# Intro to NLP
# Sentiment Analysis with Logistic Regression

Gennaro S. Rodrigues, Ph.D.

# Overview

1. Supervised Machine Learning
2. Sentiment Analysis
3. Vocabulary and Features Extraction
4. Logistic Regression
5. TF-IDF

# Sentiment Analysis

- Naïve Bayes vs. Logistic Regression
  - **Discriminative**: Learns how to **distinguish** between classes. What characteristics the examples of that given class have in common that are not as present in examples of other classes.
  - **Generative**: Learns and **understands** what a class looks like. Given an input, decides which model of which class is a better fit.

# Sentiment Analysis

- Naïve Bayes vs. Logistic Regression
  - **Discriminative**: Learns how to **distinguish** between classes. What characteristics the examples of that given class have in common that are not as present in examples of other classes.
  - **Generative**: Learns and **understands** what a class looks like. Given an input, decides which model of which class is a better fit.

  - Logistic Regression is Discriminative.
  - Naïve Bayes is Generative.
    - We can even use it to generate text.

# Sentiment Analysis

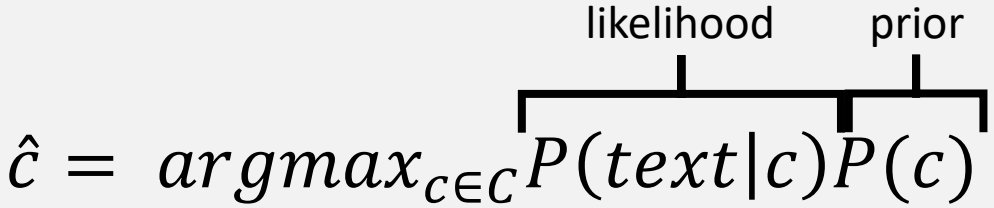
- Naïve Bayes vs. Logistic Regression
  - Logistic Regression is Discriminative.
  - Naïve Bayes is Generative.

- Remember Naïve Bayes:
  - Classifies computing a likelihood.
    - Expresses how to generate features of a document *"if we knew it was of class A"*.
  - Not a direct probability.

# Sentiment Analysis

- Naïve Bayes vs. Logistic Regression
  - Logistic Regression is Discriminative.
  - Naïve Bayes is Generative.

- Remember Naïve Bayes:
  - Classifies computing a likelihood.
    - Expresses how to generate features of a document *"if we knew it was of class A"*.
  - Not a direct probability.



# Sentiment Analysis

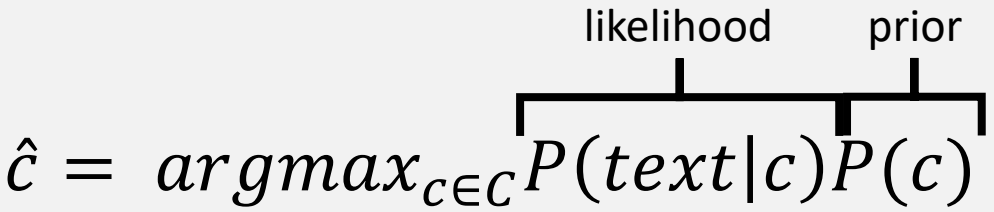
- Naïve Bayes vs. Logistic Regression
  - Logistic Regression is Discriminative.
  - Naïve Bayes is Generative.

- A discriminative model:
  - Will directly compute the probability of a text being of a certain class A.

# Sentiment Analysis

- Remember Naïve Bayes:
  - Classifies computing a likelihood.
    - Expresses how to generate features of a document *"if we knew it was of class A"*.
  - Not a direct probability.

- A discriminative model:
  - Will directly compute the probability of a text being of a certain class A.




**Generative models** make use of a **likelihood**, which expresses how to **generate features** of a document *if we knew it was of c class*.

**Discriminative models** will compute the **probability** directly. Will assign high weights to features to improve the capacity of **discrimitating** between classes, but *not generating* an example of one.

