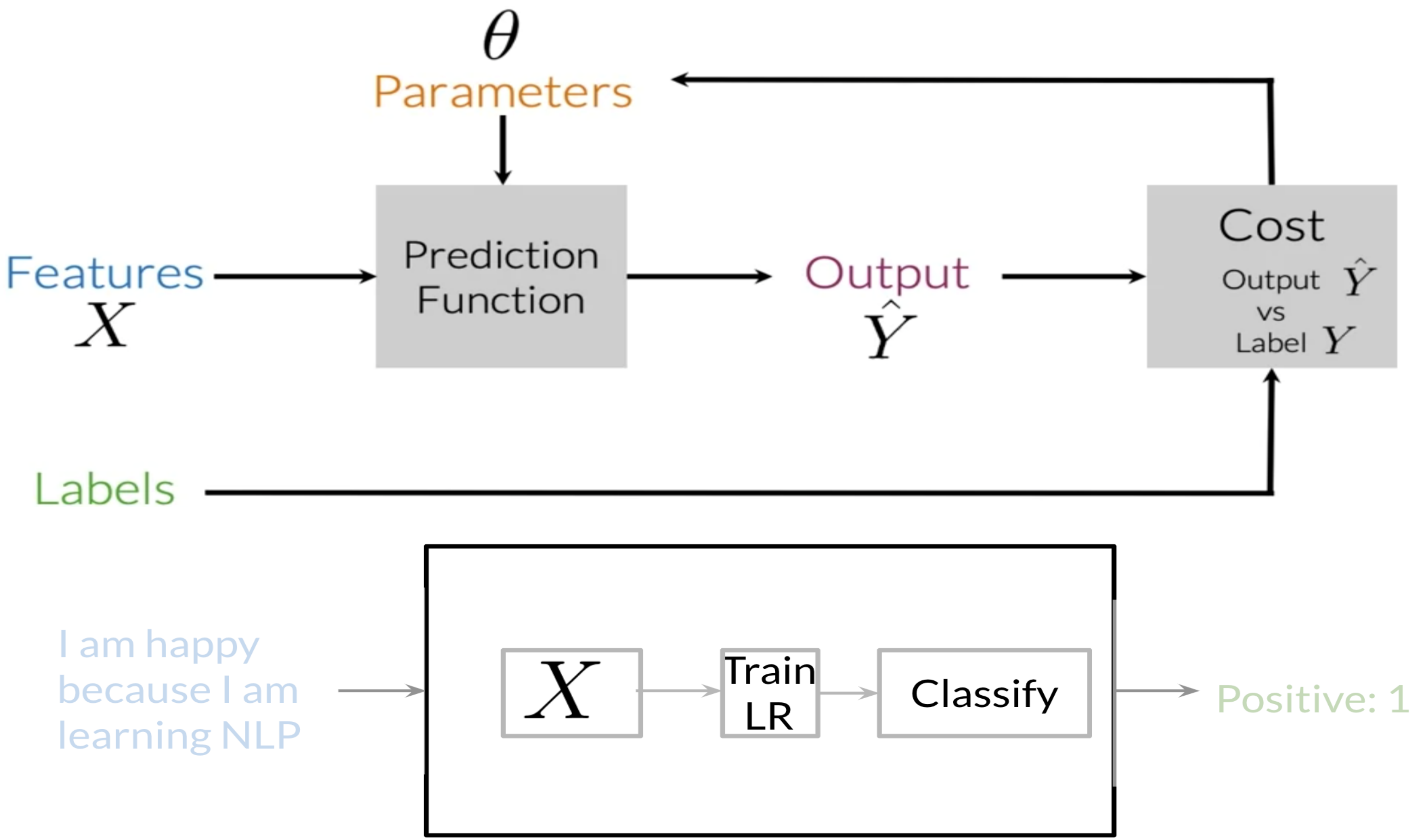
Sentiment Analysis

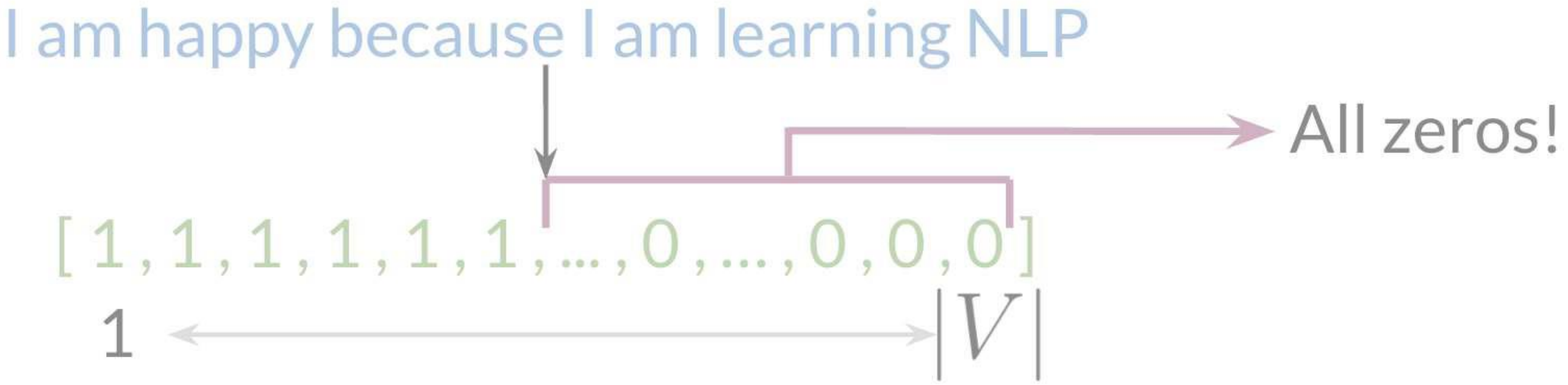
Using Logistic Regression

Compiled by:
Dr. Amit Kumar Trivedi
Department of Computer Science and Engineering
TIET, Patiala
Source of slide: DeepLearning.AI

Supervised ML



Vocabulary and Feature Extraction



$$[\theta_0, \theta_1, \theta_2, \ldots, \theta_n] \longrightarrow$$

$$n = |V|$$

1. Large training time
2. Large prediction time

Feature Extraction with Frequencies

Positive tweets

I am happy because I am learning NLP

I am happy

Negative tweets

I am sad, I am not learning NLP

I am sad

Vocabulary	PosFreq (1)	NegFreq (0)
I	3	3
am	3	3
happy	2	0
because	1	0
learning	1	1
NLP	1	1
sad	0	2
not	0	1

freqs: dictionary mapping from (word, class) to frequency

Vocabulary	PosFreq (1)
I	3
am	3
happy	2
because	1
learning	1
NLP	1
sad	0
not	0

I am sad, I am not learning NLP

$$X_m = [1, \sum_w freqs(w,1), \sum_w freqs(w,0)]$$

$\boxed{8}$

Putting it all together

I am Happy Because i am learning NLP @deeplearning

↓ Preprocessing

[happy, learn, nlp]

