

Copyright Notice

These slides are distributed under the Creative Commons License.

[DeepLearning.AI](#) makes these slides available for educational purposes. You may not use or distribute these slides for commercial purposes. You may make copies of these slides and use or distribute them for educational purposes as long as you cite [DeepLearning.AI](#) as the source of the slides.

For the rest of the details of the license, see <https://creativecommons.org/licenses/by-sa/2.0/legalcode>



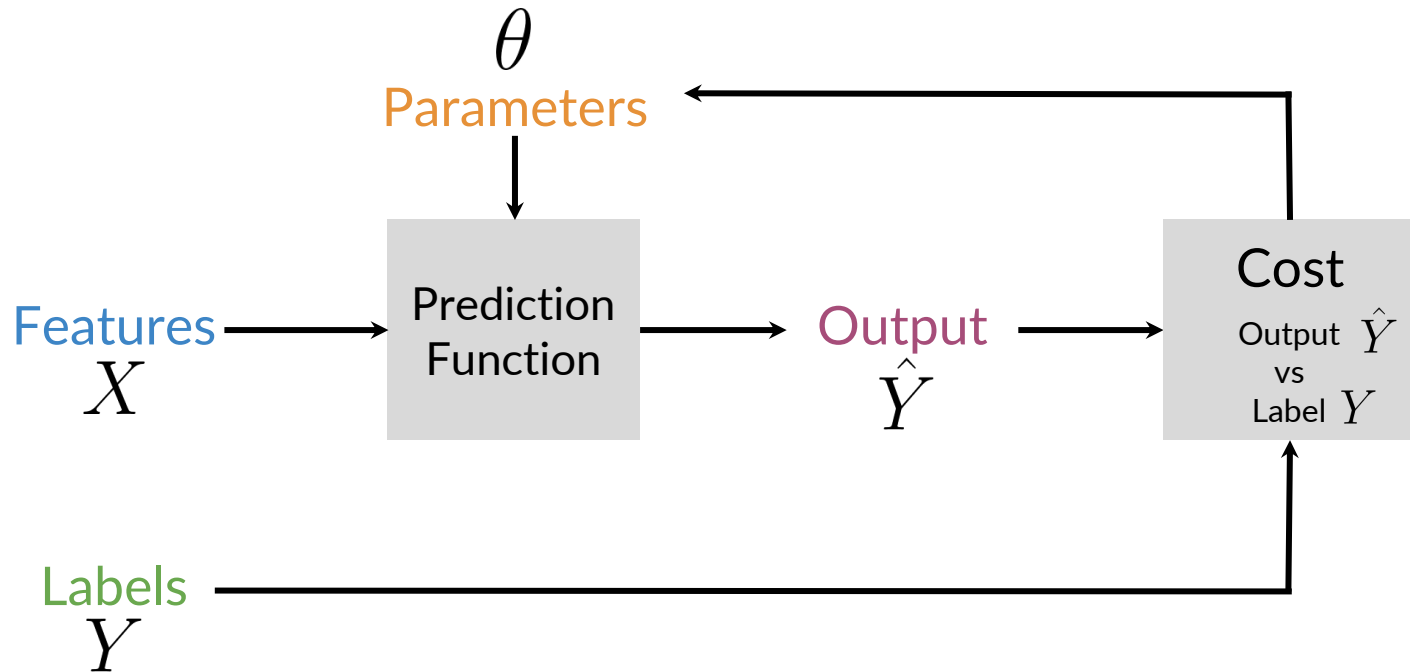
deeplearning.ai

Supervised ML and Sentiment Analysis

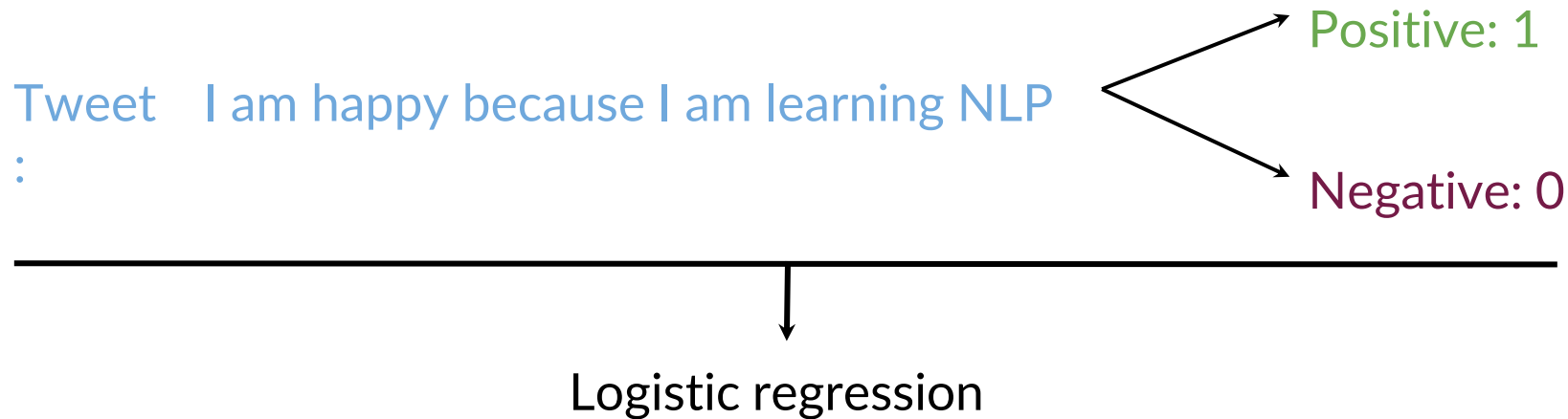
Outline

- Review Supervised ML
- Build your own tweet classifier!

