

CS 4063 – Natural Language Processing

Lecture Notes – week 5 Lec 2

Muhammad Hannan Farooq

Text classifiers

- Naïve Bayes
- Logistic regression

Naive Bayes

- **Probabilistic Classifier:** Naive Bayes is based on applying Bayes' theorem with the "naive" assumption of conditional independence between every pair of features given the class label.

- Bayes' Theorem:

$$P(C|X) = \frac{P(X|C)P(C)}{P(X)}$$

Where:

- $P(C|X)$ is the posterior probability of class C given features X .
- $P(X|C)$ is the likelihood of features X given class C .
- $P(C)$ is the prior probability of class C .
- $P(X)$ is the probability of features X .
- **Naive Assumption:** Assumes that the features (in NLP, words or tokens) are conditionally independent given the class label.

Naive Bayes

- **Multinomial Naive Bayes:** Suitable for discrete data (e.g., word counts in documents).
- **Bernoulli Naive Bayes:** Suitable for binary/boolean features (e.g., word occurrence as binary features).
- **Gaussian Naive Bayes:** Suitable for continuous data (not commonly used in NLP).

Application in NLP

- **Features:** Words or tokens extracted from text documents.
- **Classes:** Categories like "spam" vs. "ham", "positive" vs. "negative" sentiment, or topics.

Example

Training Data:

- Positive Reviews (Class = Positive):
 1. "I love this movie"
 2. "Fantastic and exciting movie"
 - Negative Reviews (Class = Negative):
 1. "This movie was boring"
 2. "Terrible and dull experience"
-

New Review to Classify:

- New Review: "exciting movie"

Example

Step 1: Calculate Prior Probabilities $P(C)$:

- $P(\text{Positive})$: There are 2 positive reviews out of 4 total reviews, so:

$$P(\text{Positive}) = \frac{2}{4} = 0.5$$

- $P(\text{Negative})$: There are also 2 negative reviews, so:

$$P(\text{Negative}) = \frac{2}{4} = 0.5$$

Example

- **Negative Class:**
 - Vocabulary in negative reviews: ["this", "movie", "was", "boring", "terrible", "and", "dull", "experience"]
 - Total words in negative reviews = 8

$$P(\text{exciting}|\text{Negative}) = \frac{0}{8}, \quad P(\text{movie}|\text{Negative}) = \frac{1}{8}$$

So, the likelihood for the negative class is:

$$P(X = \text{exciting movie}|\text{Negative}) = P(\text{exciting}|\text{Negative}) \times P(\text{movie}|\text{Negative}) = 0 \times \frac{1}{8} = 0$$

Step 2: Calculate Likelihood $P(X|C)$:

For each class, we need to calculate the likelihood of the words in the new review ("exciting movie") appearing given that the review is positive or negative.

- **Positive Class:**
 - Vocabulary in positive reviews: ["love", "this", "movie", "fantastic", "and", "exciting"]
 - Total words in positive reviews = 6

$$P(\text{exciting}|\text{Positive}) = \frac{1}{6}, \quad P(\text{movie}|\text{Positive}) = \frac{2}{6}$$

So, the likelihood for the positive class is:

$$P(X = \text{exciting movie}|\text{Positive}) = P(\text{exciting}|\text{Positive}) \times P(\text{movie}|\text{Positive}) = \frac{1}{6} \times \frac{2}{6} = \frac{2}{36}$$

Example

Step 3: Calculate Posterior Probability $P(C|X)$:

Now we can calculate the posterior probability for each class.

- For Positive Class:

$$P(\text{Positive}|X = \text{exciting movie}) \propto P(\text{Positive}) \times P(X|\text{Positive}) = 0.5 \times \frac{2}{36} = \frac{1}{36}$$

- For Negative Class:

$$P(\text{Negative}|X = \text{exciting movie}) \propto P(\text{Negative}) \times P(X|\text{Negative}) = 0.5 \times 0 = 0$$

Example

Step 4: Classification Decision

Since $P(\text{Positive}|X = \text{exciting movie}) > P(\text{Negative}|X = \text{exciting movie})$, we classify the new review as Positive.

Complete Example w.r.t NLP

Detailed Example: Sentiment Analysis Using Multinomial Naive Bayes

Let's apply the Multinomial Naive Bayes classifier to a sentiment analysis task on a simplified movie review dataset.

Dataset

We will use a small dataset of movie reviews labeled as positive or negative.