↓ Feature Extraction

Bias ← [1, 4, 2] → Sum negative frequencies

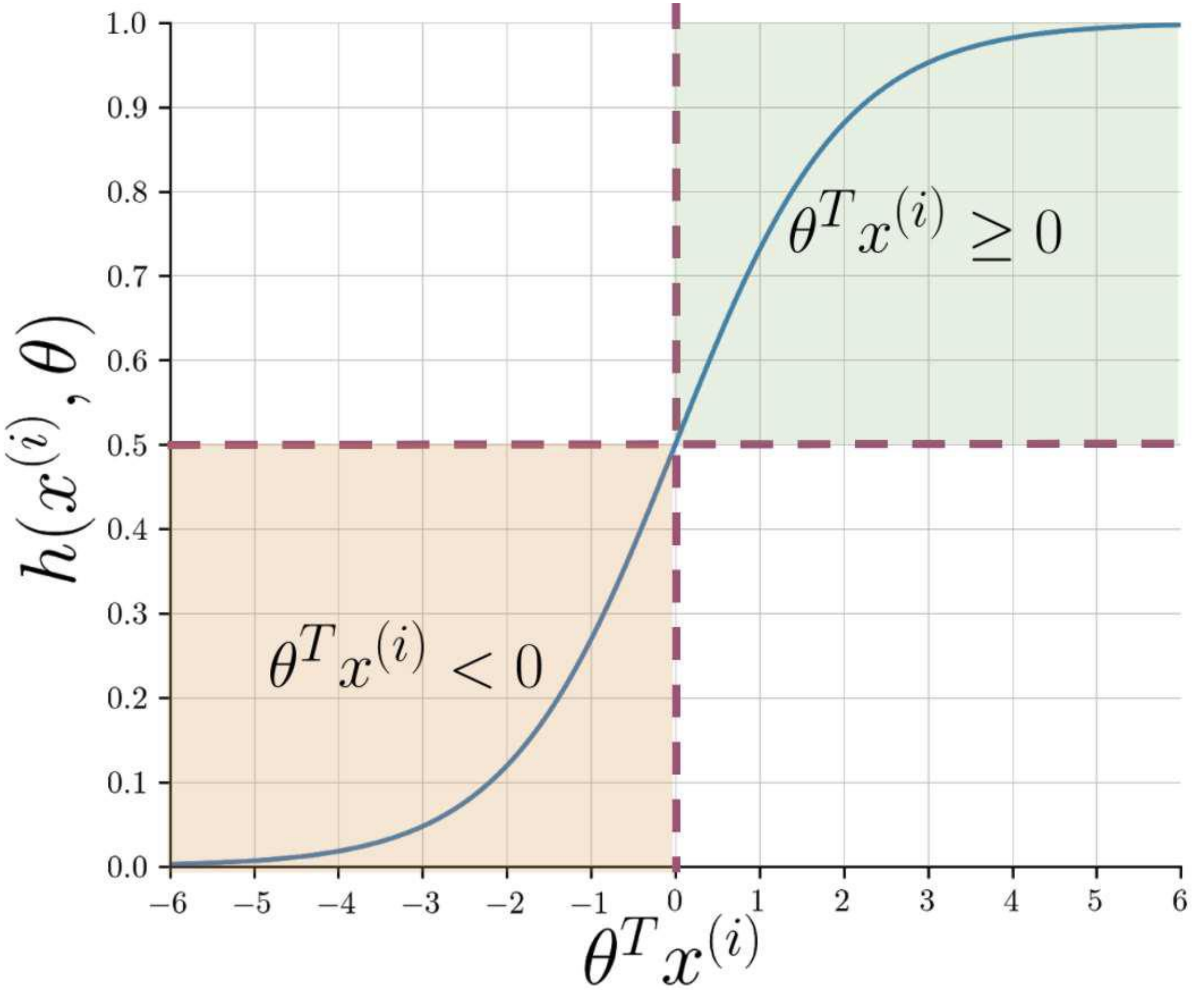
↓

Sum positive frequencies

$$\boldsymbol{X} = \begin{bmatrix} 1 & X_1^{(1)} & X_2^{(1)} \\ 1 & X_1^{(2)} & X_2^{(2)} \\ \vdots & \vdots & \vdots \\ 