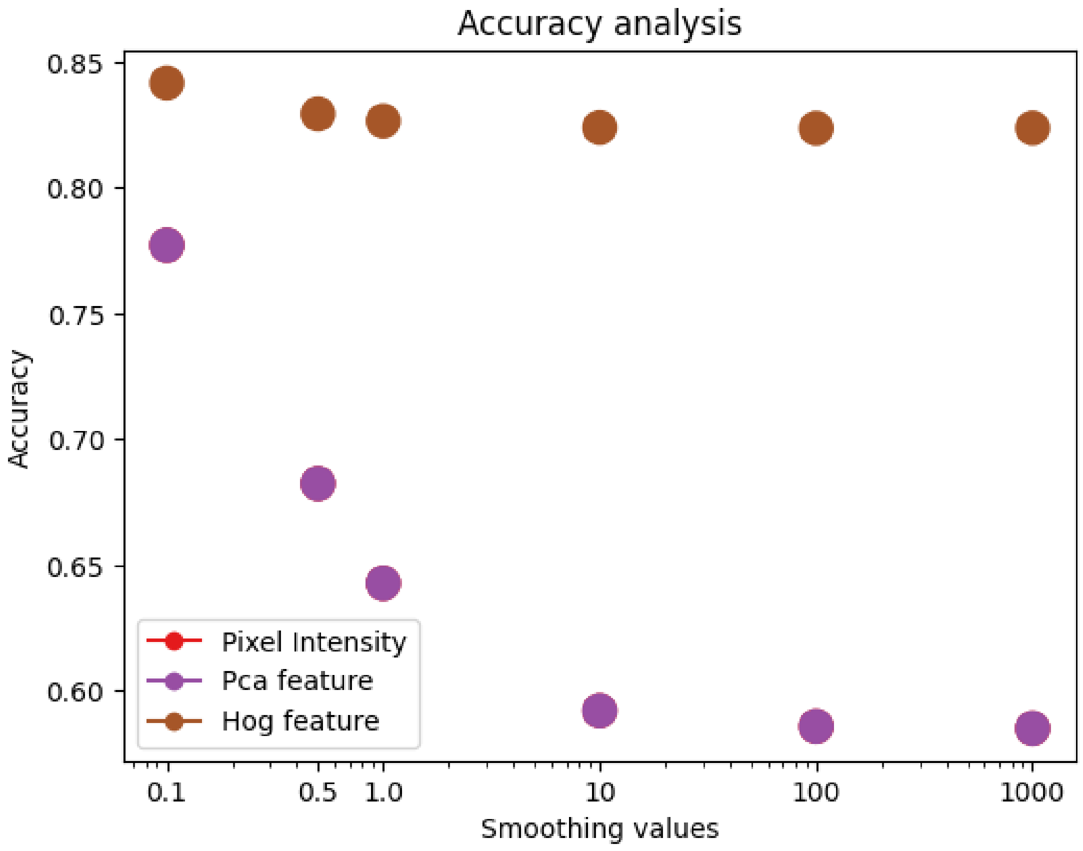


Fig1. Accuracy analysis of using naïve bayes algorithm on mnist dataset

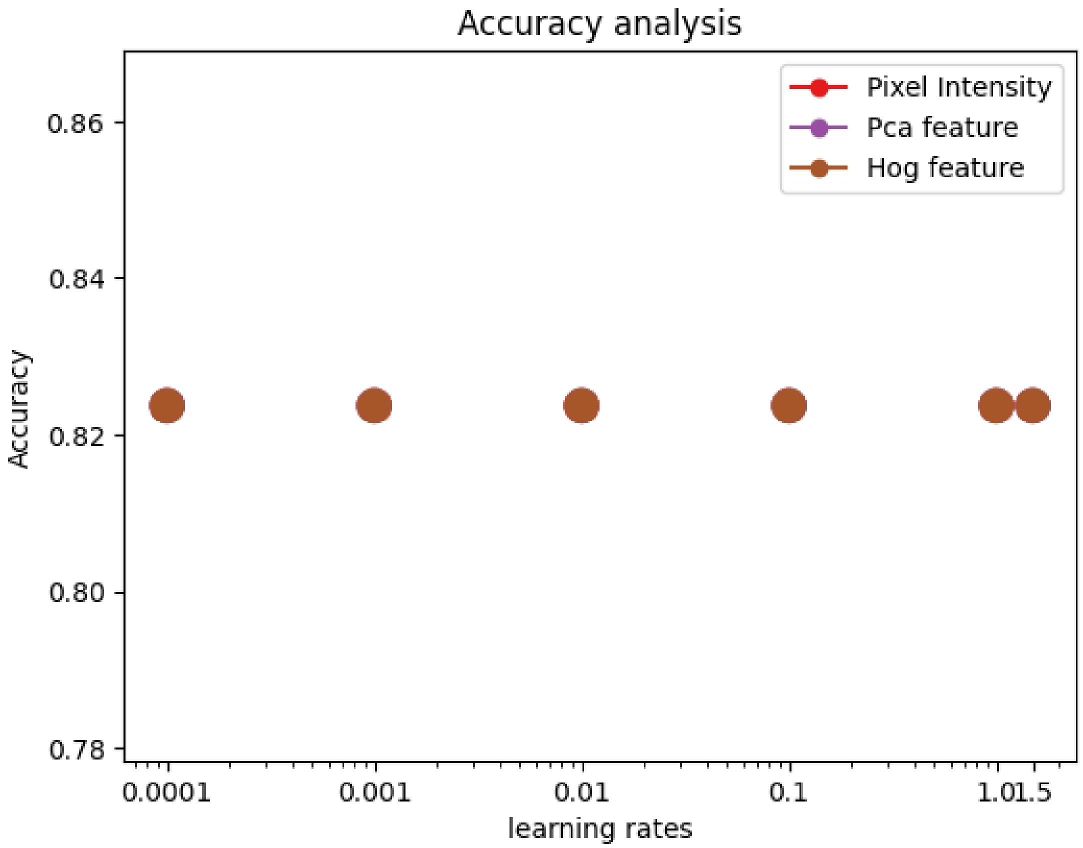


Fig2. Accuracy analysis of using logistic regression algorithm on mnist dataset

Feature selection and accuracy analysis for the BBC-News Data Set

Bag of Words :

This extraction method is a method which creates a vocabulary of unique words from the corpus and represents each document as a vector where each dimension corresponds to a word in the vocabulary.

We implement it because:

- It is simple and computationally efficient
- To capture and use the frequency of words
- It is common in text classification

Term Frequency-Inverse Document Frequency:

This method represents a text document as a vector and also assigns weights to words based on their importance.

We implement it because:

- It helps us to use importance words in the document
- It is useful in ranking documents based on their relevance
- It enables us to identify words that are unique to a particular document

Inverse Document Frequency:

The third method we use for the feature extraction is the inverse document only i.e., TF_IDF without TF.

We implement it because:

- We want to see the effect of TF in IDF i.e., the importance of frequency
- TF alone gives high weights to common terms that appear frequently to avoid this we use it
- And as the third extraction method to full the instruction

The graph comparison for the features in each algorithm is here below

The first graph is for a logistic regression algorithm. It shows the performance for each feature at different learning rates.

The second graph is for the naive bayes algorithm. It shows the performance for each feature at different LaPlace values.

