UNIVERSITÀ DEGLI STUDI DI NAPOLI PARTHENOPE

Natural Language Processing

Text Classification: Sentiment Analysis

LESSON 7

prof. Antonino Staiano

M.Sc. In ''Machine Learning e Big Data'' - University Parthenope of Naples

Text classification

- Focus on text categorization or classification
  - The task of assigning a label or category to an entire text or document
- Many language processing tasks involve classification
  - Sentiment Analysis
  - Spam detection
  - Language id
  - Authorship's attribution
  - Assigning a library subject category to a text

Sentiment Analysis

- Extraction of sentiment
  - The positive or negative orientation that a writer expresses toward some object
    - A review of a movie, book, or product on the web
    - An editorial or political text, expressing a sentiment toward a candidate or political action
- Relevant for extracting consumer or public sentiment for fields from marketing to politics
- The words of reviews provide cues on sentiments

  \+ *…characters and richly applied satire, and some great plot twists*

  \- *It was pathetic. The worst part about it was the boxing scenes …*

  \+ *…awesome caramel sauce and sweet toasty almonds. I love this place!*

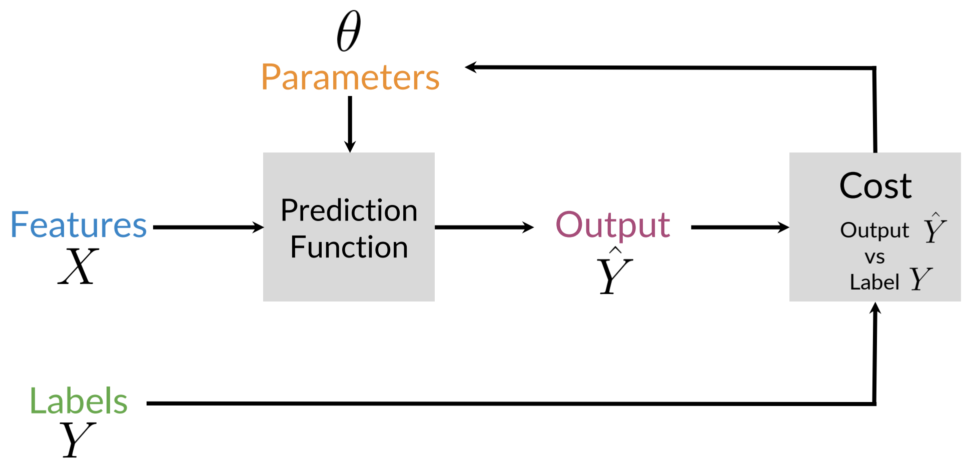
  \- *… awful pizza and ridiculously overpriced …*

Sentiment Analysis

- Let's suppose you have 1.000 product reviews, i.e., pieces of text written by users
- **Goal:** To build a system to automatically determine what fraction of them are positive vs negative reviews
- A classification problem where:
  - A sample text is labeled as a positive sentiment or negative sentiment

Sentiment Analysis as a classification task

- Supervised ML
  - given input features X and their labels Y
- Goal: minimize the cost as much as possible through a learning process
  - A prediction function takes in parameters data to map features to output labels
  - the best mapping from features to labels is achieved when the difference between the expected and predicted values is minimized



Discriminative vs Generative classifiers

- Given a training set of documents each hand-labeled with a class
  - (d_1,c_1), (d_2, c_2), …, (d_N, c_N)
- A **probabilistic** (generative) classifier maps a new document *d* to its correct class $c \in C$, providing us with the probability of the observation being in the class
  - Builds a model of how a class could generate some input data
    - Given an observation, it returns the class most likely to have generated the observation
- A discriminative classifier learns what features from the input are most useful to discriminate between the different possible classes
- We're going to use logistic regression (discriminative) and naïve Bayes (generative) for our task