# Sentiment Analysis

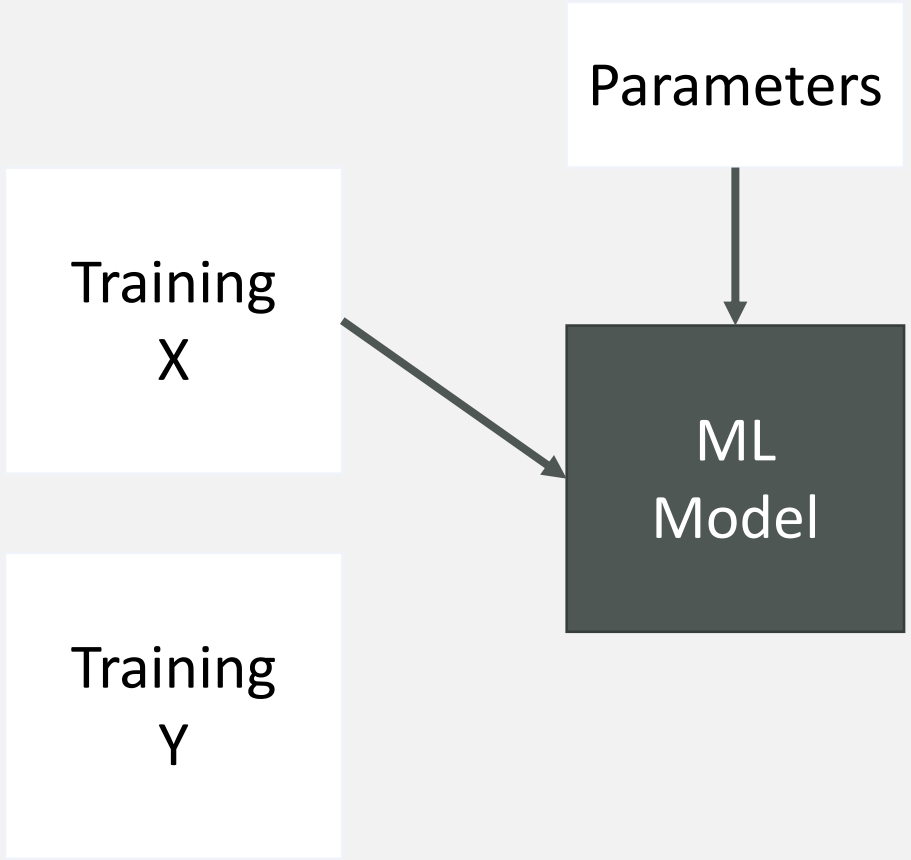
- Supervised Machine Learning

Training
X

Training
Y

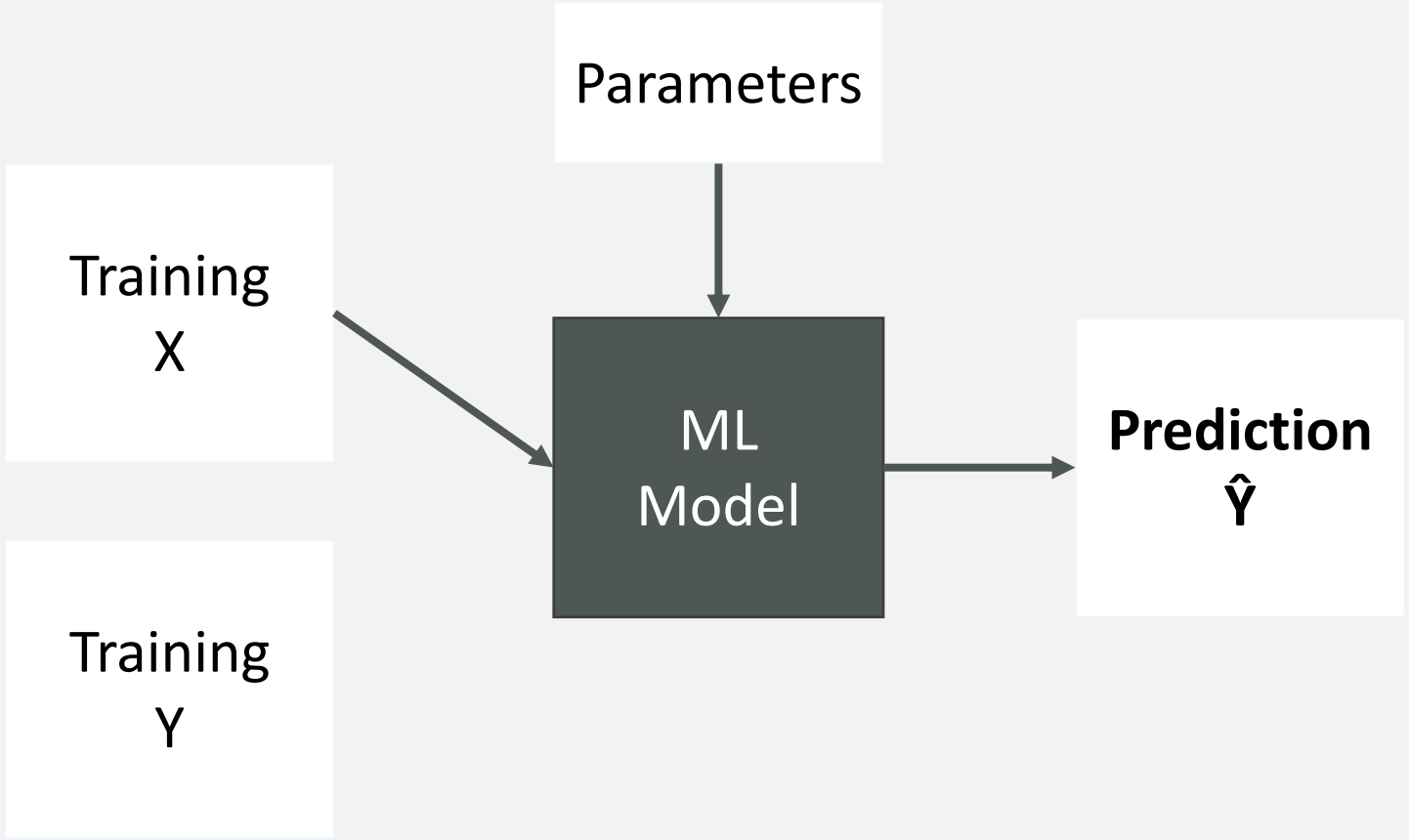
# Sentiment Analysis

- Supervised Machine Learning



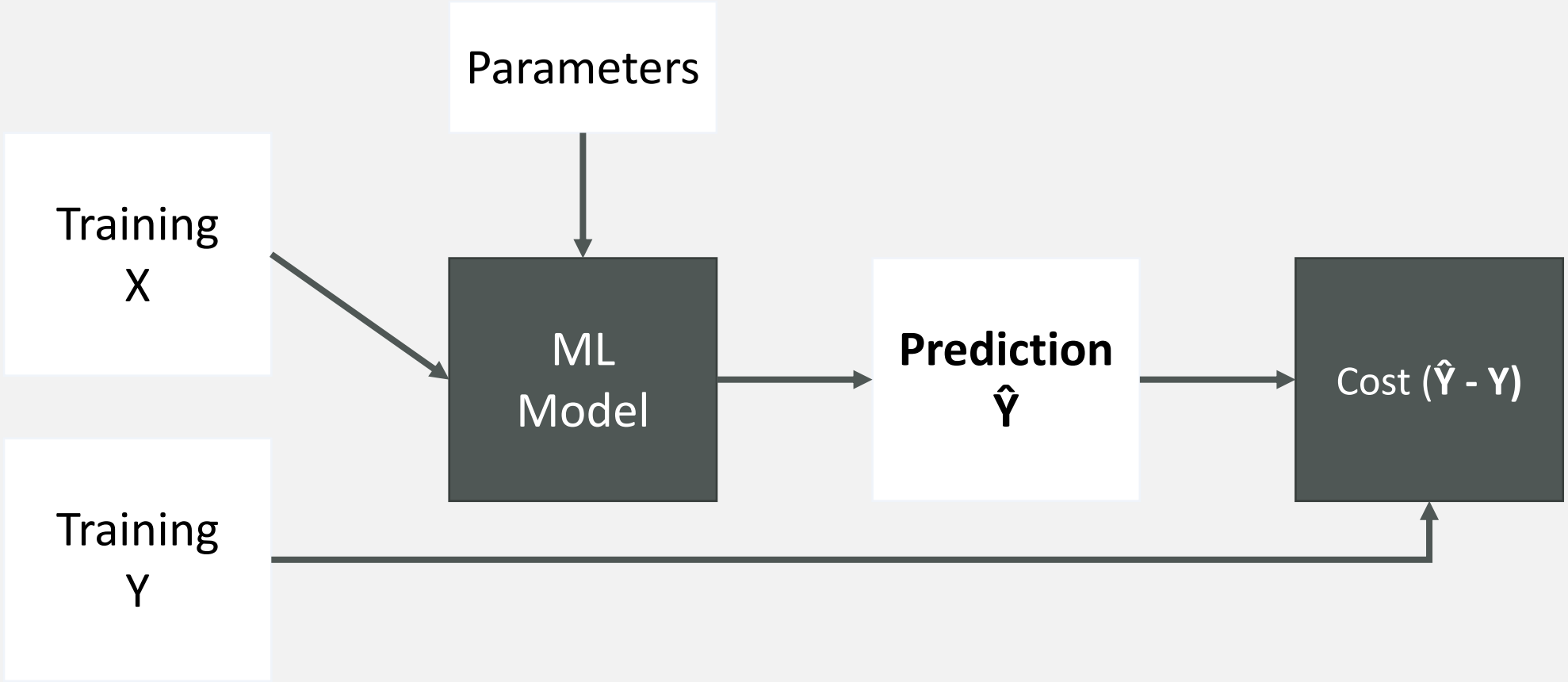
# Sentiment Analysis

- Supervised Machine Learning



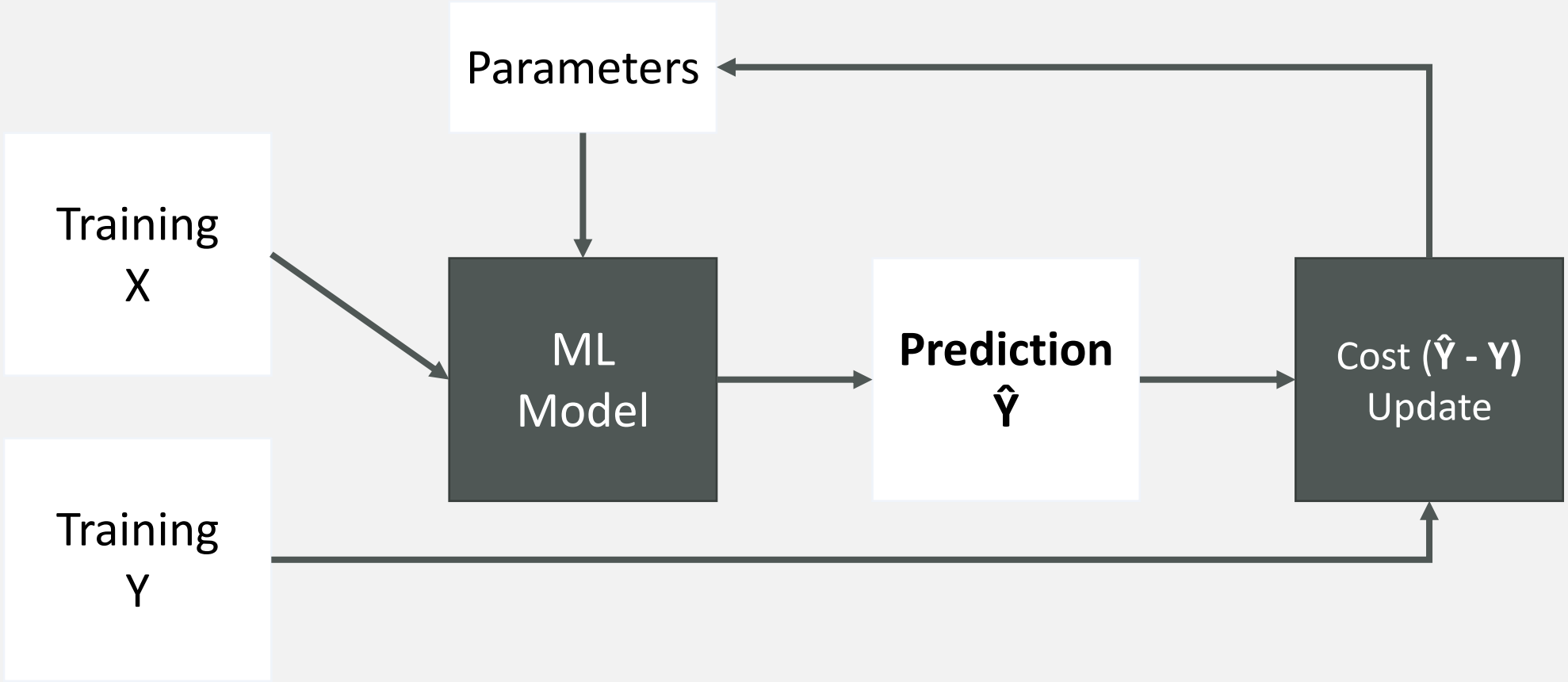
# Sentiment Analysis

- Supervised Machine Learning



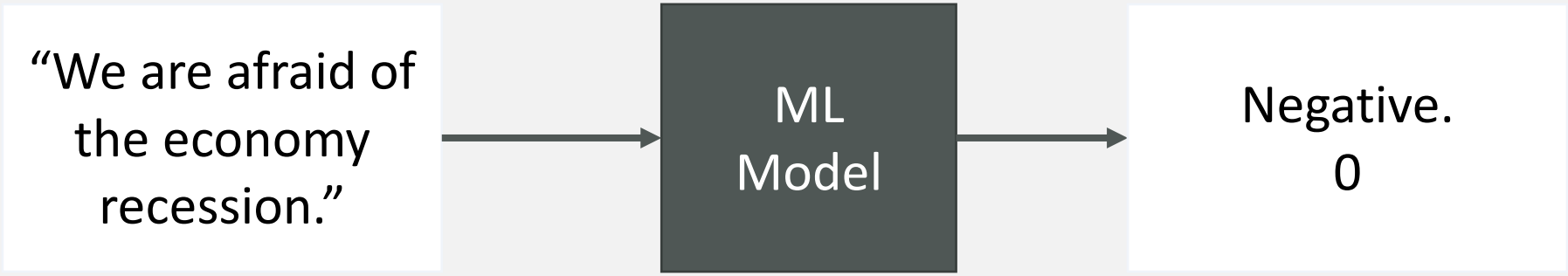
# Sentiment Analysis

- Supervised Machine Learning



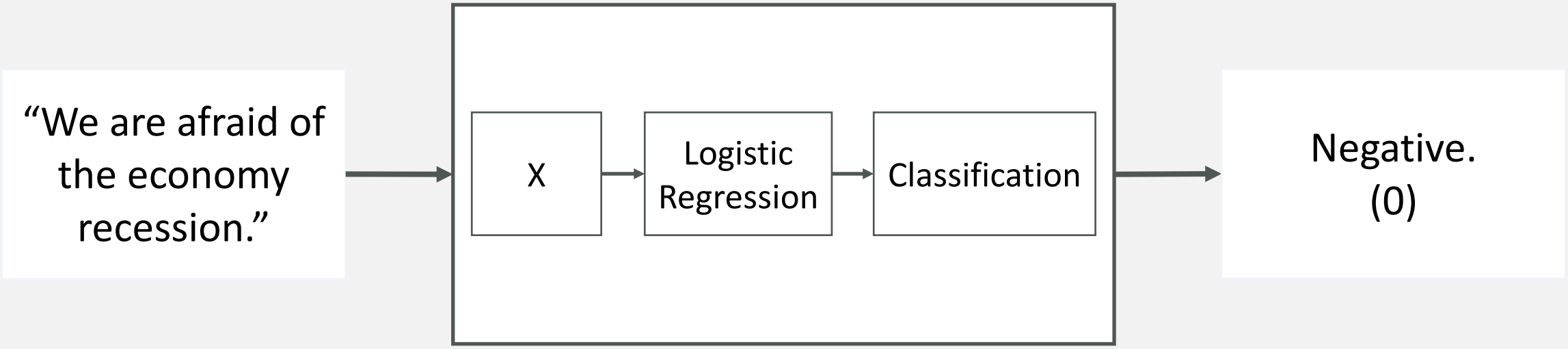
# Sentiment Analysis

- Supervised Machine Learning.
- Goal: Predict a classification for a text.



# Sentiment Analysis

- Supervised Machine Learning.
- Goal: Predict a classification for a text.



# Sentiment Analysis

- Given input/output $(x^i, y^i)$ pairs we shall:
  - Represent the $x^i$ input as a set of N $[x_1, x_2, \ldots, x_n]$ features.
  - Have a classification function to compute $y^i$ from $x^i$, via an estimation $p(y|x)$ for the ŷ estimated class, like **sigmoid** or **softmax**.
  - Define an objective functions for learning, normally minimizing error on $(x^i, y^i)$ training examples pairs. That´s also what we call a **loss function**.
  - Have an algorithm for the optimization of the objective function, such as **stochastic gradient descent**.

