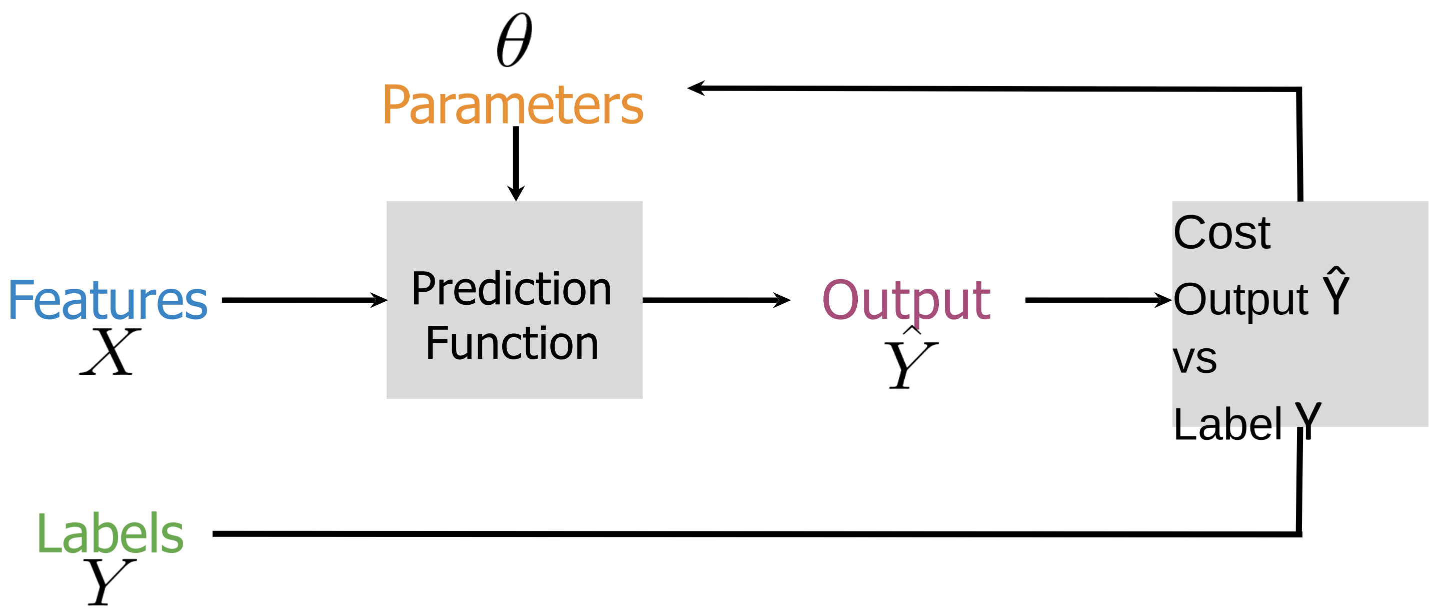
Classification and Vector Spaces

FPT Education
FPT UNIVERSITY

- Logistic regression
  - Supervised Machine Learning & Sentiment Analysis
  - Feature Extraction
  - Preprocessing
  - Training, Testing Logistic Regression

Supervised Machine Learning



- Input features X and a set of labels Y; minimize your error rates or cost
- Run the prediction function which takes in parameters data to map features to output labels Ŷ
- Calculate the cost function F does this by comparing how closely the output Ŷ is to the label Y.
- Update the parameters and repeat the whole process until your cost is minimized