Sentiment Analysis as a classification task

- Tweet: *I am happy because I am learning NLP*

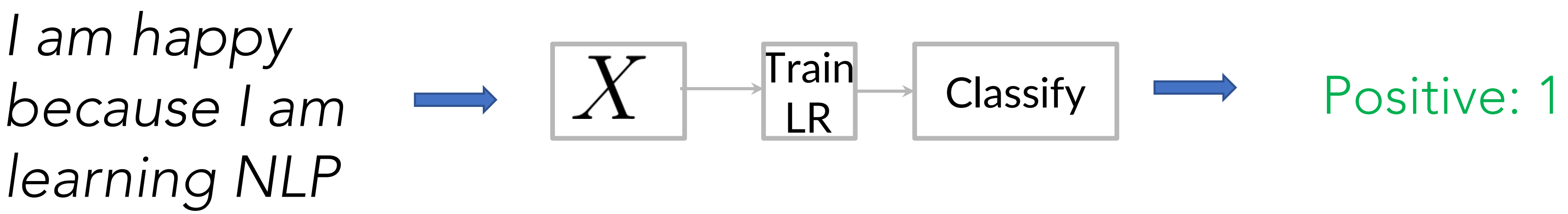
Positive: 1

Negative: 0

Logistic Regression

Steps for Sentiment Analysis with LR

- To perform sentiment analysis on a tweet
  1. Represent the text (i.e., "*I am happy because I am learning NLP*") as features
  2. Train an LR classifier
  3. use LR to classify the text



Sentiment Analysis

Vocabulary & text representation

Vocabulary

- In order to feed tweets into the LR classifier, we need to represent the text as a vector
- Firstly, we build a vocabulary V that makes it possible to encode any text or tweet as an array of numbers
- V would be the list of unique words from your list of tweet

- Example
  - Tweets: [tweet_1, tweet_2, …, tweet_m] →

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

- V = [I, am, happy, because, NLP, …, hated, the, movie]

Text representation

- The vocabulary is used to represent the tweets
  - For every tweet, scan every word from the vocabulary and put a 1 if it appears in the tweet, 0 otherwise (**BoW**)

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]

  - Note: this type of representation with a small number of non-zero values is called a sparse representation

Problems with sparse representation

- If n = |V|, with the sparse representation, a logistic regression model would have to learn **n+1** parameters
- For large vocabulary sizes:
  - Large training time
  - Large time for predictions

I am happy because I am learning NLP

$[1\ 1\ 1\ 1\ 1\ 1\ \ldots\ \underbrace{0\ \ldots\ 0\ 0\ 0}]$ ⇒ all zeros

$1 \longrightarrow |V|$

Sentiment Analysis

Negative and Positive frequencies

Positive and Negative frequency

- A better representation could be obtained by a frequency count of the vocabulary for each class of tweets
- Example
  - Let's suppose to have a corpus consisting of four tweets and associated with that corpus, a set of unique words, i.e., the vocabulary V
  - |V|=8
  - Two positive tweets and two negative tweets
- Count the times any word in V appears in the positive tweets and negative tweets

Corpus

I am happy because I am learning NLP

I am happy

I am sad, I am not learning NLP

I am sad

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
I
am
happy
because
learning
NLP
sad
not

Positive and Negative frequency

- Given a corpus with positive and negative tweets as follows:

Positive tweets	Negative tweets
I am happy because I am learning NLP	I am sad, I am not learning NLP
I am happy	I am sad

- Create a dictionary to map the word and the class it appeared in (positive or negative)

Word frequencies in classes

- Create a dictionary to map the word, and the class it appeared in (positive or negative)

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 with frequencies

- Encode a tweet as a 3D-vector

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

Features of tweet m

Sum Pos Frequencies

Sum Neg Frequencies

Bias

Feature extraction with frequencies

- Encode the positive feature:

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

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

$$\boxed{8}$$

Feature extraction with frequencies

- Encode the negative feature

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

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

$$\downarrow$$

$\boxed{11}$

- Tweet *I am sad, I am not learning* -> [1,8,11], where 1 is for the bias, 8 the positive feature, and 11 the negative feature

Feature extraction

- Tweet *I am sad, I am not learning* -> [1,8,11], where 1 is for the bias, 8 the positive feature, and 11 the negative feature

I am sad, I am not learning NLP

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

$$\downarrow$$

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

Sentiment Analysis

Preprocessing

Tweet preprocessing

- When working with any text corpus the first step to accomplish is a preprocessing step where the text is clean of unmeaningful information
- That’s true also for *tweet corpus*
- When preprocessing, the following steps are performed
  1. Eliminate handles and URLs
  2. Tokenize the string into words
  3. Remove stop words like “and, is, a, on, etc.”
  4. Stemming or converting every word to its stem, e.g., dancer, dancing, danced, becomes 'danc'
  5. Convert all your words to lowercase

Stop words and punctuation

- Let’s consider the following tweet

> @AntSta and @UniParthNLPStud are tuning a GREAT AI model at https://neptunia.uniparthenope.it!!!