Supervised ML (training)

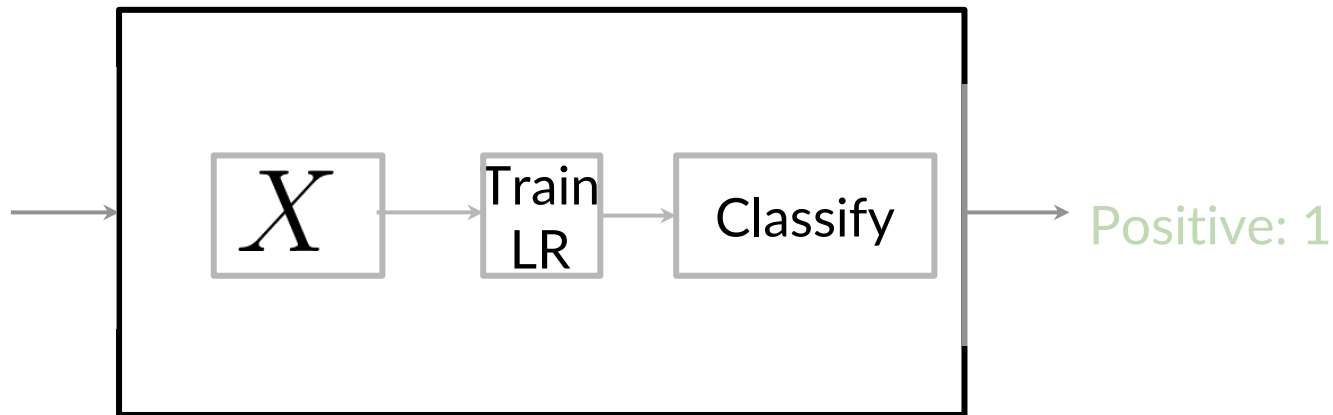


Sentiment analysis



Sentiment analysis

I am happy
because I am
learning NLP



Summary

- Features, Labels \longrightarrow Train \longrightarrow Predict
- Extract features \longrightarrow Train-LR \longrightarrow Predict sentiment



deeplearning.ai

Vocabulary and Feature Extraction

Outline

- Vocabulary
- Feature extraction
- Sparse representations and some of their issues

Vocabulary

Tweets:

[tweet_1, tweet_2, ..., tweet_m]



I am happy because I am learning
NLP
...
I hated the movie

$V =$

[I, am, happy, because, learning, NLP, ... hated, the, movie]