Sentiment analysis

FPT Education
FPT UNIVERSITY

Tweet: I am happy because I am learning NLP

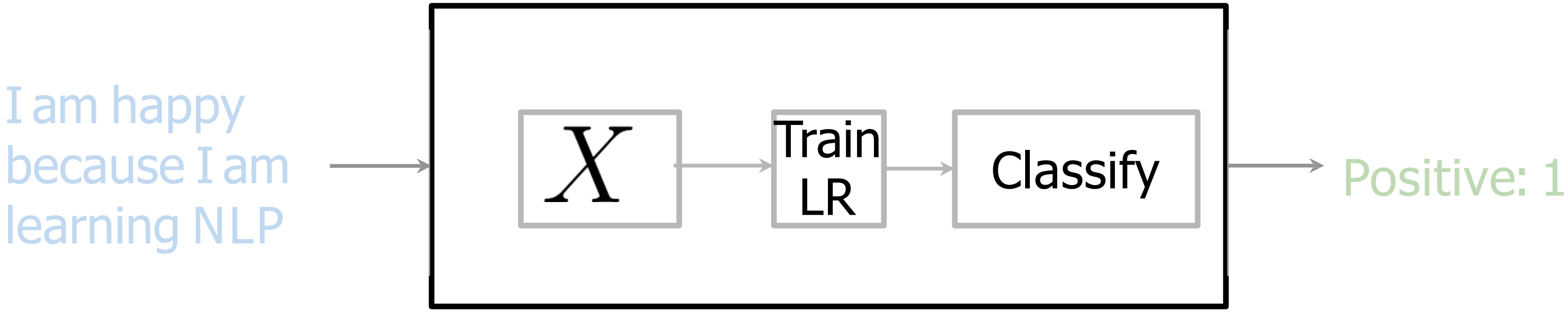
Positive: 1

Negative: 0

Logistic regression

- How can predict whether a tweeta has a positive or a negative sentiment.
- Training set where tweets with a positive sentiment have a label of one, and the tweets with a negative sentiment have a label of zero
- Logistic regression classifierwhich assigns its observations to two distinct classes.

Sentiment analysis



- First process the raw tweets in your training sets and extract useful features
- Train your logistic regression classifier while minimizing the cost
- Make your prediction

Vocabulary and Feature Extraction

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

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

FPT Education

FPT UNIVERSITY

- Sparse Represenation
  - To represent a text as a vector, we have to build a vocabulary and that will allow to encode any text or any tweet as an array of numbers
  - The vocabulary V would be the list of unique words from your list of tweets.
  - Sparse Represenation assign a value of 1 to that a word of a tweet,
  - If it doesn't appear in the vocabulary V we assign a value of 0.

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

I am happy because I am learning NLP

All zeros!

[1,1,1,1,1,1,…,0,…,0,0,0]

1 $|V|$

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

$$n = |V|$$

1. Large training time
2. Large prediction time

- A logistic regression model would have to learn N+1 parameters, where N is the size of the vocabulary V
- Large training time
- Large prediction time

Negative and Positive Frequencies
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

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

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

Word frequency in classes

FPT Education
FPT UNIVERSITY

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

- Divide tweet corpus into two classes: positive and negative
- Count each time each word appears in either class

=>Feature extraction for training and prediction!

Feature extraction with frequencies

FPT Education
FPT UNIVERSITY

- Word frequency in classes
  - Extract features from your frequencies dictionary to create a features vector

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

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

- X_m → Features of tweet m
- 1 → Bias
- $\sum_w freqs(w, 1)$ → Sum Pos. Frequencies
- $\sum_w freqs(w, 0)$ → Sum Neg. Frequencies