Stop words
and
is
are
at
has
for
a

Punctuation
,
.
:
!
“
‘

Stop words and punctuaction

@AntSta ~~and~~ @UniParthNLPStud ~~are~~ tuning ~~a~~ GREAT AI model ~~at~~ https://neptunia.uniparthenope.it!!!

↓

@AntSta @UniParthNLPStud tuning GREAT AI model https://neptunia.uniparthenope.it!!!

Stop words
and
is
are
at
has
for
a

Punctuation
,
.
:
!
“
‘

Stop words and punctuation

@AntSta @UniParthNLPStud tuning GREAT AI model https://neptunia.uniparthenope.it~~!!!~~

↓

@AntSta @UniParthNLPStud tuning GREAT AI model https://neptunia.uniparthenope.it

Stop words
and
is
a
at
has
for
of

Punctuation
,
.
:
!
“
‘

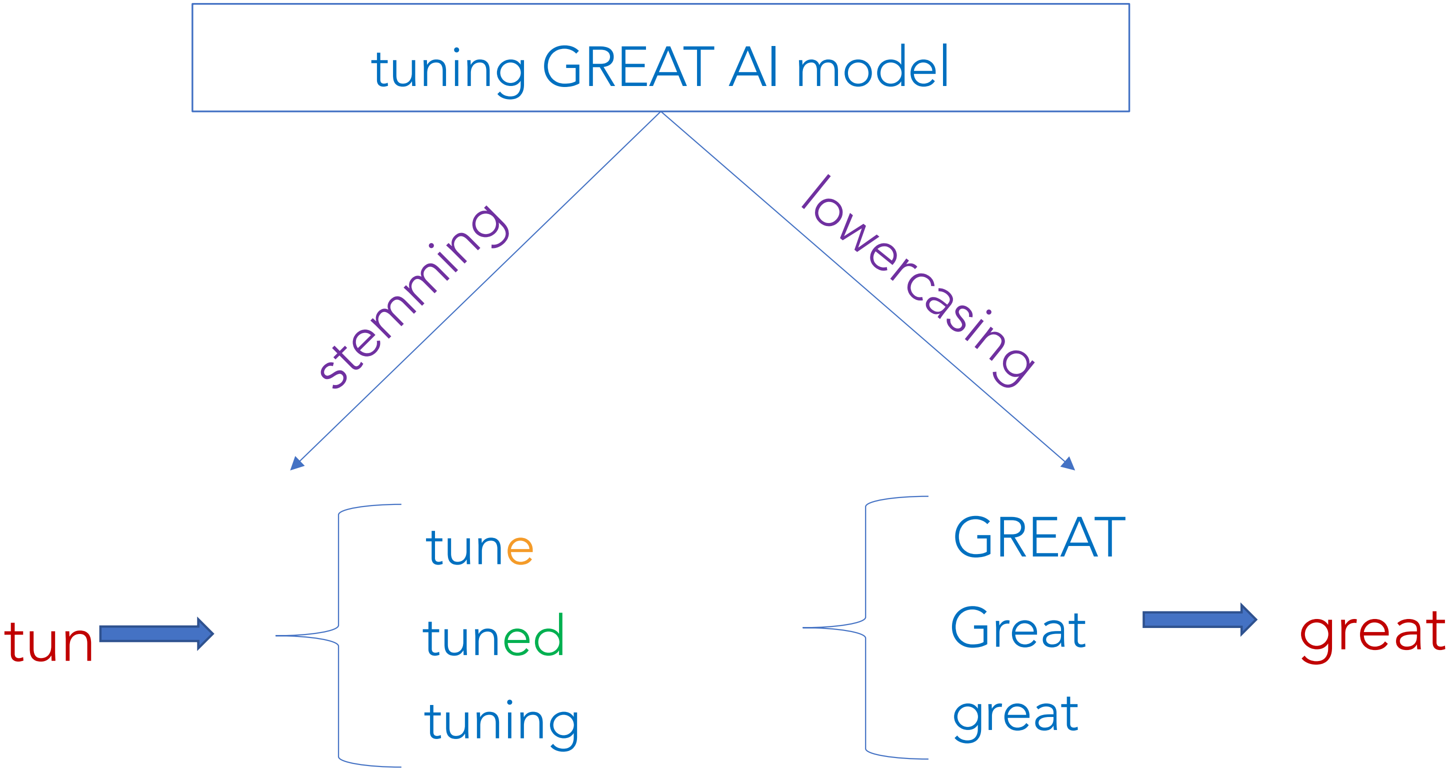
Handles and URLs

~~@AntSta @UniParthNLPStud~~ tuning GREAT AI model ~~https://neptunia.uniparthenope.it~~