Review ID	Review Text	Sentiment Label
1	"I love this movie. It's fantastic and exciting."	Positive (Pos)
2	"This film was terrible and boring."	Negative (Neg)
3	"What an excellent and thrilling experience!"	Positive (Pos)
4	"I did not enjoy this movie. It was dull."	Negative (Neg)

Step 1: Data Preparation and Preprocessing

1.1 Text Preprocessing

For each review:

- Lowercasing: Convert text to lowercase.
- Removing Punctuation: Strip punctuation marks.
- Tokenization: Split text into individual words (tokens).
- Stop Word Removal: Remove common words (e.g., "this", "and", "was").
- Stemming/Lemmatization: Reduce words to their base forms (optional for simplicity).

Processed Reviews:

1. Review 1:

- Original: "I love this movie. It's fantastic and exciting."
- Preprocessed Tokens: ["love", "movie", "fantastic", "exciting"]

2. Review 2:

- Original: "This film was terrible and boring."
- Preprocessed Tokens: ["film", "terrible", "boring"]

3. Review 3:

- Original: "What an excellent and thrilling experience!"
- Preprocessed Tokens: ["excellent", "thrilling", "experience"]

4. Review 4:

- Original: "I did not enjoy this movie. It was dull."
- Preprocessed Tokens: ["not", "enjoy", "movie", "dull"]

Step 2: Building Vocabulary and Calculating Frequencies

2.1 Building the Vocabulary

Create a set of all unique words from the preprocessed reviews.

Vocabulary:

1. love
2. movie
3. fantastic
4. exciting
5. film
6. terrible
7. boring
8. excellent
9. thrilling
10. experience
11. not
12. enjoy
13. dull



2.2 Calculating Word Frequencies per Class

Divide the reviews into their respective classes and count the frequency of each word in each class.

Class: Positive (Pos)

- Reviews: 1 and 3
- Tokens:
 - Review 1: ["love", "movie", "fantastic", "exciting"]
 - Review 3: ["excellent", "thrilling", "experience"]

• Word Counts:

Word	Count
love	1
movie	1
fantastic	1
exciting	1
excellent	1
thrilling	1
experience	1

- Total Word Count in Positive Class (N_{Pos}): 7

Class: Negative (Neg)

- Reviews: 2 and 4
- Tokens:
 - Review 2: ["film", "terrible", "boring"]
 - Review 4: ["not", "enjoy", "movie", "dull"]

• Word Counts:

Word	Count
film	1
terrible	1
boring	1
not	1
enjoy	1
movie	1
dull	1

- Total Word Count in Negative Class (N_{Neg}): 7

Step 3: Calculating Prior Probabilities

- Number of Documents (D): 4
- Number of Positive Documents (D_{Pos}): 2
- Number of Negative Documents (D_{Neg}): 2

Prior Probabilities:

- $P(\text{Pos}) = \frac{D_{\text{Pos}}}{D} = \frac{2}{4} = 0.5$
 - $P(\text{Neg}) = \frac{D_{\text{Neg}}}{D} = \frac{2}{4} = 0.5$
-

Step 4: Calculating Likelihoods with Laplace Smoothing

To prevent zero probabilities for words not seen in a class, we use Laplace smoothing (add-one smoothing).

Vocabulary Size (V)

- $V = 13$ (number of unique words)

4.1 Likelihood Calculation for Positive Class

For each word w in the vocabulary:

$$P(w|\text{Pos}) = \frac{\text{Count}(w, \text{Pos}) + 1}{N_{\text{Pos}} + V}$$

- $N_{\text{Pos}} + V = 7 + 13 = 20$

Compute $P(w|\text{Pos})$ for all words:

Word	Count in Pos	$P(w \text{Pos})$
love	1	$\frac{1+1}{20} = 0.10$
movie	1	$\frac{1+1}{20} = 0.10$
fantastic	1	0.10
exciting	1	0.10
excellent	1	0.10
thrilling	1	0.10
experience	1	0.10
film	0	$\frac{0+1}{20} = 0.05$
terrible	0	0.05
boring	0	0.05
not	0	0.05
enjoy	0	0.05
dull	0	0.05

4.2 Likelihood Calculation for Negative Class

Similarly for Negative class:

- $N_{\text{Neg}} + V = 7 + 13 = 20$

Compute $P(w|\text{Neg})$:

Word	Count in Neg	$P(w \text{Neg})$
love	0	$\frac{0+1}{20} = 0.05$
movie	1	$\frac{1+1}{20} = 0.10$
fantastic	0	0.05
exciting	0	0.05
excellent	0	0.05
thrilling	0	0.05
experience	0	0.05
film	1	0.10
terrible	1	0.10
boring	1	0.10
not	1	0.10
enjoy	1	0.10
dull	1	0.10

Step 5: Classification of a New Review

Suppose we have a new review to classify:

Review 5: "An exciting and thrilling movie with excellent acting."