# Sentiment Analysis

- A Logistic Regression has two phases.
- **Training**: where we train the system (the function weights) using gradient descent and the loss function.
- **Test**: Given a test exemple we compute its probability for each possible classification label and return the highest one.

# Vocabulary

- The text input shall be processed.
- Neural Networks work with numbers, not words.
- We will represent our examples as vectors.

# Vocabulary

- The text input shall be processed.
- Neural Networks work with numbers, not words.
- We will represent our examples as vectors.
- Vocabulary: list of words present on our text corpora.

“We are afraid of the economy recession.”

V = [We, are, afraid, of, the, economy, recession, …, I, scientist, not]

# Feature Extraction: Sparse Representation

“We are afraid of the economy recession.”

V = [We, are, afraid, of, the, economy, recession, …, I, scientist, not]

X = [ 1, 1, 1, 1, 1, 1, 1, 0, 0, 0]

- X is a list of the same size as V
- X represents the sentence with 1’s for words present in the sentence, and 0’s for the ones absent.

# Feature Extraction: Sparse Representation

“We are afraid of the economy recession.”

V = [We, are, afraid, of, the, economy, recession, …, I, scientist, not]

X = [ 1, 1, 1, 1, 1, 1, 1, 0, 0, 0]

- Problem: memory usage (V can be HUGE, often is).
- A lot of the data is 0.
- A logistic regression model would have too many parameters (|V|+1).
  - Large training and prediction time.

# Feature Extraction: Frequencies

- Given a labeled dataset, we can extract the sentiment frequency of a word.
  - In our case, positive/negative.
- Calculating the frequency on which a word appears on the positive and negative sets.
- Divide our corpus in two groups.

# Feature Extraction: Frequencies

- Divide our corpus in two groups.

Negative:
“We are afraid of the economy recession.”
“People are afraid of losing their jobs.”

Positive:
“The economy is recovering.”
“People are getting new jobs.”

# Feature Extraction: Frequencies

- Divide our corpus in two groups.

Negative:
“We are afraid of the economy recession.”
“People are afraid of losing their jobs.”

| Vocabulary | Neg Frequency (0) | Pos Frequency (1) |
|---|---|---|
| we | 1 | |
| are | 2 | |
| afraid | 2 | |
| of | 2 | |
| the | 1 | |
| economy | 1 | |
| recession | 1 | |
| people | 1 | |
| losing | 1 | |
| their | 1 | |
| jobs | 1 | |
| is | 0 | |
| recovering | 0 | |
| getting | 0 | |

# Feature Extraction: Frequencies

- Divide our corpus in two groups.