↓

tuning GREAT AI model

Stemming and lowercasing



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

Sentimental Analysis

General overview

Until now

- Given a text
  - perform preprocessing followed by a feature extraction step to convert text into numerical representation

I am Happy Because I am learning NLP @UniparthNLP

↓

[happy, learn, nlp]

↓

Bias ← [1, 4, 2] → Sum Neg Freq

↓

Sum Pos Freq

General overview

- The previous process is performed on a set of m tweets

I am Happy Because I am learning NLP @UniparthNLP	[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]

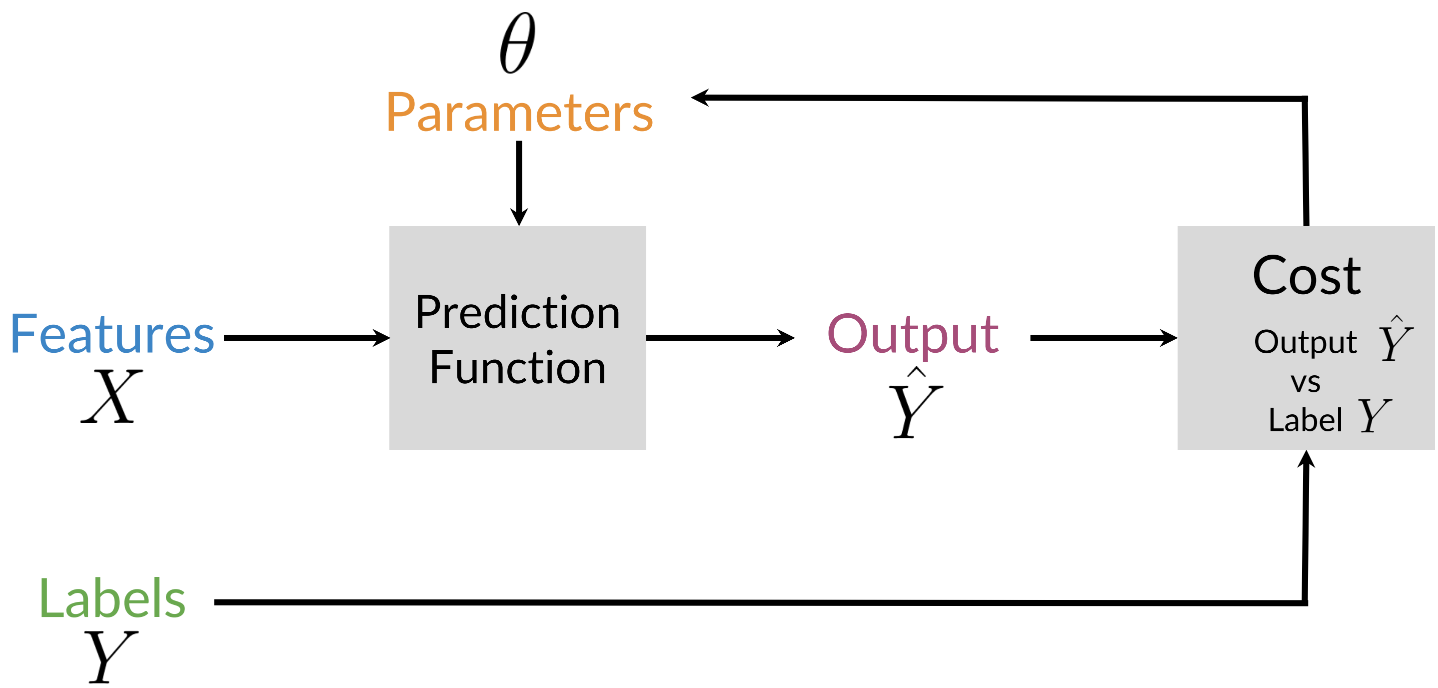
Numerical representation of tweets corpus