1 & X_1^{(m)} & X_2^{(m)} \end{bmatrix}$$

Logistic Regression: Overview

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



Logistic Regression: Overview

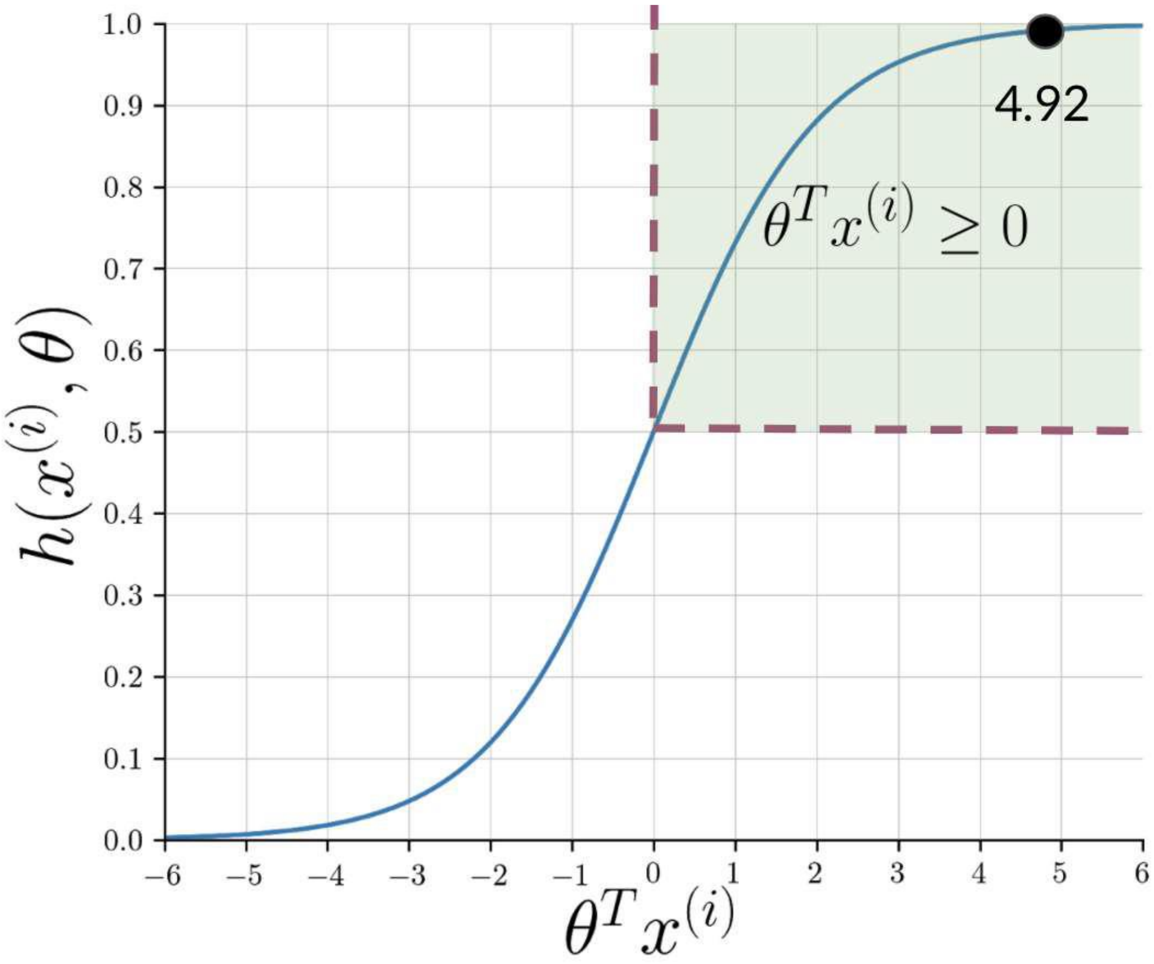
@YMourri and
@AndrewYNg are tuning a
GREAT AI model