Feature extraction

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

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

$$\sum_w freqs(w, 1) \rightarrow \boxed{8}$$

Feature extraction

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

I am sad, I am not learning NLP

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

$$\boxed{11}$$

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

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

Preprocessing

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

Preprocessing: stop words and punctuation

FPT Education
FPT UNIVERSITY

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

Stop words
and
is
are
at
has
for
a

Punctuation
,
.
:
!
“
‘

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

Preprocessing: Handles and URLs

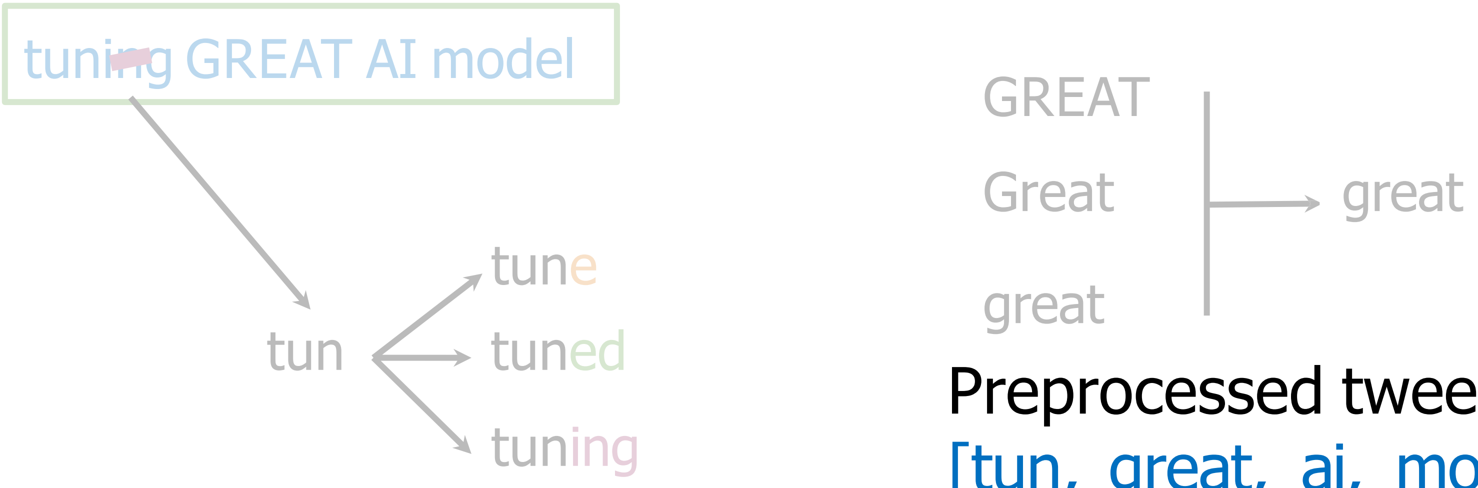