- This way we get a matrix of features for all m tweets, i.e., a mx3 matrix:

$$\boldsymbol{X} = \begin{bmatrix} \boxed{1 \quad X_1^{(1)} \quad X_2^{(1)}} \\ 1 \quad X_1^{(2)} \quad X_2^{(2)} \\ \vdots \quad \vdots \quad \vdots \\ 1 \quad X_1^{(m)} \quad X_2^{(m)} \end{bmatrix} \longleftrightarrow \begin{matrix} \boxed{[1,40,20]} \\ [1,20,50] \\ \dots \\ [1,5,35] \end{matrix}$$

- X will be the input of a LR model

Logistic regression



Logistic regression

- Input observation
  - vector $x = [x_1, x_2, \ldots, x_n]$
- Weights (one per feature)
  - $\theta = [\theta_1, \theta_2, \ldots, \theta_n]$
- Output
  - a predicted class $\hat{y} \in \{0,1\}$

How to do the classification

- For each feature x_i, weight θ_i tells us the importance of x_i
  - (Plus, we'll have a bias b)
- All the weighted features and the bias are summed up

$$h = \left(\sum_{i=0}^{n-1} \theta_i x_i\right) + b = \boldsymbol{\theta} \cdot \boldsymbol{x} + b$$

- If this sum is high, we say h=1; if low, then h=0

Making the LR's output a probability

- We need to formalize "*the sum is high*"
- We'd like a principled classifier that gives us a probability
- We want a model that can tell us:

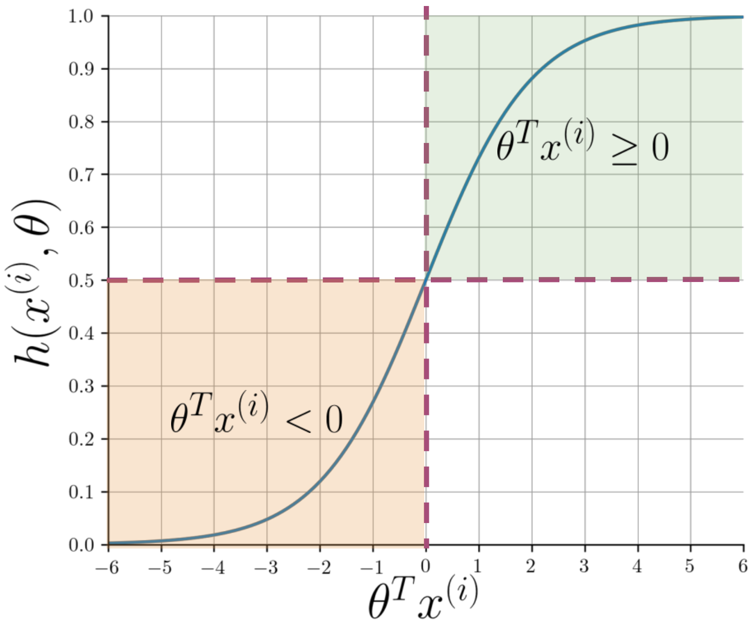
$$p(y=1|x;\theta)$$
$$p(y=0|x;\theta)$$

- Solution: use a function of h that goes from 0 to 1

Logistic regression

- Logistic regression makes use of the sigmoid function which outputs a probability between 0 and 1

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



Where do θ's come from?