Feature extraction

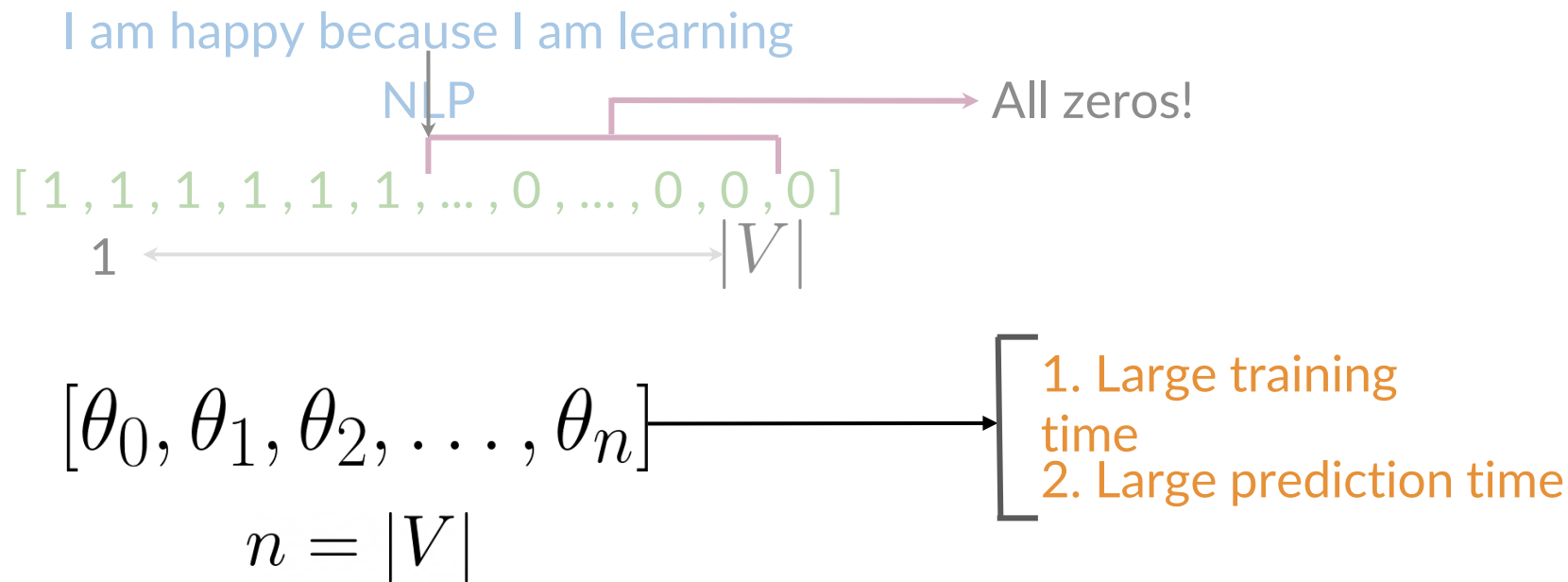
I am happy because I am learning NLP

[I , am, happy, because, learning, NLP, ... hated, the, movie]

↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
[1, 1, 1, 1, 1, 1,						...	0,	0,	0]

A lot of zeros! That's a sparse representation.

Problems with sparse representations



Summary

- Vocabulary: set of unique words
- Vocabulary, Text \longrightarrow [1 0 1 .. 0 .. 1 .. 0]
- Sparse representations are problematic for training and prediction times



deeplearning.ai

Negative and Positive Frequencies

Outline

- Populate your vocabulary with a frequency count for each class

Positive and negative counts

Corpus

I am happy because I am learning

NLP
I am happy

I am sad, I am not learning NLP

I am sad

Vocabulary

I

am

happy

because

learning

NLP

sad

not

Positive and negative counts

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

Positive and negative counts

Positive tweets

I am happy because I am learning

NLP
I am happy

Vocabulary	PosFreq (1)
------------	-------------

I

am

happy

because

learning

NLP

sad

not

0

Positive and negative counts

Vocabulary	NegFreq (0)
1	
am	
happy	
because	
learning	
NLP	
sad	
not	1

Negative tweets

I am sad, I am not learning NLP

I am sad

Word frequency in classes

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

Summary

- Divide tweet corpus into two classes: positive and negative
 - Count each time each word appears in either class
- Feature extraction for training and prediction!



deeplearning.ai

Feature extraction with frequencies

Outline

- Extract features from your frequencies dictionary to create a features vector

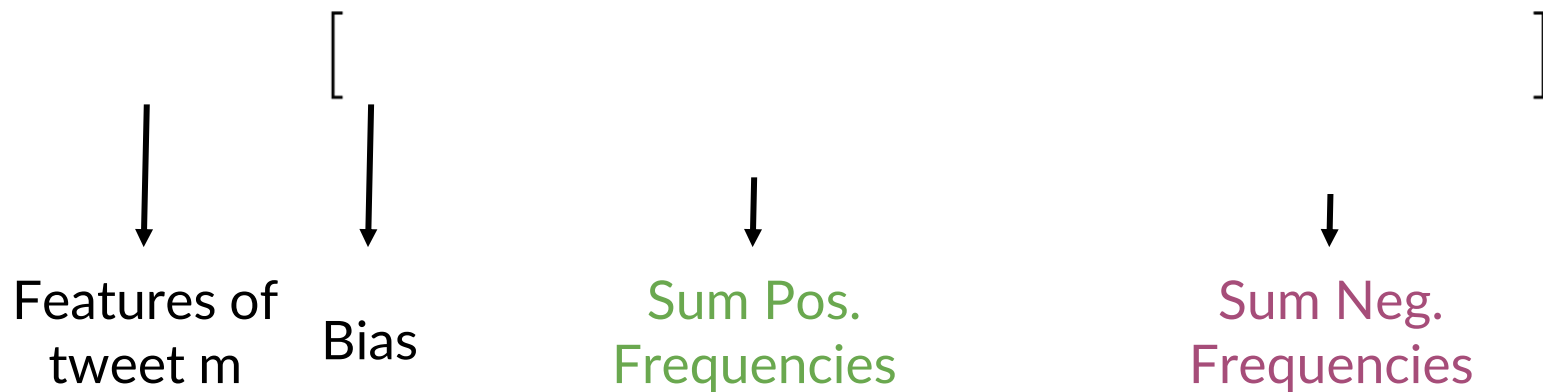
Word frequency in classes

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

Feature extraction

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

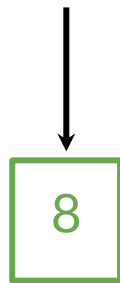


Feature extraction

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

I am sad, I am not learning NLP

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



Feature extraction

Vocabulary	NegFreq (0)
I	<u>3</u>
am	<u>3</u>
happy	0
because	0
learning	<u>1</u>
NLP	<u>1</u>
sad	<u>2</u>
not	<u>1</u>

I am sad, I am not learning NLP

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

↓
11

Feature extraction

I am sad, I am not learning NLP

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



$$X_m = [1, 8, 11]$$

Summary

- Dictionary mapping (word,class) to frequencies

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

→ Cleaning unimportant information from your tweets



deeplearning.ai

Preprocessing

Outline

- Removing stopwords, punctuation, handles and URLs
- Stemming
- Lowercasing

Preprocessing: stop words and punctuation

@YMourri and @AndrewYNg are
tuning a GREAT AI model at
<https://deeplearning.ai!!!>

Stop words	Punctuation
and	,
is	.
are	:
at	!
has	"
for	'
a	

Preprocessing: stop words and punctuation

@YMourri and @AndrewYNg are
tuning a GREAT AI model at
<https://deeplearning.ai!!!>

@YMourri @AndrewYNg tuning
GREAT AI model
<https://deeplearning.ai!!!>

Stop words

and

is

are

at

has

for

a

Punctuation

,

.