↓

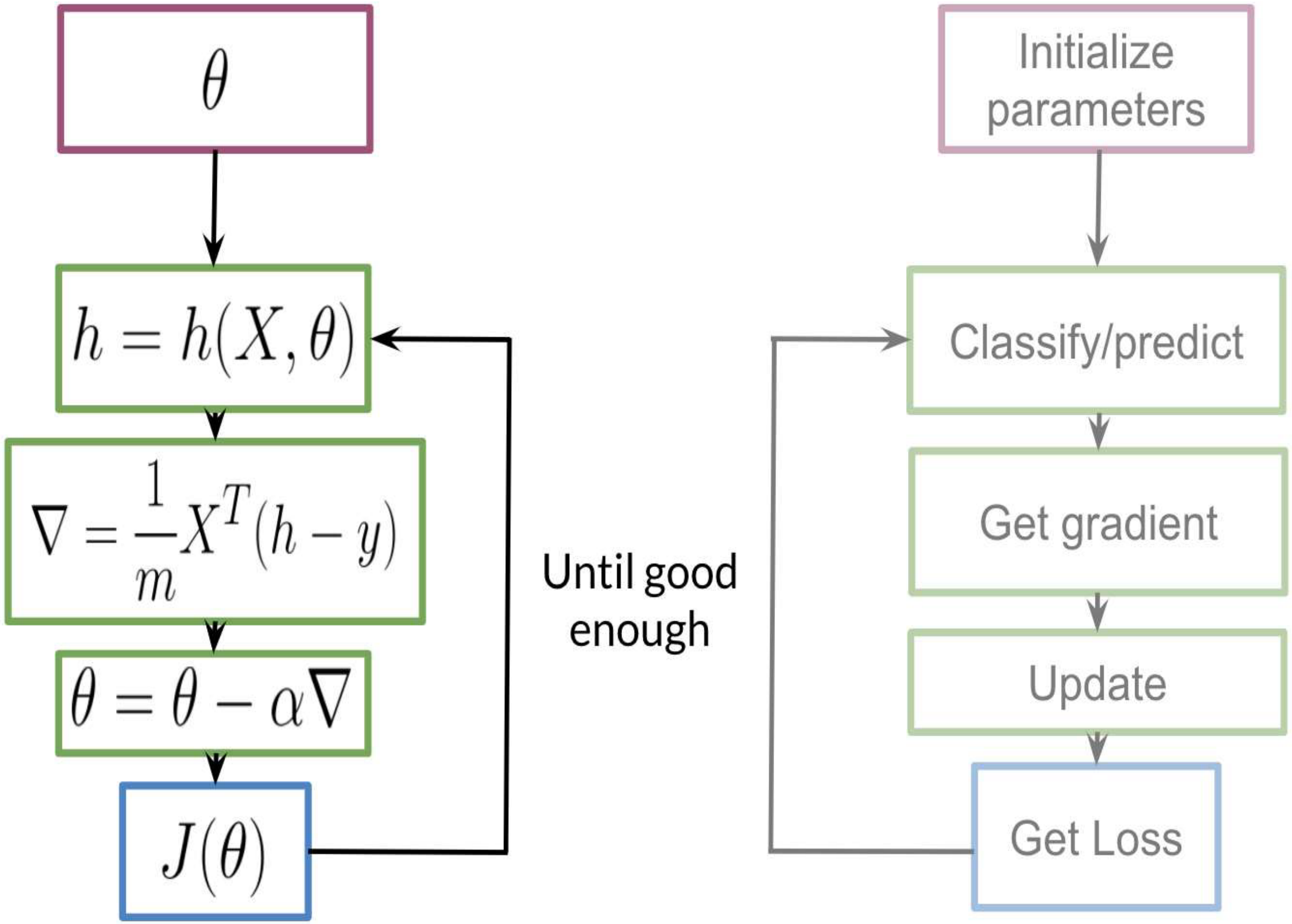
[tun, ai, great, model]