- Supervised classification:
- We know the correct label y (either 0 or 1) for each x
- But what the system produces is an estimate, $\hat{y}$
- We want to set **θ** and b to minimize the **distance** between our estimate $\hat{y}^{(i)}$ and the true $y^{(i)}$
- We need a distance estimator
  - a loss function or a cost function
- We need an optimization algorithm to update **θ** and b to minimize the loss

Cross-Entropy Loss

- A case of conditional maximum likelihood estimation
- We choose the parameters θ, b that maximize
  - The log probability …
  - of the true y labels in the training data …
  - given the observation x

Cross-entropy loss

- **Goal**: maximize the probability of the correct label p(y/x)
- Since there are only 2 discrete outcomes (0 or 1) we can express the probability p(y/x) from our classifier as

$$p(y|x) = \hat{y}^{y}\,(1-\hat{y})^{1-y}$$

- Noting:
  - If y=1, this simplifies to $\hat{y}$
  - If y=0, this simplifies to 1- $\hat{y}$

Cross-entropy loss

- Now take the log of both sides (mathematically handy)

$$
\begin{aligned}
\log p(y|x) &= \log\left[\hat{y}^{y}\,(1-\hat{y})^{1-y}\right] \\
&= y\log\hat{y} + (1-y)\log(1-\hat{y})
\end{aligned}
$$

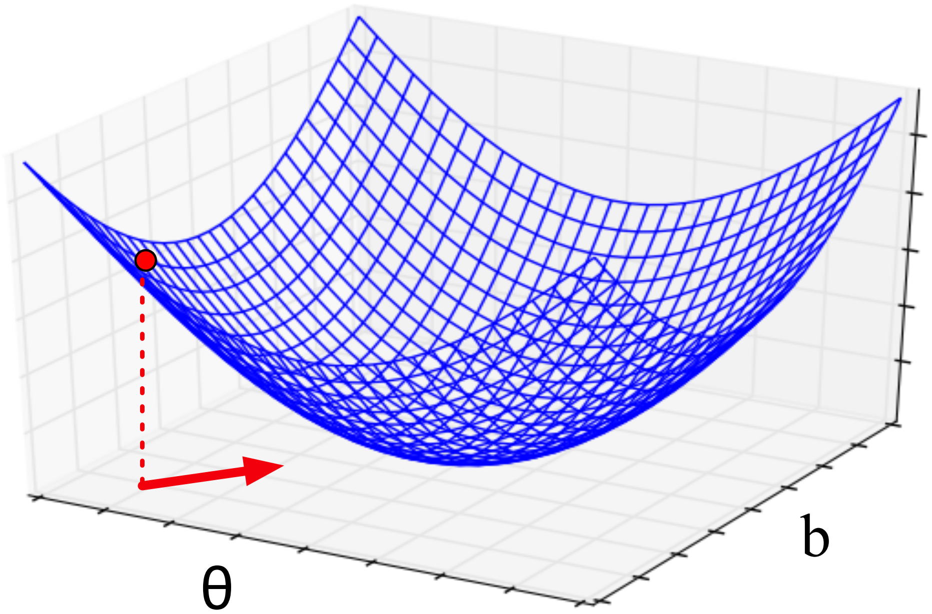
Stochastic Gradient Descent

- Let's make explicit that the loss function is parameterized by weights θ
  - we'll represent $\hat{y}$ as h (x; **θ**) to make the dependence on **θ** more obvious
- We want the weights that minimize the loss, averaged over all examples:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \frac{1}{m} \sum_{i=1}^{m} L_{\mathrm{CE}}(h(x^{(i)}; \theta), y^{(i)})$$

Minimizing the loss

- For logistic regression, the loss function is convex
- A convex function has just one minimum
- Gradient descent starting from any point is guaranteed to find the minimum
    - The **gradient** of a function of many variables is a vector pointing in the direction of the greatest increase in a function
- Gradient Descent: Find the gradient of the loss function at the current point and move in the **opposite** direction