:

!

“

‘

Preprocessing: stop words and punctuation

@YMourri @AndrewYNg tuning
GREAT AI model
<https://deeplearning.ai!!!>

@YMourri @AndrewYNg tuning
GREAT AI model
<https://deeplearning.ai>

Stop words

and
is
a
at
has
for
of

Punctuation

,
.
:
!
"
'

Preprocessing: Handles and URLs

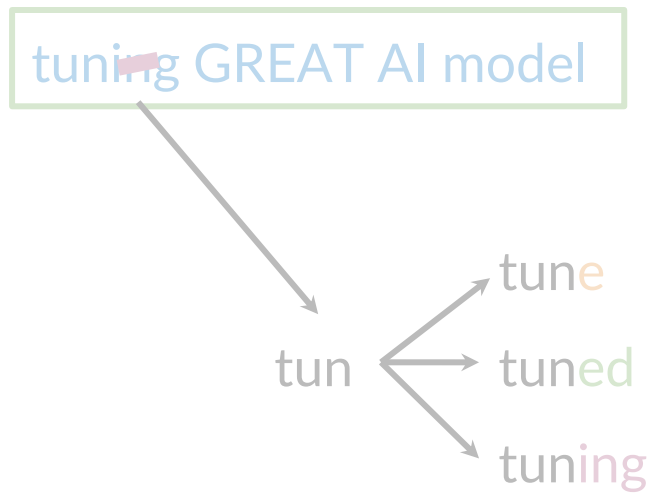
~~@YMurri @AndrewYNg tuning GREAT AI
model~~

<https://deeplearning.ai>



tuning GREAT AI model

Preprocessing: Stemming and lowercasing



Preprocessed tweet:
[tun, great, ai, model]

Summary

- Stop words, punctuation, handles and URLs
- Stemming
- Lowercasing
- Less unnecessary info → Better times



deeplearning.ai

Putting it all together

Outline

- Generalize the process
- How to code it!

General overview

I am Happy Because i am learning NLP @deeplearning

↓ Preprocessing

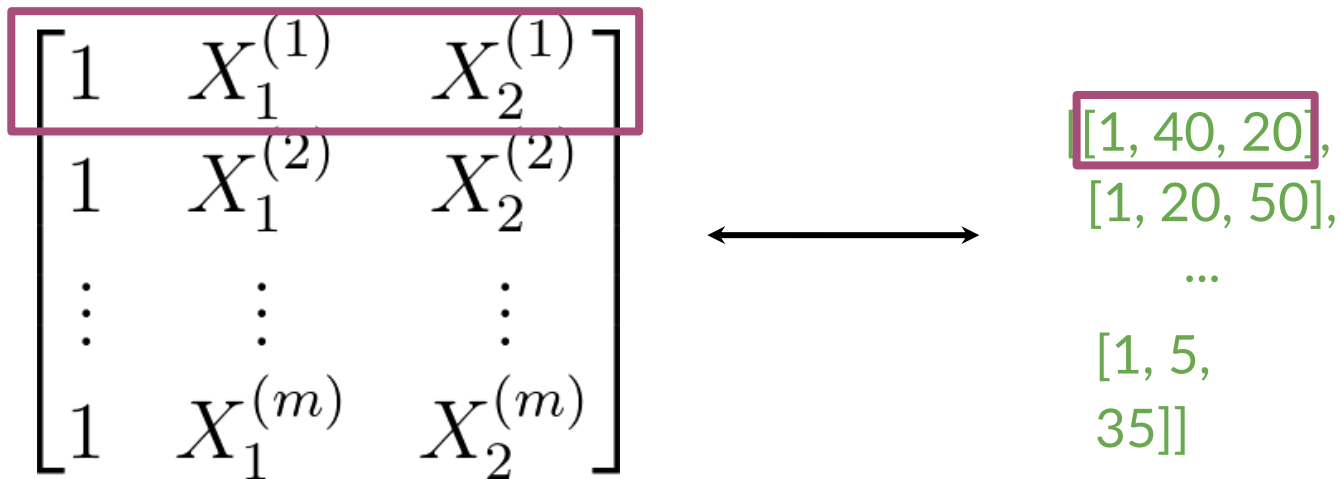
[happy, learn, nlp]

↓ Feature Extraction

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

↓
Sum positive frequencies

General overview



General Implementation

```
freqs = build_freqs(tweets, labels) #Build frequencies dictionary
X = np.zeros((m, 3)) #Initialize matrix X
for i in range(m): #For every tweet
    p_tweet = process_tweet(tweets[i]) #Process tweet
    X[i, :] = extract_features(p_tweet, freqs) #Extract Features
```

Summary

- Implement the feature extraction algorithm for your entire set of tweets
- Almost ready to train!