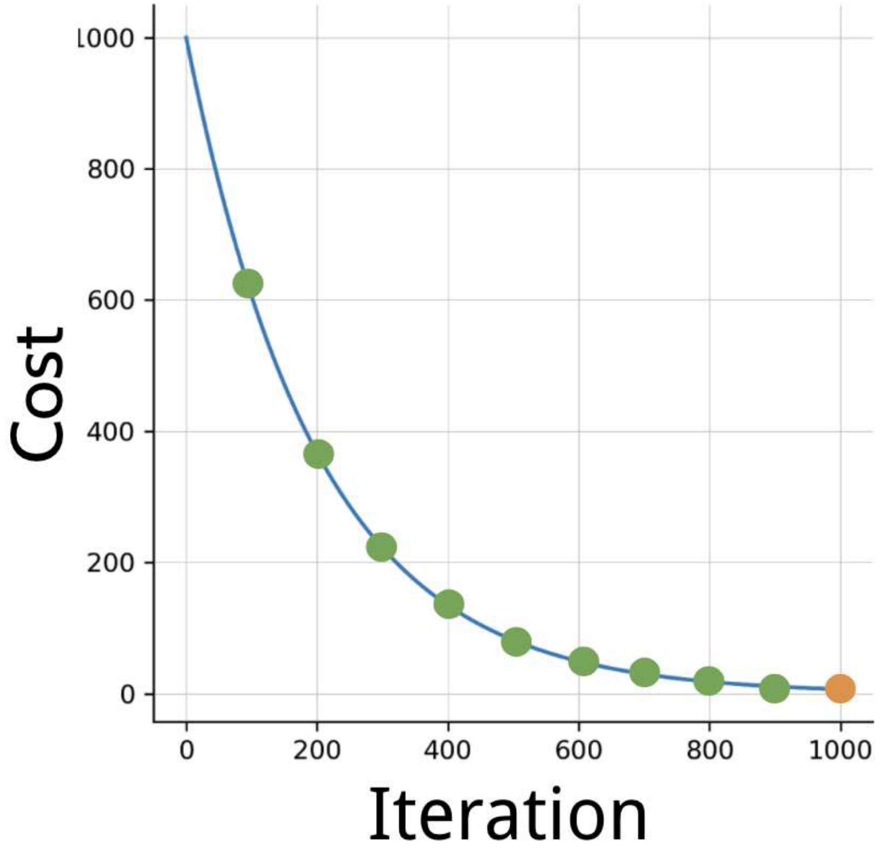
↓

$$x^{(i)} = \begin{bmatrix} 1 \\ 3476 \\ 245 \end{bmatrix} \quad \theta = \begin{bmatrix} 0.00003 \\ 0.00150 \\ -0.00120 \end{bmatrix}$$



Logistic Regression: Training





Logistic Regression: Testing

- X_{val} Y_{val} θ

$$h(X_{val}, \theta)$$

$$pred = h(X_{val}, \theta) \geq 0.5 \qquad \begin{bmatrix} 0.3 \\ 0.8 \\ 0.5 \\ \vdots \\ h_m \end{bmatrix} \geq 0.5 = \begin{bmatrix} 0.3 \geq 0.5 \\ 0.8 \geq 0.5 \\ 0.5 \geq 0.5 \\ \vdots \\ pred_m \geq 0.5 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ \vdots \\ pred_m \end{bmatrix}$$

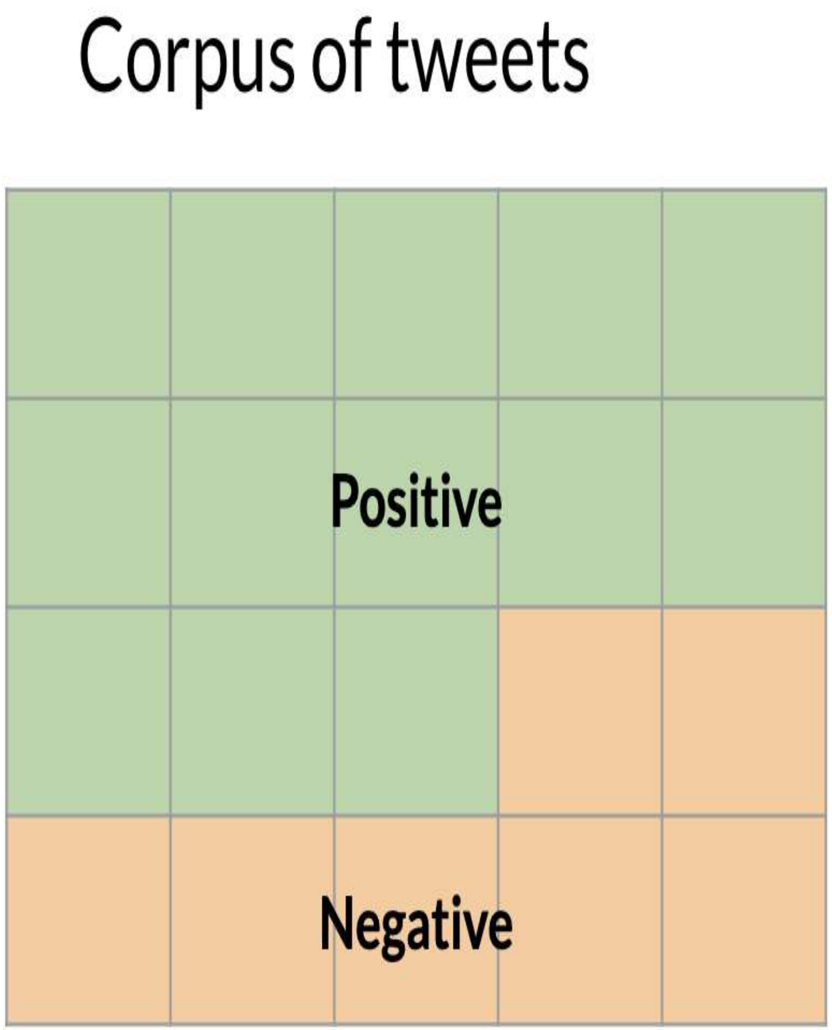
Accuracy $\longrightarrow \sum_{i=1}^{m} \frac{(pred^{(i)} == y_{val}^{(i)})}{m}$