5.1 Preprocessing

- Tokens: ["exciting", "thrilling", "movie", "excellent", "acting"]

Note: "acting" is a new word not in the vocabulary. We need to update our vocabulary and recompute probabilities, or we can ignore it for this example.

For simplicity, we'll assume "acting" is not in our vocabulary, and we'll handle it using Laplace smoothing.

Updated Vocabulary Size (V): 14

Update counts:

- For both classes, add "acting" with count 0, so $N_{\text{Pos}} + V = 7 + 14 = 21$, $N_{\text{Neg}} + V = 7 + 14 = 21$.

Recalculate Likelihoods including "acting":

- For "acting":
 - $P(\text{acting}|\text{Pos}) = \frac{0+1}{21} = \frac{1}{21} \approx 0.0476$
 - $P(\text{acting}|\text{Neg}) = \frac{0+1}{21} = 0.0476$

Other word probabilities need to be adjusted due to the change in $N_{\text{Pos}} + V$ and $N_{\text{Neg}} + V$. Let's recalculate them for the Positive class:

- Words with count 1 in Positive class:
 - $P(w|\text{Pos}) = \frac{1+1}{21} = \frac{2}{21} \approx 0.0952$
- Words with count 0 in Positive class:
 - $P(w|\text{Pos}) = \frac{0+1}{21} = \frac{1}{21} \approx 0.0476$

Similarly for Negative class.

Step 5: Classification of a New Review

Suppose we have a new review to classify:

Review 5: "An exciting and thrilling movie with excellent acting."

5.1 Preprocessing

- Tokens: ["exciting", "thrilling", "movie", "excellent", "acting"]

Note: "acting" is a new word not in the vocabulary. We need to update our vocabulary and recompute probabilities, or we can ignore it for this example.

For simplicity, we'll assume "acting" is not in our vocabulary, and we'll handle it using Laplace smoothing.

Updated Vocabulary Size (V): 14

Update counts:

- For both classes, add "acting" with count 0, so $N_{\text{Pos}} + V = 7 + 14 = 21$, $N_{\text{Neg}} + V = 7 + 14 = 21$.

Recalculate Likelihoods including "acting":

- For "acting":
 - $P(\text{acting}|\text{Pos}) = \frac{0+1}{21} = \frac{1}{21} \approx 0.0476$
 - $P(\text{acting}|\text{Neg}) = \frac{0+1}{21} = 0.0476$

Other word probabilities need to be adjusted due to the change in $N_{\text{Pos}} + V$ and $N_{\text{Neg}} + V$. Let's recalculate them for the Positive class:

- Words with count 1 in Positive class:
 - $P(w|\text{Pos}) = \frac{1+1}{21} = \frac{2}{21} \approx 0.0952$
- Words with count 0 in Positive class:
 - $P(w|\text{Pos}) = \frac{0+1}{21} = \frac{1}{21} \approx 0.0476$

Similarly for Negative class.

5.2 Calculating Log Posterior Probabilities

We need to compute:

$$\log P(\text{Pos}|X) \propto \log P(\text{Pos}) + \sum_{w \in X} \log P(w|\text{Pos})$$

$$\log P(\text{Neg}|X) \propto \log P(\text{Neg}) + \sum_{w \in X} \log P(w|\text{Neg})$$

Compute Log Priors:

- $\log P(\text{Pos}) = \log 0.5 = \ln 0.5 \approx -0.6931$
- $\log P(\text{Neg}) = -0.6931$

Compute Log Likelihoods for Each Class:

Words in the Review:

- "exciting"
- "thrilling"
- "movie"
- "excellent"
- "acting"



Positive Class Likelihoods:

| Word | $\log P(w|\text{Pos})$ | |-----| | exciting |
 $\log 0.0952 \approx -2.3514$ | | thrilling | -2.3514 | | movie | -2.3514 | | excellent | -2.3514 | | acting |
 $\log 0.0476 \approx -3.0445$ |