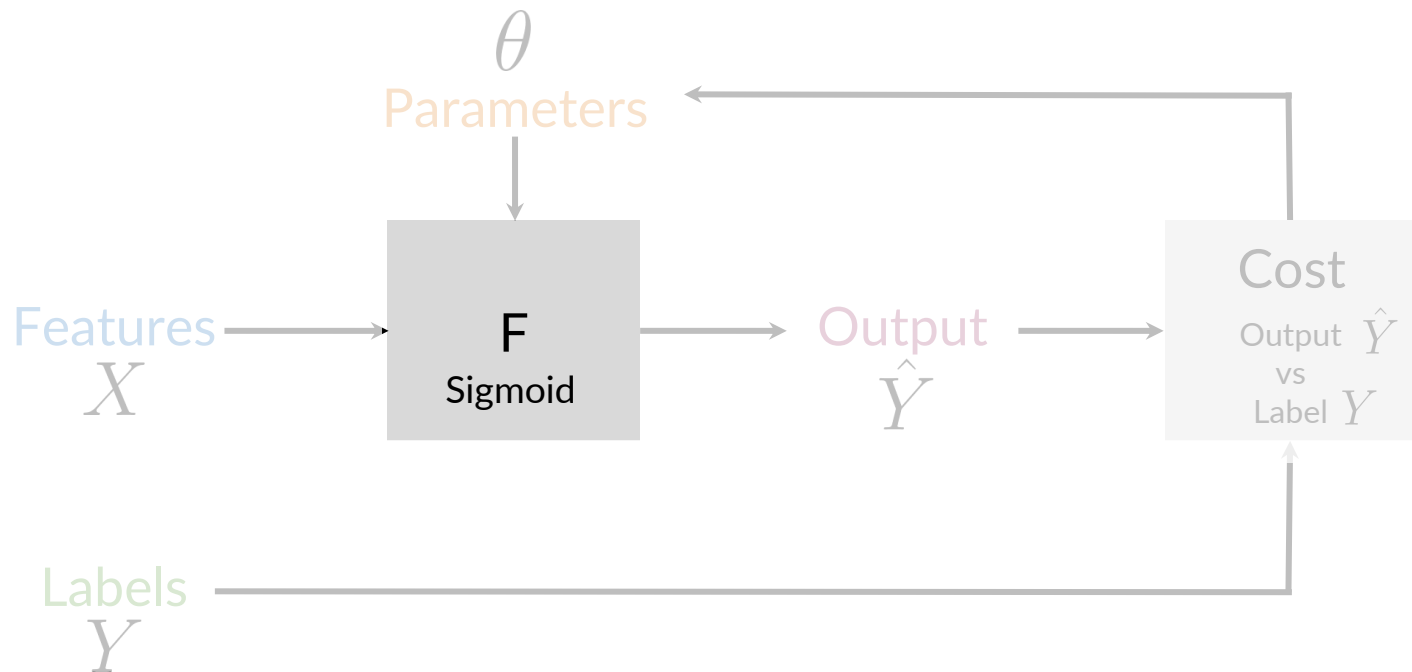
deeplearning.ai

Logistic Regression Overview

Outline

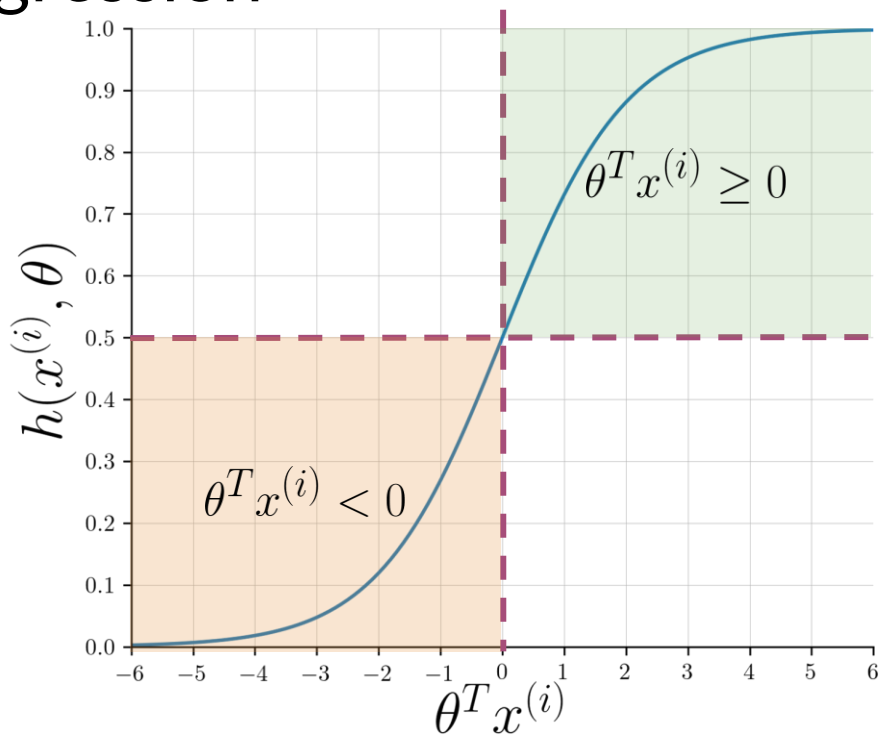
- Supervised learning and logistic regression
- Sigmoid function