FPT Education
FPT UNIVERSITY

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

tuning GREAT AI model

@ deeplearning.ai

Preprocessing: Stemming and lowercasing



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

- Stemming in NLP is simply transforming any word to its base stem, which you could define as the set of characters that are used to construct the word and its derivatives.

Putting it together

- General overview

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

General overview

I am Happy Because i am learning NLP @deeplearning		[happy, learn, nlp]		[[1, 40, 20],
I am sad not learning NLP	→	[sad, not, learn, nlp]	→	[1, 20, 50],
...		...		...
I am sad :(		[sad]		[1, 5, 35]]

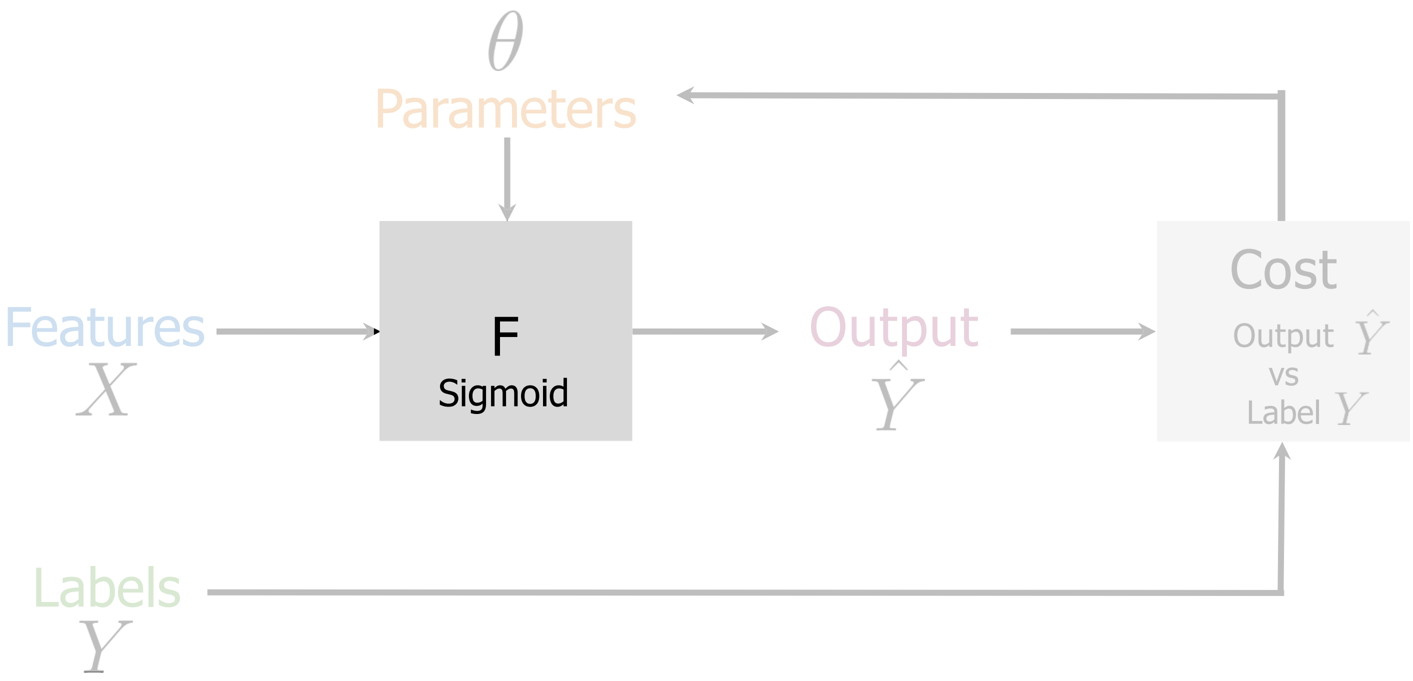
General Implementation

```
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
```