Sum of Log Likelihoods for Positive Class:

$$\text{Sum}_{\text{Pos}} = -2.3514 \times 4 + (-3.0445) = -9.4056 - 3.0445 = -12.4501$$

Total Log Posterior for Positive Class:

$$\log P(\text{Pos}|X) \propto -0.6931 + (-12.4501) = -13.1432$$

Negative Class Likelihoods:

| Word | $\log P(w|\text{Neg})$ | |-----| | exciting |
 $\log 0.0476 \approx -3.0445$ | | thrilling | -3.0445 | | movie | $\log 0.0952 \approx -2.3514$ | | excellent |
 -3.0445 | | acting | -3.0445 |

Sum of Log Likelihoods for Negative Class:

$$\text{Sum}_{\text{Neg}} = -3.0445 \times 4 + (-2.3514) = -12.178 - 2.3514 = -14.5294$$

Total Log Posterior for Negative Class:

$$\log P(\text{Neg}|X) \propto -0.6931 + (-14.5294) = -15.2225$$



Active

Step 6: Making the Classification Decision

- Compare Log Posterior Probabilities:
 - $\log P(\text{Pos}|X) = -13.1432$
 - $\log P(\text{Neg}|X) = -15.2225$
 - Decision:
 - Since $\log P(\text{Pos}|X) > \log P(\text{Neg}|X)$, we predict Positive sentiment for the review.
-

Logistic Regression

- A statistical method used for binary classification tasks—predicting the probability that a given input belongs to one of two classes.
- In the context of Natural Language Processing (NLP), logistic regression is commonly applied to tasks such as sentiment analysis, spam detection, topic classification, and more.

Logistic Regression

What is Logistic Regression?

- **Purpose:** Logistic regression models the probability that a given input x belongs to a particular category (e.g., positive or negative sentiment).
- **Output:** Produces a probability between 0 and 1 using the logistic (sigmoid) function.
- **Decision Boundary:** Assigns inputs to classes based on a threshold probability (commonly 0.5).

Mathematical Formulation

For a binary classification problem:

1. Model Representation:

$$P(y = 1|x) = \sigma(z) = \frac{1}{1 + e^{-z}}$$

...

Where:

- y is the binary target variable (0 or 1).
- x is the input feature vector.
- $z = \theta^T x$ is the linear combination of input features.
- θ is the vector of model parameters (weights).
- $\sigma(z)$ is the sigmoid function.

2. Sigmoid Function:

- **Graph:** S-shaped curve mapping any real value into the (0, 1) interval.
- **Properties:** Monotonically increasing, differentiable.

Logistic Regression

What is Logistic Regression?

- **Purpose:** Logistic regression models the probability that a given input x belongs to a particular category (e.g., positive or negative sentiment).
- **Output:** Produces a probability between 0 and 1 using the logistic (sigmoid) function.
- **Decision Boundary:** Assigns inputs to classes based on a threshold probability (commonly 0.5).

Mathematical Formulation

For a binary classification problem:

1. Model Representation:

$$P(y = 1|x) = \sigma(z) = \frac{1}{1 + e^{-z}}$$

...

Where:

- y is the binary target variable (0 or 1).
- x is the input feature vector.
- $z = \theta^T x$ is the linear combination of input features.
- θ is the vector of model parameters (weights).
- $\sigma(z)$ is the sigmoid function.

2. Sigmoid Function:

- **Graph:** S-shaped curve mapping any real value into the (0, 1) interval.
- **Properties:** Monotonically increasing, differentiable.

Logistic Regression

Logistic Regression in NLP

In NLP, logistic regression is used to classify text data by transforming words or phrases into numerical features that can be input into the model.

Common NLP Applications:

- **Sentiment Analysis:** Classifying text as expressing positive or negative sentiment.
- **Spam Detection:** Identifying whether an email or message is spam or not.
- **Topic Classification:** Assigning predefined categories to text documents.

Logistic Regression

Dataset Description

We'll use a simplified dataset consisting of a few movie reviews labeled as **positive** or **negative**. This small dataset will allow us to perform detailed calculations manually.

Dataset:

Review ID	Review Text	Sentiment Label
1	"I love this movie. It's fantastic and exciting."	Positive (1)
2	"This film was terrible and boring."	Negative (0)
3	"What an excellent and thrilling experience!"	Positive (1)
4	"I did not enjoy this movie. It was dull."	Negative (0)

Logistic Regression

Step 1: Data Preparation and Preprocessing

1.1 Text Preprocessing

For each review, we'll perform the following preprocessing steps:

- **Lowercasing:** Convert all text to lowercase.
- **Removing Punctuation:** Remove punctuation marks.
- **Tokenization:** Split text into individual words (tokens).
- **Stop Word Removal:** Remove common stop words (e.g., "this", "and", "was").
- **Stemming/Lemmatization:** Reduce words to their base form.