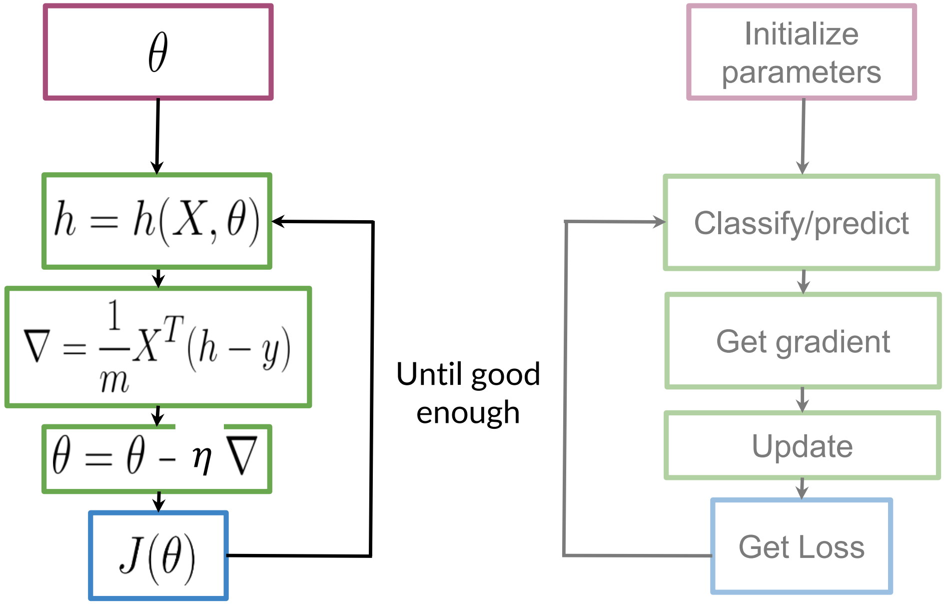
Gradient descent

- The value of the gradient $\frac{d}{d\theta}L(h(x;\theta),y)$ weighted by a **learning rate η**
- A higher learning rate means move θ faster

$$\theta^{t+1} = \theta^{t} - \eta \frac{d}{d\theta}L(h(x;\theta),y)$$

LR training

- To train LR function, the following procedure is performed
  - Initialize the parameter theta
  - compute the gradient to update theta
  - calculate the cost until good enough



Logistic regression

- Given a tweet, you can transform it into a vector and run it through your sigmoid function to get a prediction

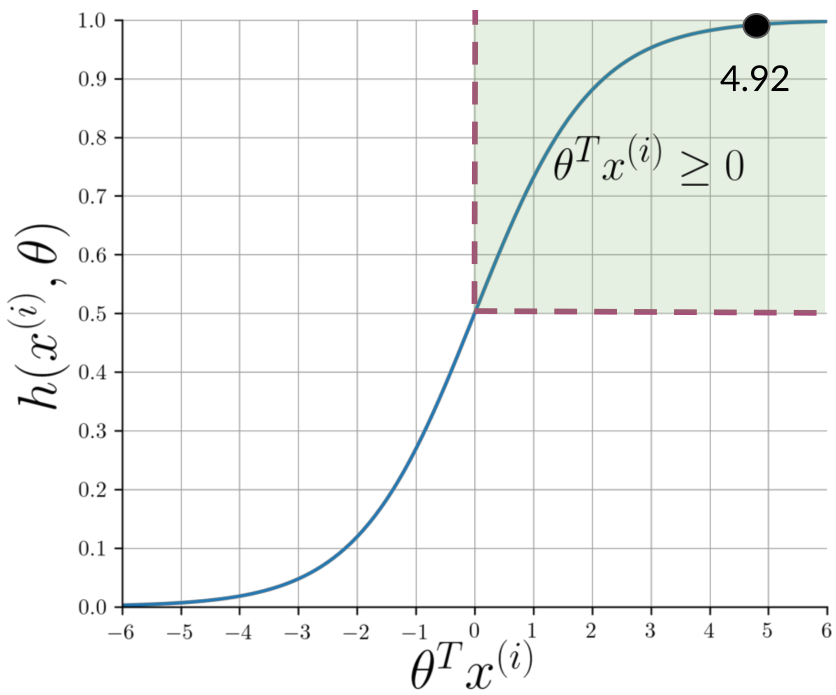
@AntSta and @UniParthNLPStud are tuning a GREAT AI model at https://neptunia.uniparthenope.it!!!

↓

[tun, ai, great, model]

↓

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



Testing the LR classifier

- Compute the LR prediction on each tweet from a test set and compare it to corresponding label

$$y_{val}^{(i)} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 1 \end{bmatrix} \qquad pred^{(i)} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

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