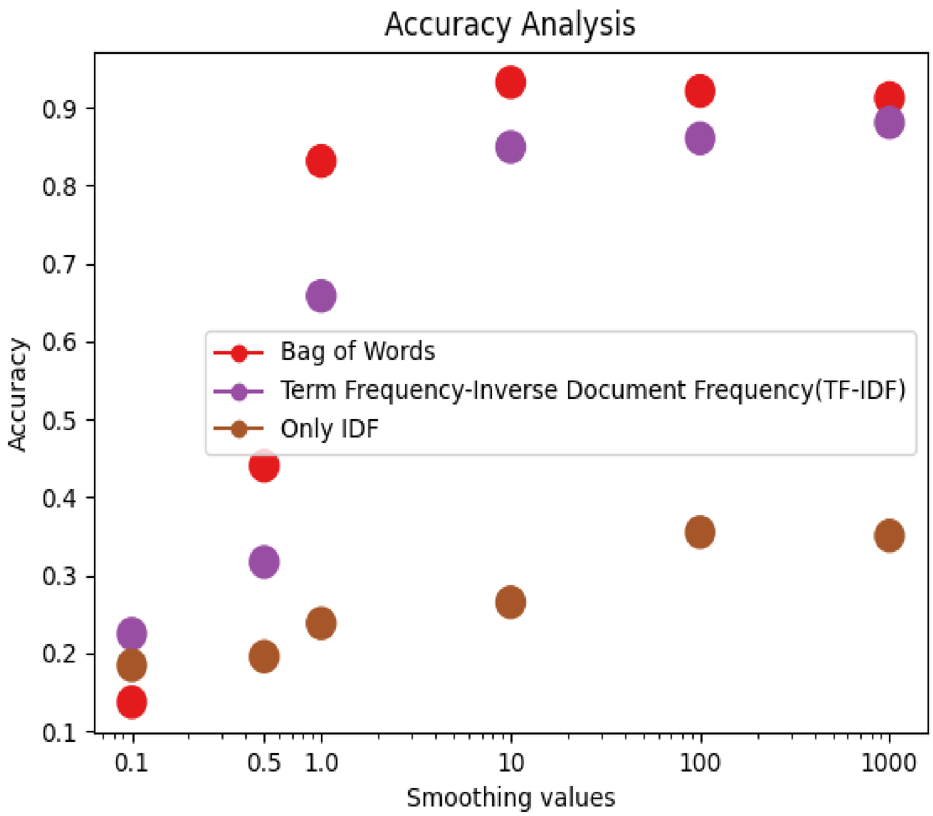


Fig3. Accuracy analysis of using logistic regression on BBC dataset

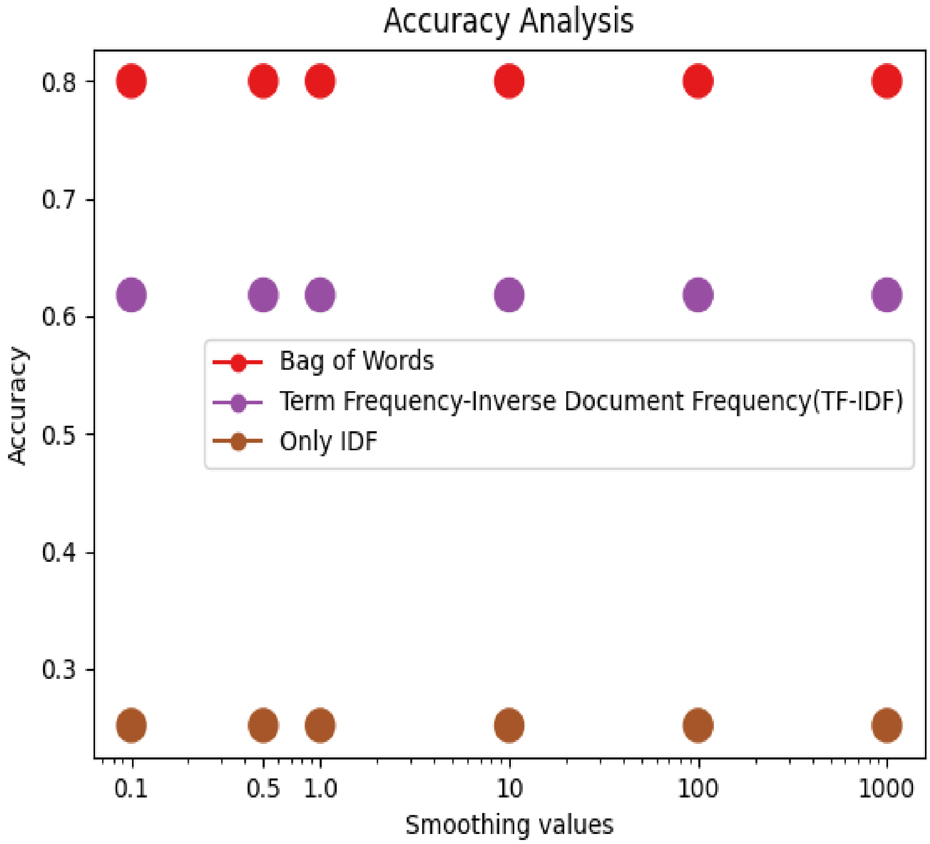


Fig4. Accuracy analysis of using naïve bayes algorithm on BBC dataset

Part II - Probability Questions

1. **Suppose that $P(A) = 0.4$, $P(B) = 0.3$ and $P((A \cup B)C) = 0.42$. Are A and B independent?**

We know that A and B are independent if and only if

- **$P (A \cap B) = p (A) \text{ x } P (B)$**
- **We have $p (A \cup B)' = 1 - p (A \cup B) = 0.42$**
- **So we have $p (A \cup B) = 0.58$**
- **$p (A \cup B) = p (A) + p(B) - p (A \cap B)$**
- **$0.58 = 0.4 +0.3- p (A \cap B)$**
- **$P (A \cap B) = 0.7 - 0.58 = 0.12$**
- **And $p (A) \text{ x } p (B) = 0.4 \text{ x } 0.3 = 0.12$**
- **So here $p (A \cap B) = p (A) \text{ x } p (B)$**