Positive:
“The economy is recovering.”
“People are getting new jobs.”

| Vocabulary | Neg Frequency (0) | Pos Frequency (1) |
|---|---|---|
| we | 1 | 0 |
| are | 2 | 1 |
| afraid | 2 | 0 |
| of | 2 | 0 |
| the | 1 | 1 |
| economy | 1 | 1 |
| recession | 1 | 0 |
| people | 1 | 1 |
| losing | 1 | 0 |
| their | 1 | 0 |
| jobs | 1 | 1 |
| is | 0 | 1 |
| recovering | 0 | 1 |
| getting | 0 | 1 |

# Feature Extraction: Frequencies

| Vocabulary | Neg Frequency (0) | Pos Frequency (1) |
|---|---|---|
| we | 1 | 0 |
| are | 2 | 1 |
| afraid | 2 | 0 |
| of | 2 | 0 |
| the | 1 | 1 |
| economy | 1 | 1 |
| recession | 1 | 0 |
| people | 1 | 1 |
| losing | 1 | 0 |
| their | 1 | 0 |
| jobs | 1 | 1 |
| is | 0 | 1 |
| recovering | 0 | 1 |
| getting | 0 | 1 |

# Feature Extraction: Frequencies

- We can now use the frequencies extractions to create features for our logistic regression models.
- With the sparse representation, we would need |V| + 1 features.
- Now we can generate a vector that represents a sentence with as little as 3 features.
- E.g.: feature extraction for a text *m*:

$$X_m = \left[1, \sum Neg.Freqs., \sum Pos.Freqs.\right]$$

# Feature Extraction: Frequencies

- E.g.: “The economy is recovering from recession”

$$X_m = \left[1, \sum Neg.Freqs., \sum Pos.Freqs.\right]$$

$$X_m = [1, 3, 4]$$

| Vocabulary | Neg Frequency (0) | Pos Frequency (1) |
|---|---|---|
| we | 1 | 0 |
| are | 2 | 1 |
| afraid | 2 | 0 |
| of | 2 | 0 |
| the | 1 | 1 |
| economy | 1 | 1 |
| recession | 1 | 0 |
| people | 1 | 1 |
| losing | 1 | 0 |
| their | 1 | 0 |
| jobs | 1 | 1 |
| is | 0 | 1 |
| recovering | 0 | 1 |
| getting | 0 | 1 |

# Preprocessing

- Some texts might contain undesirable information.
- Not all information is equally valid.
- Some words are variations from other words and have the same meaning.
- Words that are very frequent in a vocabulary are less meaningful.
  - Stop Words.
  - E.g.: I, my, me, are, is, you, he, …

# Preprocessing

E.g.:

@friendlyGamer I am very afraid of the war on Ukraine!!!! #world

# Preprocessing: stop words removal

E.g.:

@friendlyGamer ~~I am~~ very afraid ~~of the~~ war ~~on~~ Ukraine!!!! #world

Stop words: I, am, of, the, on

(the list of stop words is usually very big)

# Preprocessing: punctuation removal

E.g.:

@friendlyGamer ~~I am~~ very afraid ~~of the~~ war ~~on~~ Ukraine~~!!!!~~ #world

Punctuation: !

ALERT: For some models and use cases, punctuation might be kept for their meaning (we will see that in the future).

# Preprocessing: internet removal

E.g.:

~~@friendlyGamer I am~~ very afraid ~~of the~~ war ~~on~~ Ukraine~~!!!! #world~~

User tags, hashtags, links, etc.

ALERT: For some models and use cases, hashtags could be kept. They might represent emotions.

# Preprocessing: stemming and lowercasing

E.g.:

~~@friendlyGamer I am~~ very afraid ~~of the~~ war ~~on~~ ukraine~~!!!! #world~~

- Lowercasing assures different models of a word are the same (“Ukraine” and “ukraine”)
- Stemming: recovering, recovered -> recover
- Reduces the vocabulary considerably.

# Preprocessing: stemming and lowercasing

E.g.:

~~@friendlyGamer I am~~ very afraid ~~of the~~ war ~~on~~ ukraine~~!!!! #world~~

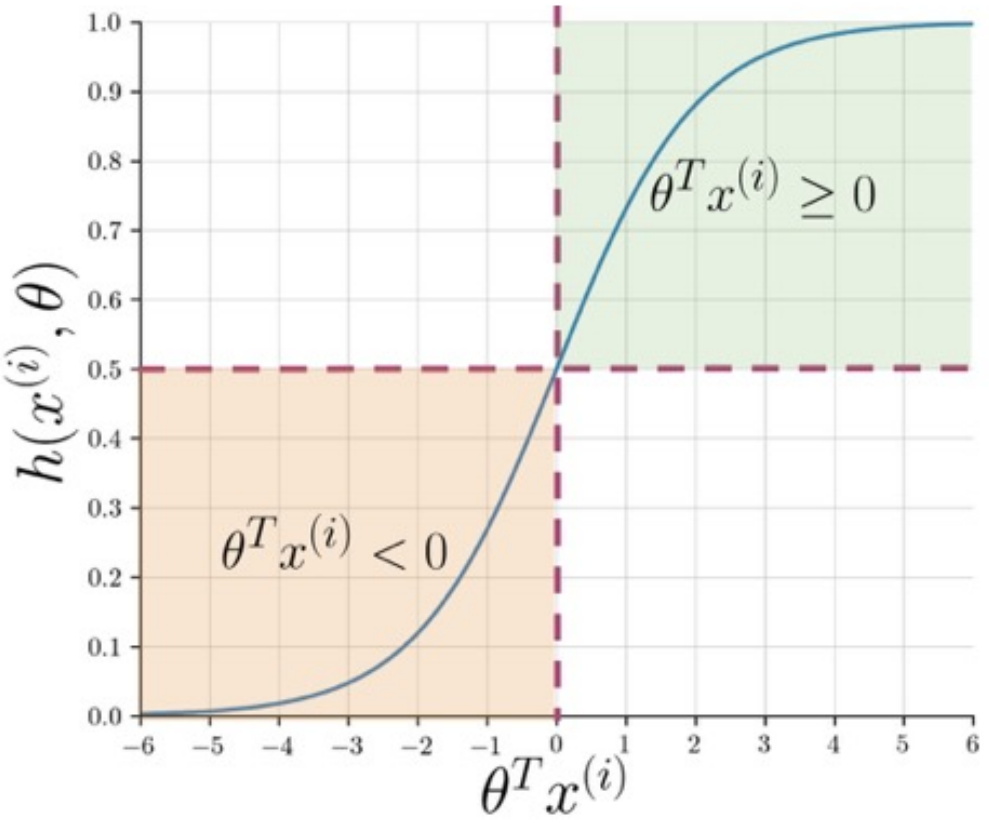
Preprocessed text:

[very, afraid, war, ukraine]

# Logistic Regression

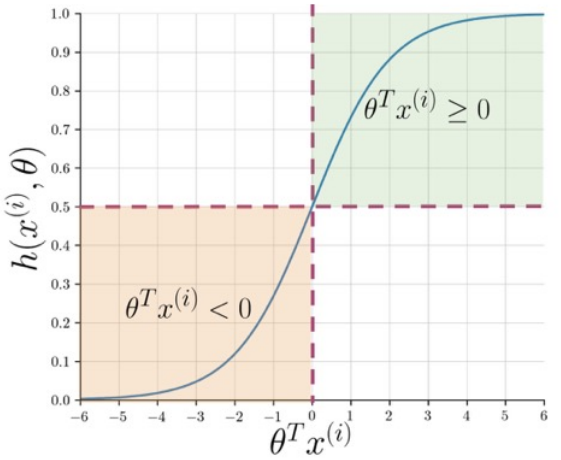
- Remember Logistic Regression.

$$h(x^{(i)}, \theta) = \frac{1}{1 + e^{-\theta^T x^{(i)}}}$$



# Logistic Regression

$$h(x^{(i)}, \theta) = \frac{1}{1 + e^{-\theta^T x^{(i)}}}$$




- Remember Logistic Regression.

$$z = \left( \sum_{i=1}^{n} w_i x_i \right) + b$$

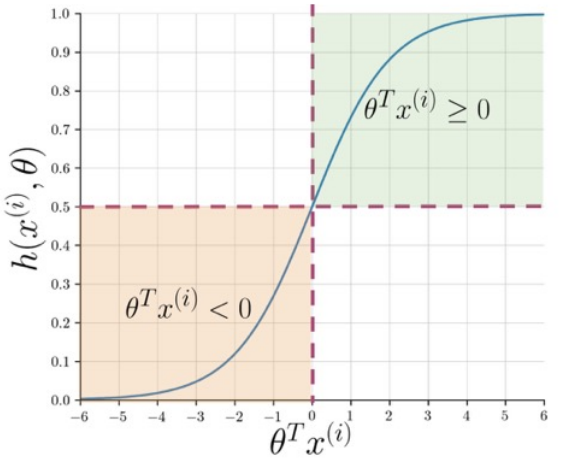
$$z = \boldsymbol{w} \cdot \boldsymbol{x} + b$$

To have a probability:

$$\sigma(z) = 1/(1 + e^{-z})$$

# Logistic Regression

$$h(x^{(i)}, \theta) = \frac{1}{1 + e^{-\theta^T x^{(i)}}}$$




To have a probability:

$$\sigma(z) = 1/(1 + e^{-z})$$

$$P(y = 1) = \sigma(\boldsymbol{w} \cdot \boldsymbol{x} + b)$$
$$P(y = 0) = 1 - \sigma(\boldsymbol{w} \cdot \boldsymbol{x} + b)$$

But the sigmoid function hast the property $1 - \sigma(x) = \sigma(-x)$

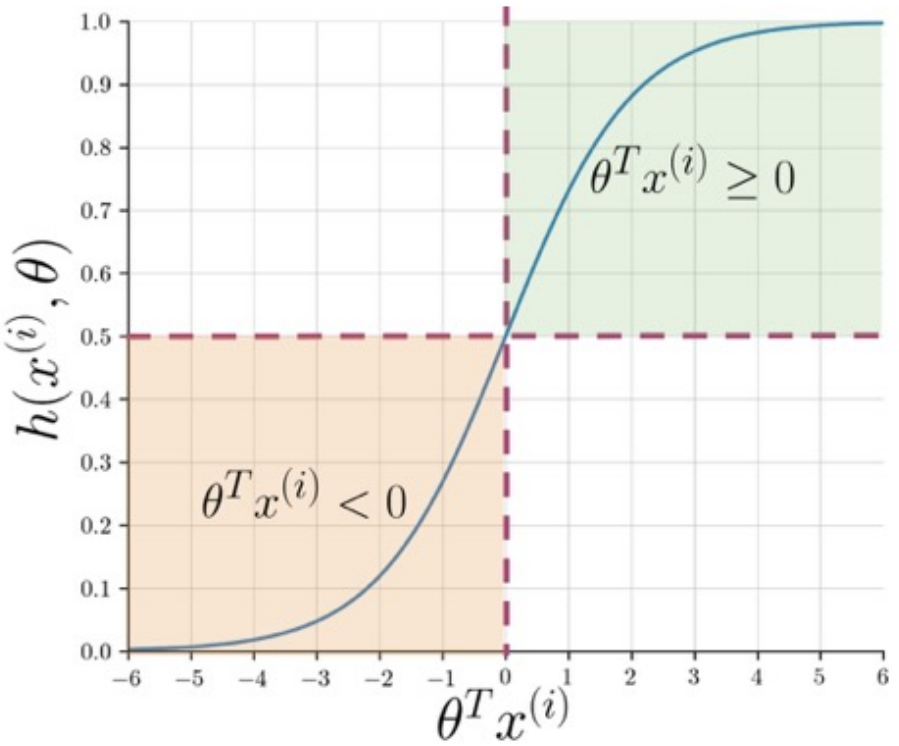
Therefore:

$$P(y = 0) = \sigma(-(\boldsymbol{w} \cdot \boldsymbol{x} + b))$$

# Logistic Regression

Decision boundary:

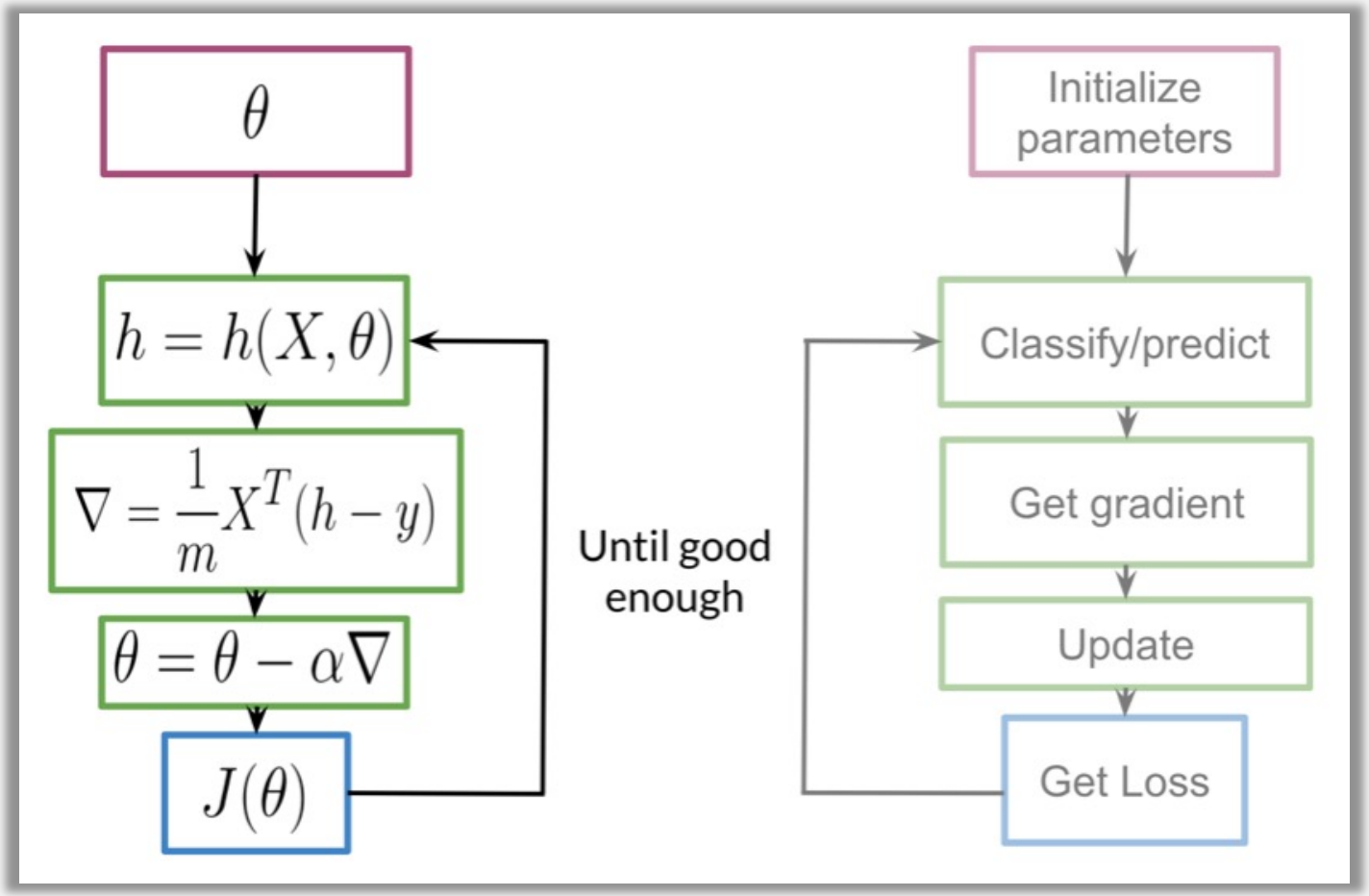
$$h(x^{(i)}, \theta) = \frac{1}{1 + e^{-\theta^T x^{(i)}}}$$




$$\begin{cases} 1, \text{ if } P(y = 1|x) > 0.5 \\ 0 \text{ otherwise} \end{cases}$$

# Logistic Regression

- Remember Logistic Regression.



# Logistic Regression

- Example:

w = [2.5,−5.0,−1.2,0.5,2.0,0.7]

b = 0.1

$$P(y=1|x) = \sigma(\boldsymbol{w}\cdot\boldsymbol{x}+b)$$
$$= \sigma([2.5,-5,-1.2,0.5,2,0.7]\cdot[3,2,1,3,0,4.19]+0.1)$$
$$= \sigma(0.833)$$
$$P(y=1|x) = 0.70$$

$$P(y=0|x) = 1-\sigma(\boldsymbol{w}\cdot\boldsymbol{x}+b)$$
$$P(y=0|x) = 0.30$$

| Var | Definition | Value in Fig. 5.2 |
|---|---|---|
| $x_1$ | count(positive lexicon words ∈ doc) | 3 |
| $x_2$ | count(negative lexicon words ∈ doc) | 2 |
| $x_3$ | $\begin{cases} 1 & \text{if "no"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$ | 1 |
| $x_4$ | count(1st and 2nd pronouns ∈ doc) | 3 |
| $x_5$ | $\begin{cases} 1 & \text{if "!"} \in \text{doc} \\ 0 & \text{otherwise} \end{cases}$ | 0 |
| $x_6$ | ln(word count of doc) | $\ln(66) = 4.19$ |