Processed Reviews:

1. Review 1:

- Original: "I love this movie. It's fantastic and exciting."
- Preprocessed Tokens: ["love", "movie", "fantastic", "exciting"]

2. Review 2:

Logistic Regression

Processed Reviews:

1. Review 1:

- Original: "I love this movie. It's fantastic and exciting."
- Preprocessed Tokens: ["love", "movie", "fantastic", "exciting"]

2. Review 2:

- Original: "This film was terrible and boring."
- Preprocessed Tokens: ["film", "terrible", "boring"]

3. Review 3:

- Original: "What an excellent and thrilling experience!"
- Preprocessed Tokens: ["excellent", "thrilling", "experience"]

4. Review 4:

- Original: "I did not enjoy this movie. It was dull."
- Preprocessed Tokens: ["not", "enjoy", "movie", "dull"]

Logistic Regression

Step 2: Feature Extraction Using Bag-of-Words Model

2.1 Building the Vocabulary

Create a list of unique words from all the preprocessed reviews.

Vocabulary:

1. love
2. movie
3. fantastic
4. exciting
5. film
6. terrible
7. boring
8. excellent
9. thrilling
10. experience
11. not
12. enjoy
13. dull



Logistic Regression

Vocabulary Size (V): 13

Word Index Mapping:

Word	Index
love	1
movie	2
fantastic	3
exciting	4
film	5
terrible	6
boring	7
excellent	8
thrilling	9
experience	10
not	11
enjoy	12
dull	13

Logistic Regression

2.2 Constructing Feature Vectors

For each review, create a feature vector x of size V , where each element represents the count of the corresponding word in the review.

Feature Vectors:

1. Review 1 (Positive, Label $y = 1$)

$$x^{(1)} = [1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0]^T$$

- Contains "love", "movie", "fantastic", "exciting"

Logistic Regression

Step 3: Mathematical Formulation of Logistic Regression

3.1 Model Representation

For each review i :

- Input Feature Vector: $x^{(i)} \in \mathbb{R}^V$
- Target Label: $y^{(i)} \in \{0, 1\}$

The logistic regression model predicts the probability that the review is positive ($y = 1$):

$$P(y^{(i)} = 1 | x^{(i)}, \theta) = \sigma(z^{(i)}) = \frac{1}{1 + e^{-z^{(i)}}}$$

Where:

- $z^{(i)} = \theta^T x^{(i)}$
- $\theta \in \mathbb{R}^V$ is the parameter vector (weights)
- $\sigma(z)$ is the sigmoid function



3.2 Cost Function

Logistic Regression

3.2 Cost Function

We use the logistic loss (cross-entropy loss):

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^m \left[y^{(i)} \log P(y^{(i)} = 1 | x^{(i)}, \theta) + (1 - y^{(i)}) \log(1 - P(y^{(i)} = 1 | x^{(i)}, \theta)) \right]$$

Where:

- m is the number of training examples (here, $m = 4$)

3.3 Gradient Descent Update Rule

To minimize $J(\theta)$, we update θ using gradient descent:

$$\theta := \theta - \alpha \nabla_{\theta} J(\theta)$$

Where:

- α is the learning rate
- $\nabla_{\theta} J(\theta)$ is the gradient of the cost function with respect to θ

Gradient Calculation:

$$\nabla_{\theta} J(\theta) = \frac{1}{m} \sum_{i=1}^m \left(P(y^{(i)} = 1 | x^{(i)}, \theta) - y^{(i)} \right) x^{(i)}$$

Logistic Regression

Step 4: Training the Logistic Regression Model

We'll perform one epoch of training with a small learning rate for demonstration.

4.1 Initialization

Initialize weights θ to zeros:

$$\theta = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]^T$$

4.2 Forward Pass and Calculations for Each Review

We'll compute $z^{(i)}$, $P^{(i)}$, and the contribution to the gradient from each training example.