- Implement the feature extraction algorithm for your entire set of tweets

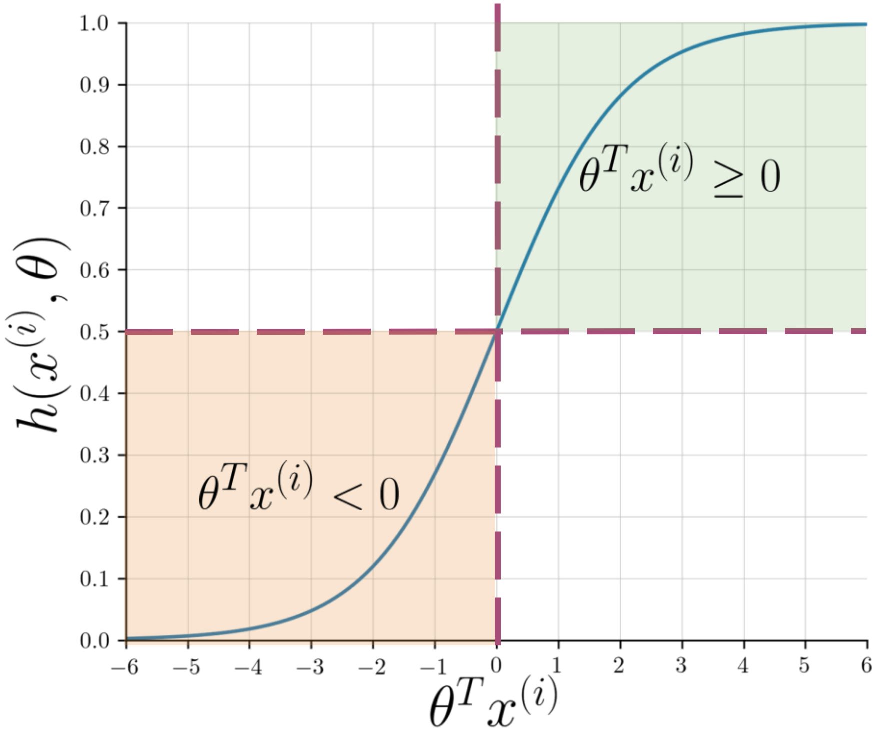
Overview of Logistic Regression

- Overview of logistic regression



Overview of logistic regression

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

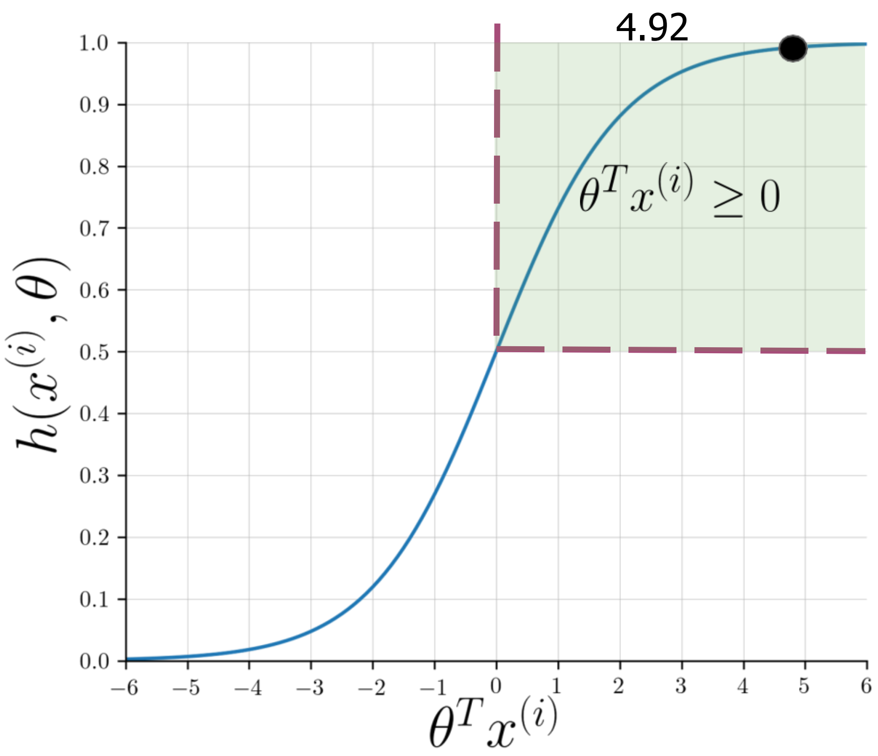


Overview of logistic regression

@YMourri and @AndrewYNg are tuning a GREAT AI model

[tun, ai, great, model]

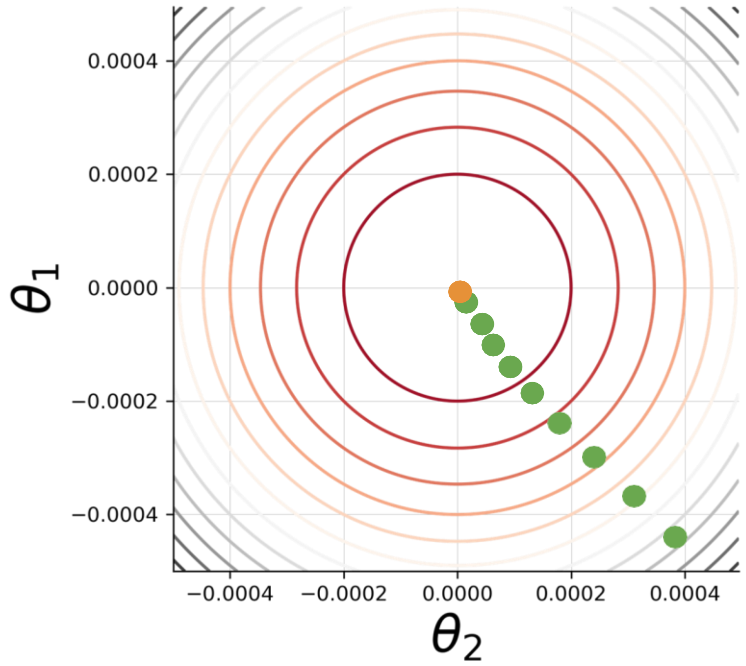
$$x^{(i)} = \begin{bmatrix} 1 \\ 3476 \\ 245 \end{bmatrix} \quad \theta = \begin{bmatrix} 0.00003 \\ 0.00150 \\ -0.00120 \end{bmatrix}$$