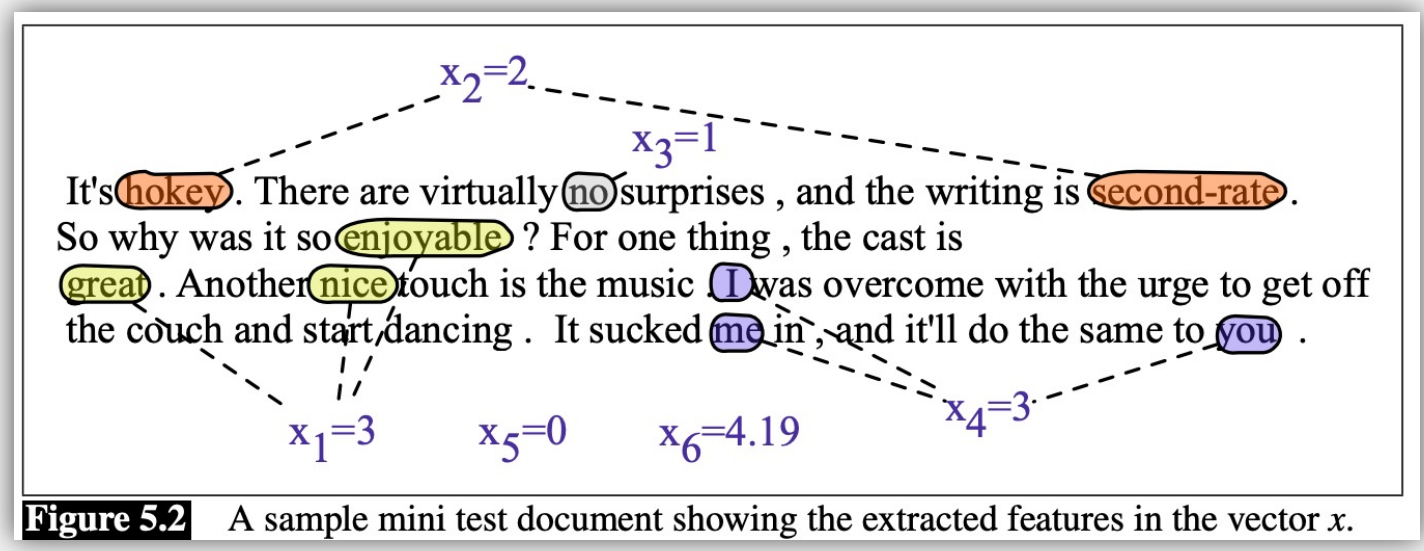



**Figure 5.2** A sample mini test document showing the extracted features in the vector *x*.

(Source: Jurafsky et. Al., Speech and Language Processing [3rd ed. draft])

# Logistic Regression

- Testing:

$$pred = h(X \cdot \theta) \geq 0.5$$

$$\begin{bmatrix} 0.4 \\ 0.3 \\ 0.9 \end{bmatrix} \geq 0.5 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$accuracy = \sum_{i=1}^{m} \frac{pred_i == yval_i}{m}$$

# Logistic Regression x Naïve Bayes

- Naïve Bayes has strong independence assumptions.
- Logistic Regression is much more robust to correlated features.
  - If two features are correlated, will divide the weight between the two.
- When there are many correlated features, logistic regression will provide a better probability.
  - Best for larger datasets.
- Naïve Bayes works better for very small datasets and documens.
- Naïve Bayes is easy to implement and very fast to train (no optimization step)

# A First Step into Language Models

"Tell me who you walk with, and I will tell you who you are."


(Source: Jurafsky et. Al., Speech and Language Processing [3rd ed. draft])

# TF-IDF

- Words occurring together frequently are important
- Words occurring TOO frequently (like stopwords) are not.
  - How to balance those two?
- TF: Term Frequency
- IDF: Inverse Document Frequency

# TF-IDF

- TF-IDF:
- TF:

$$tf_{t,d} = count(t, d)$$

The frequency of the term *t* at document *d*

# TF-IDF

- TF-IDF:
- TF:

$$tf_{t,d} = \log_{10}(count(t,d) + 1)$$

The frequency of the term *t* at document *d*

# TF-IDF

- TF-IDF:
- Now we want to emphasize discriminative words, i.e., those which occur in a few documents.
  - Terms that occur in a few documents are more useful in discriminating those from the rest of the dataset.
- Document Frequency (df): number of documents the term appears in.
- Collection Frequency: total number of occurrences in whole dataset.

# TF-IDF

- TF-IDF:
- IDF:

$$idf_t = \log_{10} \frac{N}{df_t}$$

Where *N* is the total number of documents

# TF-IDF

- IDF:

$$idf_t = \log_{10} \frac{N}{df_t}$$

Where *N* is the total number of documents

- TF-IDF:

$$w_{t,d} = tf_{t,d} \times idf_t$$

# TF-IDF

- Exempli gratia, from Shakespeare:

| | As You Like It | Twelfth Night | Julius Caesar | Henry V |
|---|---|---|---|---|
| **battle** | 1 | 0 | 7 | 13 |
| **good** | 114 | 80 | 62 | 89 |
| **fool** | 36 | 58 | 1 | 4 |
| **wit** | 20 | 15 | 2 | 3 |

| Word | df | idf |
|---|---|---|
| Romeo | 1 | 1.57 |
| salad | 2 | 1.27 |
| Falstaff | 4 | 0.967 |
| forest | 12 | 0.489 |
| battle | 21 | 0.246 |
| wit | 34 | 0.037 |
| fool | 36 | 0.012 |
| good | 37 | 0 |
| sweet | 37 | 0 |

| | As You Like It | Twelfth Night | Julius Caesar | Henry V |
|---|---|---|---|---|
| **battle** | 0.074 | 0 | 0.22 | 0.28 |
| **good** | 0 | 0 | 0 | 0 |
| **fool** | 0.019 | 0.021 | 0.0036 | 0.0083 |
| **wit** | 0.049 | 0.044 | 0.018 | 0.022 |

(Source: Jurafsky et. Al., Speech and Language Processing [3rd ed. draft])

# Thank you!