2. **Two dice are rolled. A = ‘sum of two dice equals 3’ B = ‘sum of two dice equals 7’ C = ‘at least one of the dice shows a 1’**
   a. **What is P(A|C)?**
   b. **What is P(B|C)?**
   c. **Are A and C independent? What about B and C?**

No of outcomes in event C will be

11 = {(1,1),(1,2),(1,3),,..(1,6), (2,1), (2,2),.....(2,6)

No of Common outcomes in **A and C = {(1,2),(2,1)}**

No of Common outcomes in **B and C = {(1,6),(6,1)}**

a) P(A|C) = P(A and C)/P(C)

- **P(A|C) = 2/11**

b) P(B|C) = P(B and C)/P(C)

- **P(B|C) = 2/11**

c) Both A and C or B and C are not independent because in both cases P(A and C) is not equal to P(A)*P(C)

3. Let C and D be two events with P(C) = 0.25, P(D) = 0.45, and P(C ∩ D) = 0.1. What is P(Cc ∩ D)?

- **C and D** are two events.
- **P(C)** represents the probability of event C occurring, which is 0.25.
- **P(D)** represents the probability of event D occurring, which is 0.45.
- **P(C ∩ D)** represents the probability of both events C and D occurring together, which is 0.1.
- Cc represents the complement of event C, which means the event of C not occurring.
- **P(Cc ∩ D)** represents the probability of event C not occurring and event D occurring.

Now, let's use the given information to find **P(Cc ∩ D).**

We know that the probability of the union of two events is given by:

- **P(C ∪ D) = P(C) + P(D) - P(C ∩ D)**

We also know that the probability of the complement of an event is given by:

- **P(Cc) = 1 - P(C)**

Now, we can use the formula for the probability of the intersection of two events:

- P(Cc ∩ D) = P(D) - P(C ∩ D)

We are given **P(D)** = 0.45 and **P(C ∩ D)** = 0.1. Plugging these values into the formula, we get:

P(Cc ∩ D) = 0.45 - 0.1

P(Cc ∩ D) = 0.35

So, the probability of event C not occurring and event D occurring is 0.35.

4. There are 3 arrangements of the word DAD, namely DAD, ADD, and DDA. How many arrangements are there of the word PROBABILITY?

In word PROBABILITY, there are:

1P, 1R 1O, 2B, 1A 2I 1L, 1T, 1Y

Total letters, **n = 11**

p1 = 2

p2 = 2

Total Arrangements **= n!/(p1! p2!)**

= 11!/(2!2!)

= 39,916,800 / 4

= 9979200

5. Let A and B be two events. Suppose the probability that neither A or B occurs is 2/3. What is the probability that one or both occur?

Given,

1. Let A and B be two events

2. Probability that neither A nor B Occurs is ⅔

Solution

Therefore, Probability that one or both Occurs is obtained below:

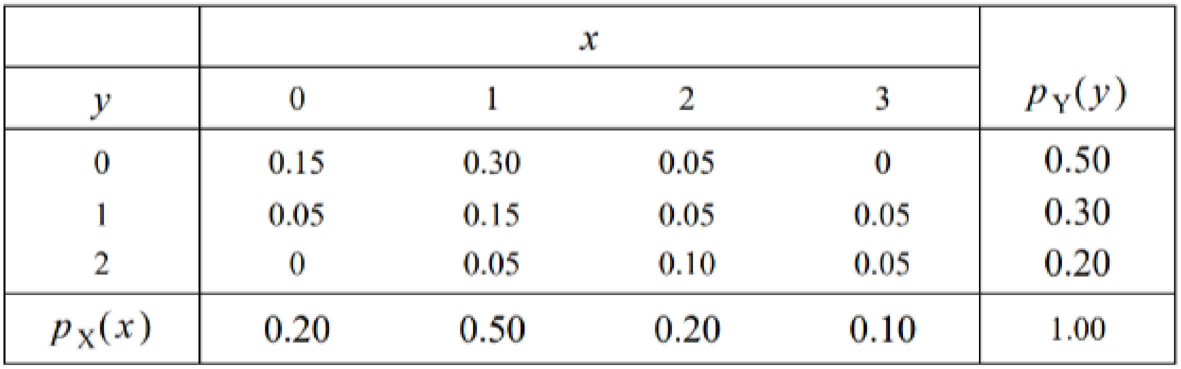
P(AUB) = 1 - P(AUB)'