Sentiment Analysis

Using Naïve Bayes

Compiled by:
Dr. Amit Kumar Trivedi
Department of Computer Science and Engineering
TIET, Patiala
Source of slide: DeepLearning.AI

Probability and Bayes' Rule

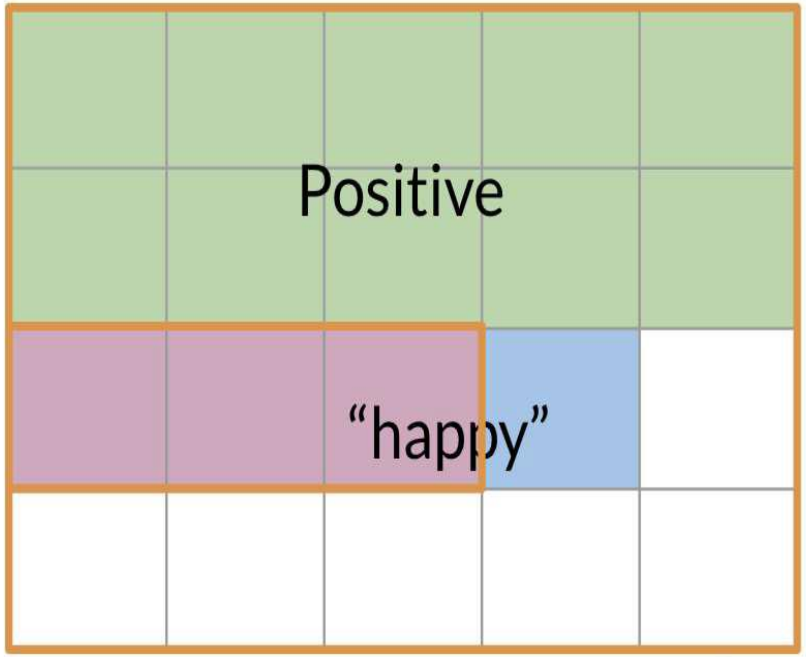


A → Positive tweet

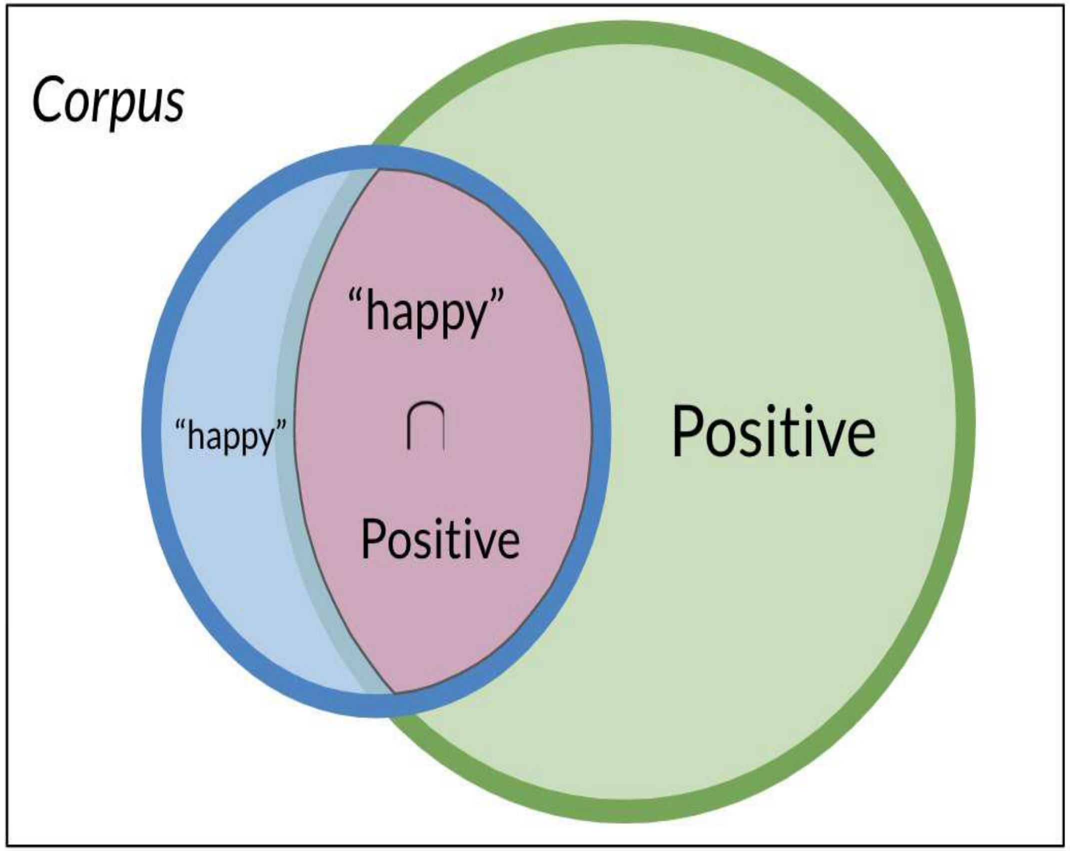
$$P(A) = N_{pos} / N = 13 / 20 = 0.65$$

$$P(Negative) = 1 - P(Positive) = 0.35$$

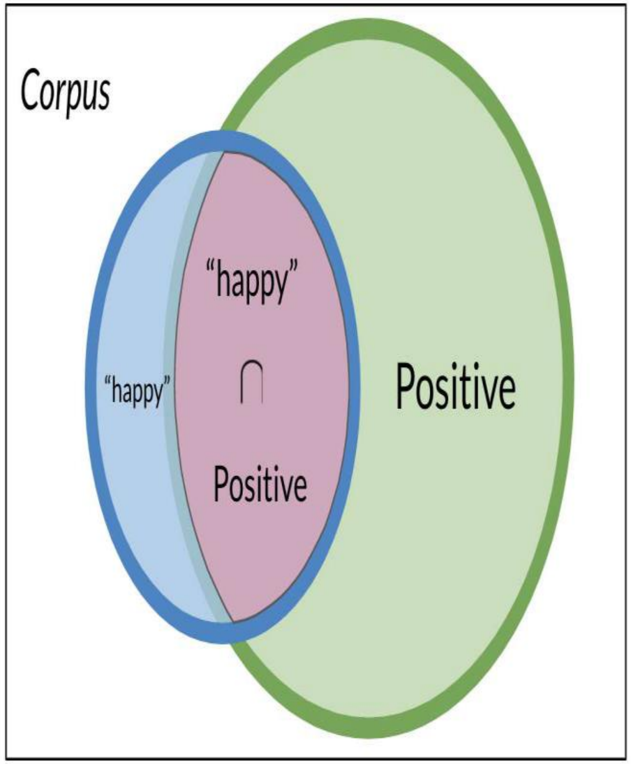
Probability and Bayes' Rule



$$P(A \cap B) = P(A, B) = \frac{3}{20} = 0.15$$



Probability and Bayes' Rule



$$P(\text{Positive}|\text{“happy”}) = \frac{P(\text{Positive} \cap \text{“happy”})}{P(\text{“happy”})}$$

$$P(\text{Positive}|\text{“happy”}) = \frac{\boxed{P(\text{Positive} \cap \text{“happy”})}}{P(\text{“happy”})}$$

$$P(\text{“happy”}|\text{Positive}) = \frac{\boxed{P(\text{“happy”} \cap \text{Positive})}}{P(\text{Positive})}$$

$$\boxed{P(\text{Positive}|\text{“happy”}) =} \boxed{P(\text{“happy”}|\text{Positive})} \boxed{\times \frac{P(\text{Positive})}{P(\text{“happy”})}}$$

Naïve Bayes Introduction

Positive tweets

I am happy because I am learning NLP

I am happy, not sad.

Negative tweets

I am sad, I am not learning NLP

I am sad, not happy

word	Pos	Neg
I	3	3
am	3	3
happy	2	1
because	1	0
learning	1	1
NLP	1	1
sad	1	2
not	1	2
N_{class}	13	12