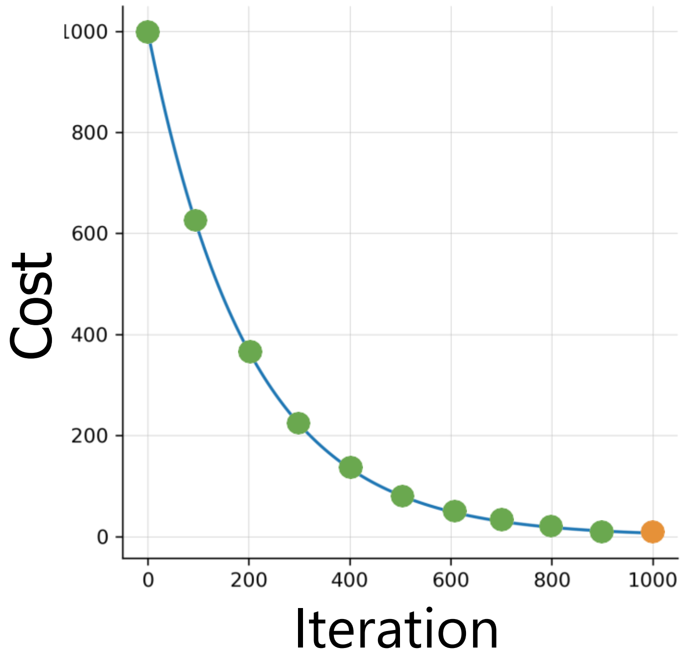


$\theta^T x^{(i)} \geq 0 \longrightarrow h(x^{(i)}, \theta) \geq 0.5$, positive

$\theta^T x^{(i)} < 0 \longrightarrow h(x^{(i)}, \theta) < 0.5$, negative

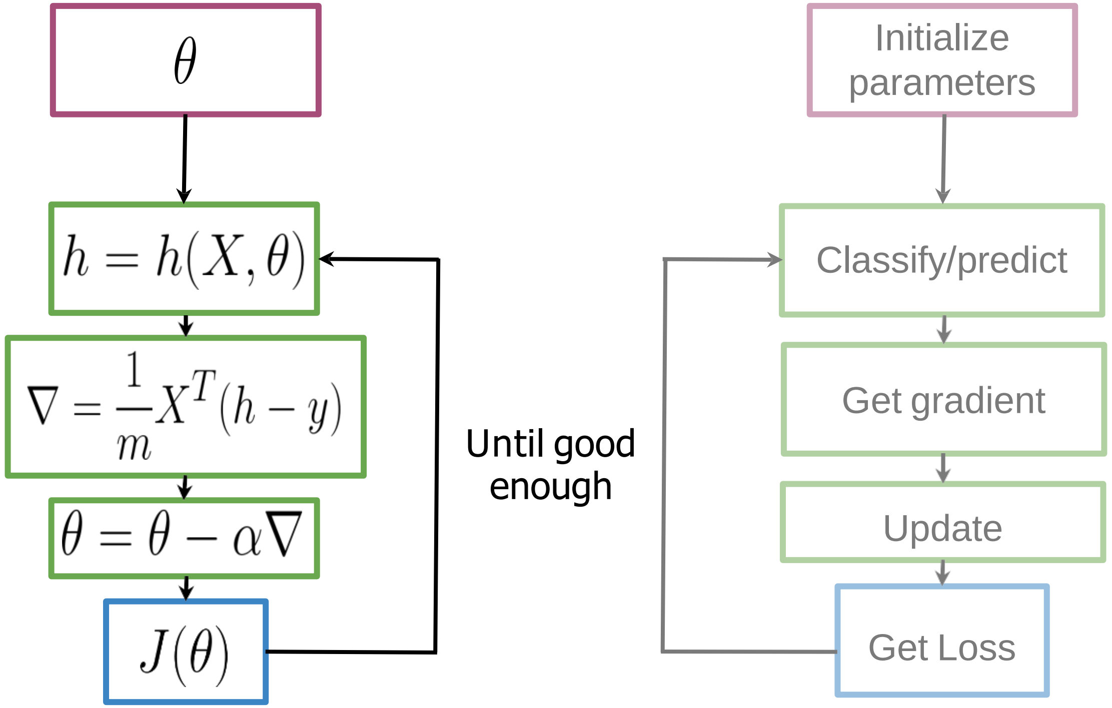
Logistic Regression: Training





- To train a logistic regression classifier, iterate until you find the set of parameters θ,that minimizes your cost function.
- This algorithm of training is called gradient descent

Training LR



- Visualize how gradient descent works
- Use gradient descent to train your logistic regression classifier

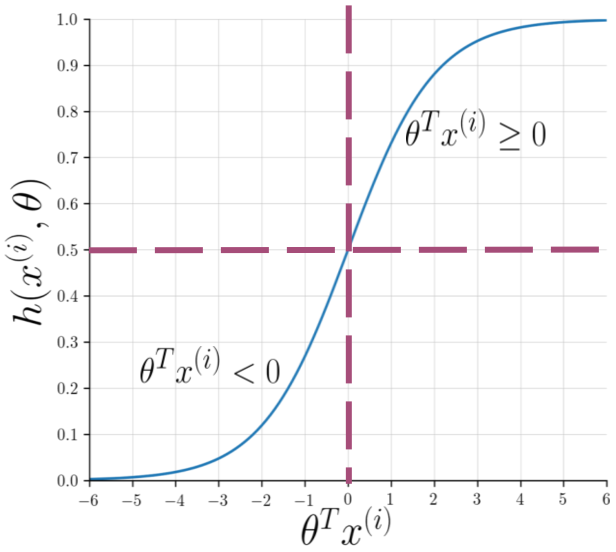
=>Compute the accuracy of your model

Testing logistic regression

- X_{val} Y_{val} θ

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

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



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

Testing logistic regression

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

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

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

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

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

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

Testing logistic regression

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

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

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

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

Cost function for logistic regression

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

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

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

Cost function for logistic regression

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

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

Cost function for logistic regression

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