= 1 - P(A n B)

= 1 - 2/3

P(AUB) = 1/3

6. Let X denote the number of times a photocopy machine will malfunction: 0, 1, 2, or 3 times, on any given month. Let Y denote the number of times a technician is called on an emergency call. The joint p.m.f. p(x, y) is presented in the table below:



	x				
y	0	1	2	3	$p_Y(y)$
0	0.15	0.30	0.05	0	0.50
1	0.05	0.15	0.05	0.05	0.30
2	0	0.05	0.10	0.05	0.20
$p_X(x)$	0.20	0.50	0.20	0.10	1.00

- Find the probability P(Y > X).
- Find p X(x), the marginal p.m.f. of X.
- Find p Y(y), the marginal p.m.f. of Y.
- Are X and Y independent?

SOLUTION

• X = number of times a photocopy machine will malfunction.

• Y = number of times a technician is called on an emergency call.

a) **P(Y>X) :-**

=> P(Y>X) = P(0,1) + P(0,2) + P(1,2)

=> P(Y>X) = 0.05 + 0 + 0.05

= [P(Y>X) = 0.1]

For ε(x) and ε(4) ÷

X	0	1	2	3	Total

f(x)	**0.20**	**0.50**	**0.20**	**0.10**	**1**

Y	0	1	2	Total
f(y)	0.5	0.3	0.2	1

- **E(X) = $\sum_{\square=\square}^{\square}$ X i Pi(x)**

= 0 + 0.5+2(0·2)+3(0.1)

= 0.5 +0.4+0.3

⇒E(X) = 1.2

E(y) = $\sum_{\square=\square}^{\square}$ yi pj (0)

= 0 = 1 (0.3) + 2 (0.2)

⇒ E(y) = 0.7

Fox (x4)=

E(xy) = $\sum_{\square-\square}^{\square}$ $\sum_{\square=\square}^{\square}$ Xiyi Pij (xy)

x	0	1	2	3	Total

y	0	1	2	0	1	2	0	1	2	0	1	2	12
Pij	0.15	0.05	0	0.3	0.15	0.05	0.05	0.05	0.1	0	0.05	0.05	0.80
xy	0	0	0	0	1	2	0	2	4	0	3	6	18

xy	0	1	2	3	4	5	6	Total
f(xy)	0.55	0.15	0.05	0.05	0.05	0.1	0.05	1

$E(xy) = 0 + 0.15 + 0.1 + 0.15 + 0.2 + 0.5 + 0.3$

$$\Rightarrow E\ (xy) = 1.40$$

If 'X' and 'Y' are independent:-

- **For independency E(XY)**
- **E(XY) = E(X) × E(Y)**
- **E(XY) = 1.2 × 0.7**
- **E(XY) = 0.84**
- **= 1.40 ≠ 0.84**

So, they are not independent.

Cov(XY) :-

- **E(XY) - E(X) × E(Y)**

- **1.4 - 1.2 × 0.7**
- **1.4 - 0.84**
- **= [Cov(XY) = 0.56]**

7. The following are data points with their labels:

(1, 2, 3, 4), 1

(5, 6, 7, 8), 0

(9, 10, 11, 12), 1

The following are the randomly set weights:

w1 = 0.1

w2 = 0.2

w3 = -0.1

w4 = 0.0

Task: make three learning updates with a learning rate of 0.1 using the data points. The updates should be based on both the Perceptron and the logistic regression. Compare the two results.

To perform the requested updates for both the Perceptron and logistic regression models, we'll first need to define some functions. For the Perceptron, we have the following update rule:

- w = w + learning_rate * (y - y_pred) * x

For logistic regression, the update rule is:

- w = w + learning_rate * (y - y_prob) * x

where y_prob is the predicted probability of the positive class (1), calculated using the sigmoid function:

- y_prob = 1 / (1 + exp(-z)), where z = w1*x1 + w2*x2 + w3*x3 + w4*x4

Now, let's perform the updates.