Overview of logistic regression



Overview of logistic regression

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

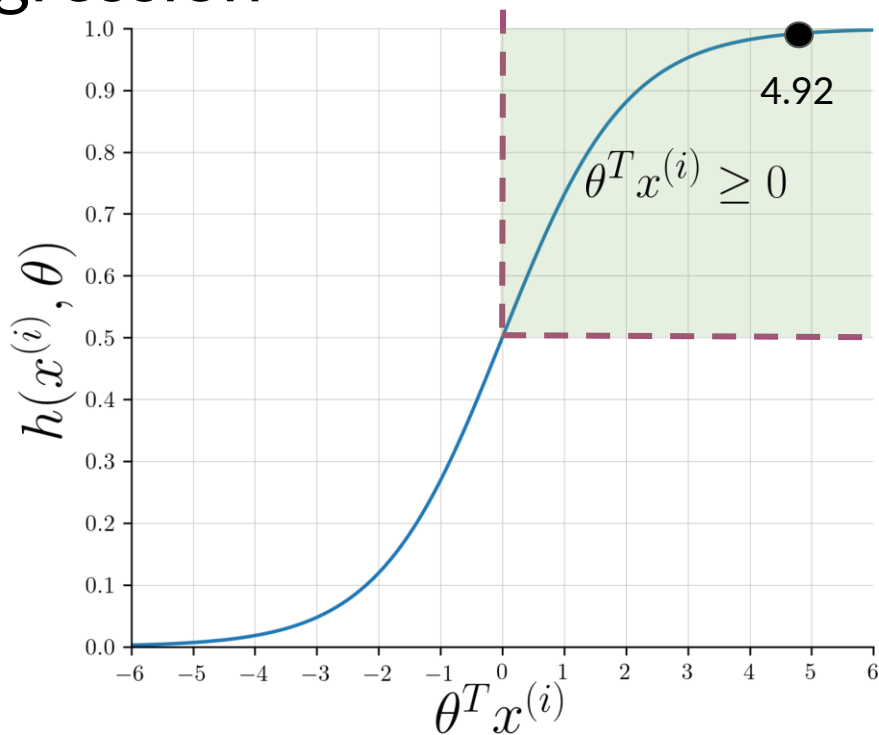


Overview of logistic regression

@YMourri and
@AndrewYNg are tuning a
GREAT AI model

[tun, ai, great,
model]

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



Summary

- Sigmoid function
- $\theta^T x^{(i)} \geq 0 \longrightarrow h(x^{(i)}, \theta) \geq 0.5$, positive
- $\theta^T x^{(i)} < 0 \longrightarrow h(x^{(i)}, \theta) < 0.5$, negative



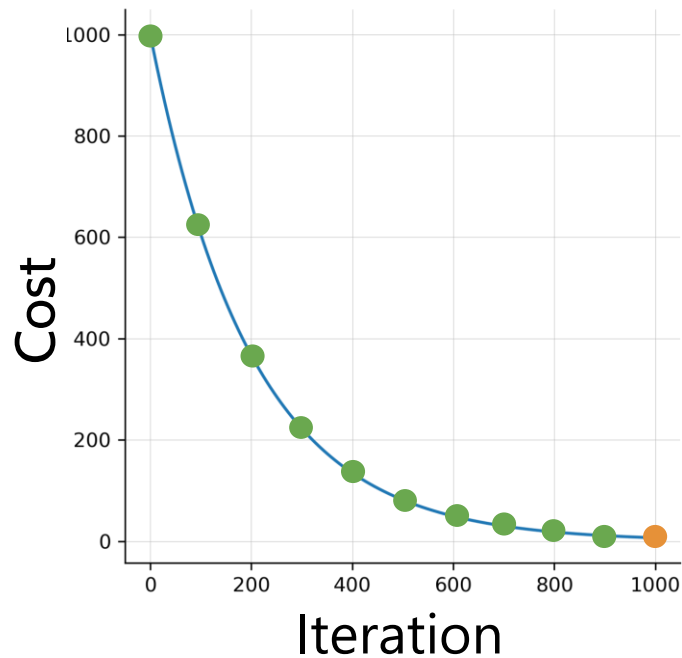
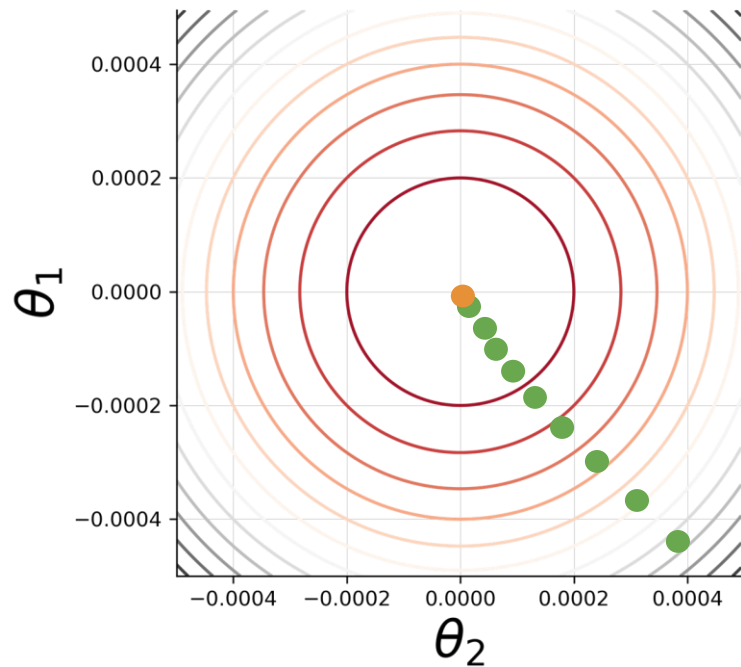
deeplearning.ai