word	Pos	Neg
I	0.24	0.25
am	0.24	0.25
happy	0.15	0.08
because	0.08	0
learning	0.08	0.08
NLP	0.08	0.08
sad	0.08	0.17
not	0.08	0.17

Naïve Bayes Introduction

Tweet: I am happy today; I am learning.

$$\prod_{i=1}^{m} \frac{P(w_i|pos)}{P(w_i|neg)} = \frac{0.14}{0.10} = 1.4 \quad > 1$$

$$\cancel{\frac{0.20}{0.20}} * \cancel{\frac{0.20}{0.20}} * \boxed{\frac{0.14}{0.10}} * \cancel{\frac{0.20}{0.20}} * \cancel{\frac{0.20}{0.20}} * \cancel{\frac{0.10}{0.10}}$$

word	Pos	Neg
I	0.20	0.20
am	0.20	0.20
happy	0.14	0.10
because	0.10	0.05
learning	0.10	0.10
NLP	0.10	0.10
sad	0.10	0.15
not	0.10	0.15

Laplacian Smoothing

We usually compute the probability of a word given a class as follows:

$$P\left(\mathrm{w_i} \mid \text{class}\right) = \frac{\text{freq}\left(\mathrm{w_i}, \text{class}\right)}{\mathrm{N_{class}}} \qquad \text{class} \in \{\text{Positive}, \text{Negative}\}$$

However, if a word does not appear in the training, then it automatically gets a probability of 0, to fix this we add smoothing as follows

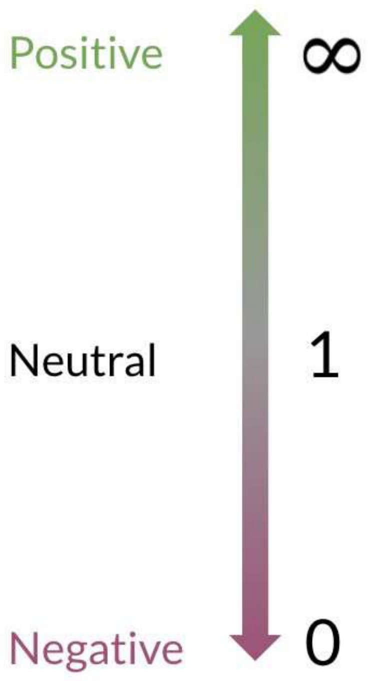
$$P\left(\mathrm{w_i} \mid \text{class}\right) = \frac{\text{freq}(\mathrm{w_i}, \text{class}) + 1}{(\mathrm{N_{class}} + \mathrm{V})}$$

Note that we added a 1 in the numerator, and since there are V words to normalize, we add V in the denominator.