Perceptron

Initial weights: w1 = 0.1, w2 = 0.2, w3 = -0.1, w4 = 0.0

Learning rate: 0.1

Update 1:

- Data point: (1, 2, 3, 4), label: 1

- z = 0.11 + 0.22 + (-0.1)3 + 0.04 = 0.1
- y_pred = 1 if z >= 0 else 0 = 1
- No update since y_pred == y

Update 2:

Data point: (5, 6, 7, 8), label: 0

- z = 0.15 + 0.26 + (-0.1)7 + 0.08 = 1.1
- y_pred = 1 if z >= 0 else 0 = 1

Update weights:

- w1 = 0.1 - 0.1 * (1 - 0) * 5 = 0.1 - 0.5 = -0.4
- w2 = 0.2 - 0.1 * (1 - 0) * 6 = 0.2 - 0.6 = -0.4
- w3 = -0.1 - 0.1 * (1 - 0) * 7 = -0.1 - 0.7 = -0.8
- w4 = 0.0 - 0.1 * (1 - 0) * 8 = 0.0 - 0.8 = -0.8

Update 3:

Data point: (9, 10, 11, 12), label: 1

- z = (-0.4)*9 + (-0.4)*10 + (-0.8)*11 + (-0.8)*12 = -32.8
- y_pred = 1 if z >= 0 else 0 = 0

Update weights:

- w1 = -0.4 + 0.1 * (1 - 0) * 9 = -0.4 + 0.9 = 0.5

- w2 = -0.4 + 0.1 * (1 - 0) * 10 = -0.4 + 1.0 = 0.6
- w3 = -0.8 + 0.1 * (1 - 0) * 11 = -0.8 + 1.1 = 0.3
- w4 = -0.8 + 0.1 * (1 - 0) * 12 = -0.8 + 1.2 = 0.4

Perceptron final weights: w1 = 0.5, w2 = 0.6, w3 = 0.3, w4 = 0.4

Logistic Regression

Initial weights: w1 = 0.1, w2 = 0.2, w3 = -0.1, w4 = 0.0

Learning rate: 0.1

We'll use the sigmoid function to calculate the predicted probabilities:

Update 1:

Data point: (1, 2, 3, 4), label: 1

- z = 0.11 + 0.22 + (-0.1)3 + 0.04 = 0.1
- y_prob = sigmoid(z) = 0.525

Update weights:

- w1 = 0.1 + 0.1 * (1 - 0.525) * 1 = 0.1475
- w2 = 0.2 + 0.1 * (1 - 0.525) * 2 = 0.295
- w3 = -0.1 + 0.1 * (1- 0.525) * 3 = -0.0525
- w4 = 0.0 + 0.1 * (1 - 0.525) * 4 = 0.19

Update 2:

Data point: (5, 6, 7, 8), label: 0

- z = 0.14755 + 0.2956 + (-0.0525)7 + 0.198 = 2.4175
- y_prob = sigmoid(z) = 0.918

Update weights:

- w1 = 0.1475 - 0.1 * (0.918 - 0) * 5 = -0.312
- w2 = 0.295 - 0.1 * (0.918 - 0) * 6 = -0.251
- w3 = -0.0525 - 0.1 * (0.918 - 0) * 7 = -0.694
- w4 = 0.19 - 0.1 * (0.918 - 0) * 8 = -0.536

Update 3:

Data point: (9, 10, 11, 12), label: 1

- z = (-0.312)*9 + (-0.251)*10 + (-0.694)*11 + (-0.536)*12 = -22.645
- y_prob = sigmoid(z) = 1.54e-10

Update weights:

- w1 = -0.312 + 0.1 * (1 - 1.54e-10) * 9 = 0.588
- w2 = -0.251 + 0.1 * (1 - 1.54e-10) * 10 = 0.749
- w3 = -0.694 + 0.1 * (1 - 1.54e-10) * 11 = 0.416
- w4 = -0.536 + 0.1 * (1 - 1.54e-10) * 12 = 0.664

Logistic regression final weights: w1 = 0.588, w2 = 0.749, w3 = 0.416, w4 = 0.664

Comparison

Perceptron final weights: w1 = 0.5, w2 = 0.6, w3 = 0.3, w4 = 0.4

Logistic regression final weights: w1 = 0.588, w2 = 0.749, w3 = 0.416, w4 = 0.664

The weights after three updates are different between the Perceptron and logistic regression models. The differences stem from the learning mechanisms of the two methods. The Perceptron uses a step function for predictions and updates weights based on the difference between the predicted and true labels. In contrast, logistic regression uses the sigmoid function to output probabilities and updates weights based on the difference between the predicted probabilities and true labels. As a result, logistic regression can be more sensitive to small changes in the input data and may converge to a different solution than the Perceptron.