Logistic Regression: Training

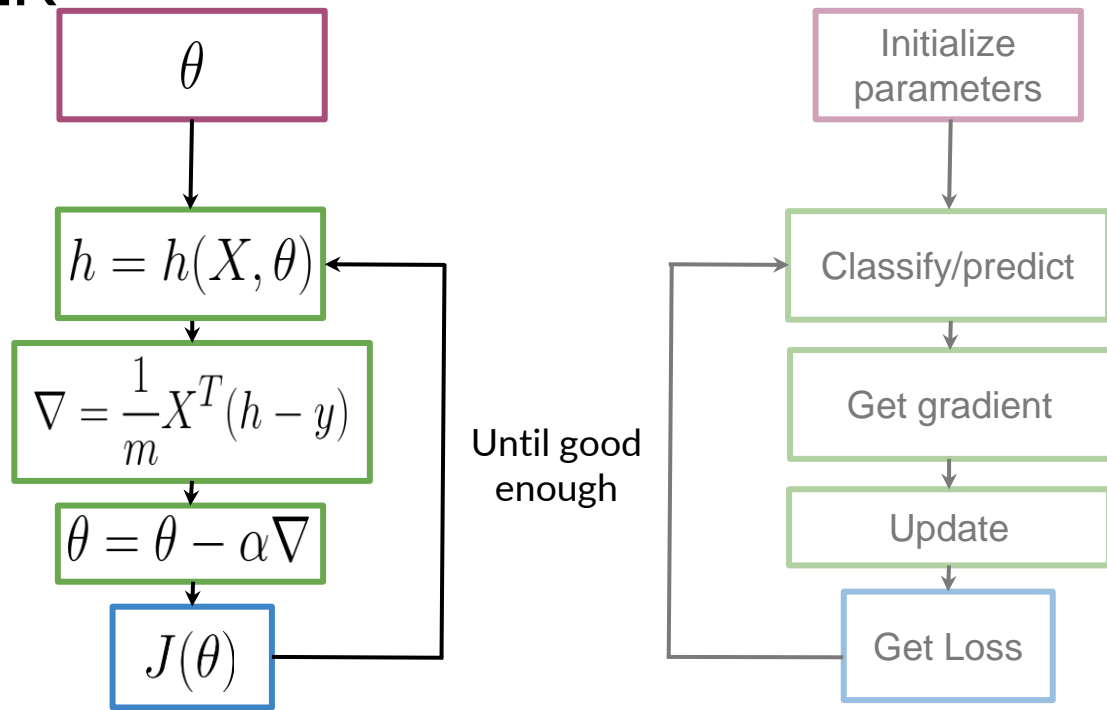
Outline

- Review the steps in the training process
- Overview of gradient descent

Training LR



Training LR



Summary

- Visualize how gradient descent works
 - Use gradient descent to train your logistic regression classifier
- Compute the accuracy of your model



deeplearning.ai

Logistic Regression: Testing

Outline

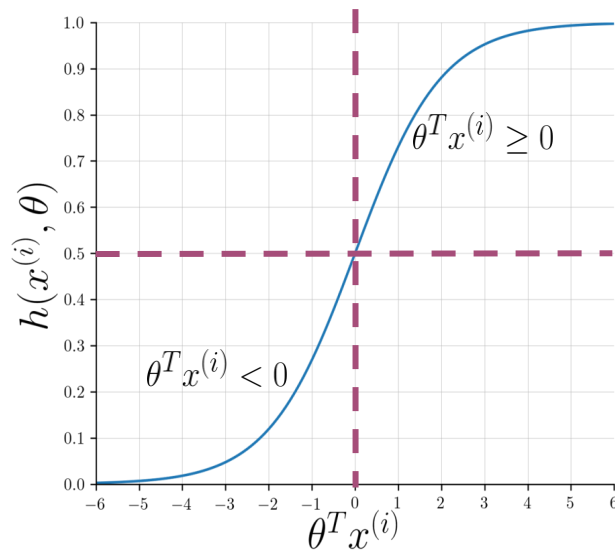
- Using your validation set to compute model accuracy
- What the accuracy metric means