N_{class}: frequency of all words in class

V: number of unique words in vocabulary

Log Likelihood



word	Pos	Neg	ratio
I	0.19	0.20	
am	0.19	0.20	
happy	0.14	0.10	
because	0.10	0.05	
learning	0.10	0.10	
NLP	0.10	0.10	
sad	0.10	0.15	
not	0.10	0.15	

$$\text{ratio}(w_i) = \frac{P(w_i \mid \text{Pos})}{P(w_i \mid \text{Neg})}$$

$$\approx \frac{freq(w_i, 1) + 1}{freq(w_i, 0) + 1}$$

Log Likelihood

To do inference, you can compute the following:

$$\frac{P(pos)}{P(neg)} \prod_{i=1}^{m} \frac{P(w_i|pos)}{P(w_i|neg)} > 1$$

As m gets larger, we can get numerical flow issues, so we introduce the $\log$, which gives you the following equation:

$$\log\left(\frac{P(pos)}{P(neg)} \prod_{i=1}^{n} \frac{P(w_i|pos)}{P(w_i|neg)}\right) \Rightarrow \log\frac{P(pos)}{P(neg)} + \sum_{i=1}^{n} \log\frac{P(w_i|pos)}{P(w_i|neg)}$$

The first component is called the log prior and the second component is the log likelihood. We further introduce λ as follows:

Log Likelihood

doc: I am happy because I am learning.

$$\lambda(w) = log\frac{P(w|pos)}{P(w|neg)}$$

$$\lambda(\text{happy}) = log\frac{0.09}{0.01} \approx 2.2$$

word	Pos	Neg	λ
I	0.05	0.05	0
am	0.04	0.04	0
happy	0.09	0.01	
because	0.01	0.01	
learning	0.03	0.01	
NLP	0.02	0.02	
sad	0.01	0.09	
not	0.02	0.03	

Log Likelihood

doc: I am happy because I am learning.

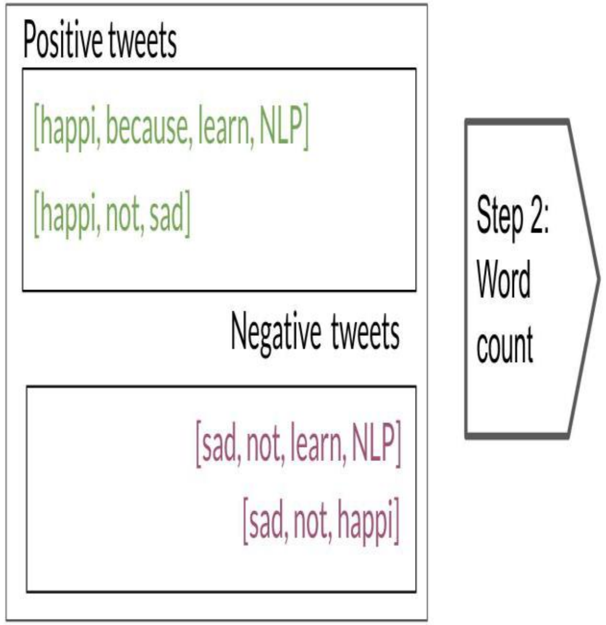
$$\sum_{i=1}^{m} log \frac{P(w_i|pos)}{P(w_i|neg)} = \sum_{i=1}^{m} \lambda(w_i)$$

log likelihood = 0 + 0 + 2.2 + 0 + 0 + 0 + 1.1 = **3.3**

word	Pos	Neg	λ
I	0.05	0.05	0
am	0.04	0.04	0
happy	0.09	0.01	2.2
because	0.01	0.01	0
learning	0.03	0.01	1.1
NLP	0.02	0.02	0
sad	0.01	0.09	-2.2
not	0.02	0.03	-0.4

Training naïve Bayes

1. Get or annotate a dataset with positive and negative tweets
2. Pre-process the tweets to get [w1, w2, w3, …]
3. Compute freq(w, class)
4. Get P(w|pos), P(w|neg)
5. Get λ(w)
6. Compute logprior=log(P(pos)/P(neg))



word	Pos	Neg
happi	2	1
because	1	0
learn	1	1
NLP	1	1
sad	1	2
not	1	2
N_{class}	7	7

freq(w, class)

Applications of Naïve Bayes

There are many applications of naive Bayes including:

- Author identification
- Spam filtering
- Information retrieval
- Word disambiguation

This method is usually used as a simple baseline. It is also really fast.