Set Learning Rate (α): 0.1



Logistic Regression

Review 1:

- Feature Vector $x^{(1)}$:

$$x^{(1)} = [1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0]^T$$

- Label $y^{(1)} = 1$
- Compute $z^{(1)}$:

$$z^{(1)} = \theta^T x^{(1)} = [0]^T \cdot x^{(1)} = 0$$


- Compute Predicted Probability $P^{(1)}$:

$$P^{(1)} = \sigma(z^{(1)}) = \frac{1}{1 + e^{-0}} = 0.5$$

- Compute Error Term:

$$\delta^{(1)} = P^{(1)} - y^{(1)} = 0.5 - 1 = -0.5$$

- Contribution to Gradient:

$$\nabla_{\theta}^{(1)} = \delta^{(1)} x^{(1)} = -0.5 \cdot x^{(1)}$$




Logistic Regression

4.3 Computing the Total Gradient

Sum the contributions from all reviews:

$$\nabla_{\theta} = \frac{1}{m} \left(\nabla_{\theta}^{(1)} + \nabla_{\theta}^{(2)} + \nabla_{\theta}^{(3)} + \nabla_{\theta}^{(4)} \right)$$

Compute the sum:

- Initialize ∇_{θ} as a zero vector of size V .

1. Initialize:

$$\nabla_{\theta} = [0]^T \in \mathbb{R}^{13}$$

2. Add Contributions:

- From Review 1 ($\delta^{(1)} = -0.5$):

$$\nabla_{\theta} += -0.5 \cdot x^{(1)} = -0.5 \cdot [1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0]^T$$

$$\nabla_{\theta} = [-0.5, -0.5, -0.5, -0.5, 0, 0, 0, 0, 0, 0, 0, 0, 0]^T$$

3. Compute Average Gradient:

$$\nabla_{\theta} = \frac{1}{4} \cdot \nabla_{\theta} = \left[\frac{-0.5}{4}, \frac{0}{4}, \frac{-0.5}{4}, \frac{-0.5}{4}, \frac{0.5}{4}, \frac{0.5}{4}, \frac{0.5}{4}, \frac{0.5}{4}, \frac{-0.5}{4}, \frac{-0.5}{4}, \frac{-0.5}{4}, \frac{0.5}{4}, \frac{0.5}{4} \right]^T$$

Simplify:

$$\nabla_{\theta} = [-0.125, 0, -0.125, -0.125, 0.125, 0.125, 0.125, -0.125, -0.125, -0.125, 0.125, 0.125, 0.125]^T$$

Logistic Regression

4.4 Update Weights

Update θ using the gradient descent rule:

$$\theta := \theta - \alpha \nabla_{\theta}$$

Using $\alpha = 0.1$:

$$\theta := \theta - 0.1 \cdot \nabla_{\theta}$$

Compute new θ :

$$\theta = [0] - 0.1 \cdot [-0.125, 0, -0.125, -0.125, 0.125, 0.125, 0.125, -0.125, -0.125, -0.125, 0.125, 0.125, 0.125]^T$$

Simplify:

$$\theta = [0.0125, 0, 0.0125, 0.0125, -0.0125, -0.0125, -0.0125, 0.0125, 0.0125, 0.0125, -0.0125, -0.0125, -0.0125]^T$$



Logistic Regression

Step 5: Evaluating the Model After One Update

Let's compute the predictions on the training data using the updated weights θ .

5.1 Forward Pass for Each Review

Review 1:

- Feature Vector $x^{(1)}$:

$$x^{(1)} = [1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0]^T$$

- Compute $z^{(1)}$:

$$z^{(1)} = \theta^T x^{(1)} = 0.0125 \cdot 1 + 0 + 0.0125 \cdot 1 + 0.0125 \cdot 1 + 0 + \dots = 0.0125 + 0 + 0.0125 + 0.0125 = 0.0375$$

- Compute Predicted Probability $P^{(1)}$:

$$P^{(1)} = \sigma(z^{(1)}) = \frac{1}{1 + e^{-0.0375}} \approx 0.5094$$

Logistic Regression

5.2 Interpretation of Predictions

- Reviews 1 and 3 (Positive Reviews):
 - Predicted probabilities: ~ 0.5094
 - Slightly above 0.5, predicted as positive.
- Reviews 2 and 4 (Negative Reviews):
 - Predicted probabilities: ~ 0.4906
 - Slightly below 0.5, predicted as negative.

5.3 Observations

- After one update, the model slightly adjusts predictions in the correct direction.
- The model needs more iterations and possibly a larger learning rate or more sophisticated optimization to improve performance.