Testing logistic regression

- X_{val} Y_{val} θ

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

$$pred = h(X_{val}, \theta) \geq 0.5$$



Testing logistic regression

- X_{val} Y_{val} θ

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

$$pred = h(X_{val}, \theta) \geq 0.5$$

$$\begin{bmatrix} 0.3 \\ 0.8 \\ 0.5 \\ \vdots \\ h_m \end{bmatrix} \geq 0.5 = \begin{bmatrix} \underline{0.3 \geq 0.5} \\ \underline{0.8 \geq 0.5} \\ \underline{0.5 \geq 0.5} \\ \vdots \\ pred_m \geq 0.5 \end{bmatrix} = \begin{bmatrix} \underline{0} \\ \underline{1} \\ \underline{1} \\ \vdots \\ pred_m \end{bmatrix}$$

Testing logistic regression

- $X_{val} \ Y_{val} \ \theta$

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

$$pred = h(X_{val}, \theta) \geq 0.5$$

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

$$\begin{bmatrix} \frac{0}{1} \\ 1 \\ \vdots \\ pred_m \end{bmatrix} == \begin{bmatrix} \frac{0}{0} \\ 1 \\ \vdots \\ Y_{val_m} \end{bmatrix}$$

$$\begin{bmatrix} \frac{1}{0} \\ 1 \\ \vdots \\ pred_m == Y_{val_m} \end{bmatrix}$$

Testing logistic regression

$$Y_{val} = \begin{bmatrix} 0 \\ 1 \\ \underline{1} \\ 0 \\ 1 \end{bmatrix} \quad pred = \begin{bmatrix} 0 \\ 1 \\ \underline{0} \\ 0 \\ 1 \end{bmatrix}$$

$$(Y_{val} == pred) = \begin{bmatrix} 1 \\ 1 \\ \underline{0} \\ 1 \\ 1 \end{bmatrix}$$

$$\text{accuracy} = \frac{4}{5} = 0.8$$

Summary

- $X_{val} \ Y_{val} \longrightarrow$ Performance on unseen data
- Accuracy $\longrightarrow \sum_{i=1}^m \frac{(pred^{(i)} == y_{val}^{(i)})}{m}$

To improve model: step size, number of iterations, regularization, new features, etc.



deeplearning.ai

Logistic Regression: Cost Function

Outline

- Overview of the logistic cost function, AKA the binary cross-entropy function

Cost function for logistic regression

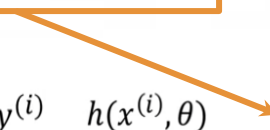
$$J(\theta) = -\frac{1}{m} \sum_{i=1}^m [y^{(i)} \log h(x^{(i)}, \theta) + (1 - y^{(i)}) \log(1 - h(x^{(i)}, \theta))]$$

Cost function for logistic regression

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

Cost function for logistic regression


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



$y^{(i)}$	$h(x^{(i)}, \theta)$	
0	any	0
1	0.99	~0
1	~0	-inf

Cost function for logistic regression

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



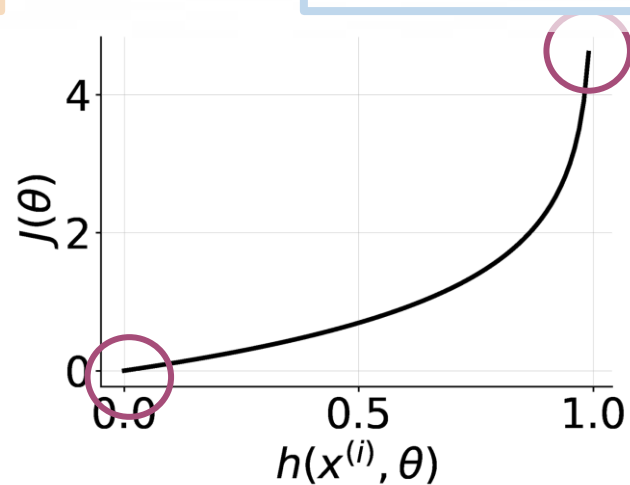
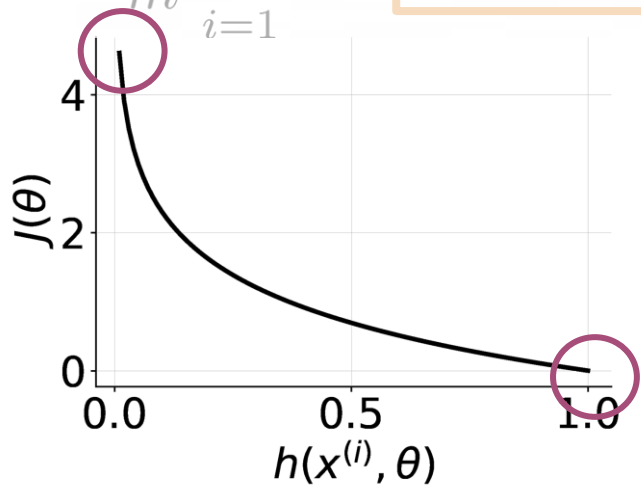
$y^{(i)}$	$h(x^{(i)}, \theta)$	
1	any	0
0	0.01	~0
0	~1	-inf

Cost function for logistic regression

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

Cost function for logistic regression

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



Summary

- Strong disagreement = high cost
- Strong agreement = low cost
- Aim for the lowest cost!