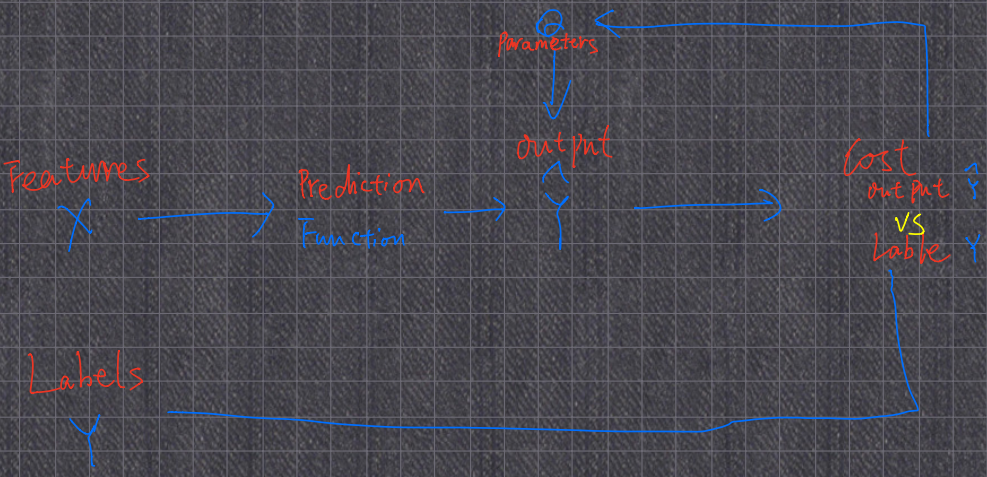
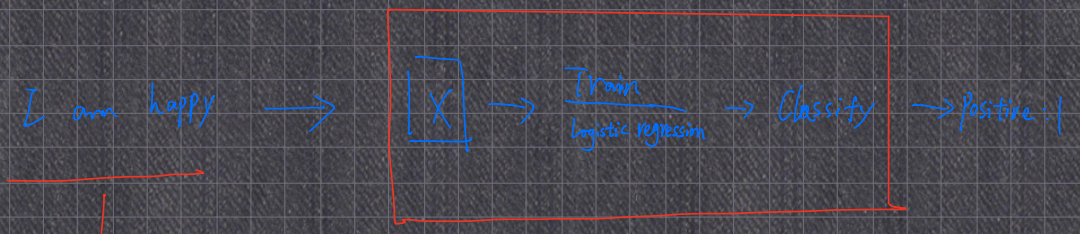


To perform sentiment analysis



Vocabulary & Feature Extraction

$V = [1, 1, 1, 0 \cdots 0]$ → [1. Large training time; 2. Large prediction time.]

对应出现的就为1，没有的就为0.

parameters: n+1 (n: 单词数)

Feature Extraction with Frequencies

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

$\sum_w freqs(w, 1)$：单词在 Pos 中的频率

$\sum_w freqs(w, 0)$：在 Neg 中的频率

Preprocessing

数据预处理

ϕ0. 去掉 the、a、an 等这类不重要的 word

将 -ing、-ed 这类的转变成原型.

NLTK

```
from nltk.corpus import stopwords          # 停词
from nltk.stem import PorterStemmer        # 词干提取
from nltk.tokenize import TweetTokenizer   # 将字符串拆分为每一个单词.
```

1. 删除字符串中不需要的字符、连接(eg: 链接、邮箱)

   可用 re.sub()

   ↑正则.

2. 将字符串拆分为每一个单词.

```
tokenizer = TweetTokenizer(preserve_case=False, strip_handles=True, reduce_len=True)
tweet_tokens = tokenizer.tokenize(tweet)
```

3. 删除停用词和标点符号.

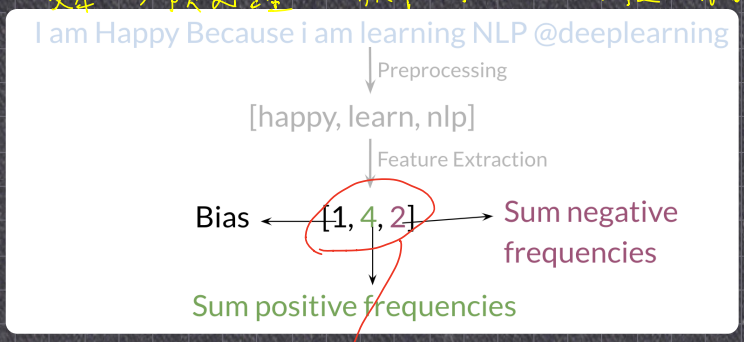
```
stop_words = stopwords.words('english')
if word not in stop_words:
    pass
```

4. 词干提取.

```
Stemmer = PorterStemming()
stem_word = stemmer.stem(word)
```

Putting it all together.

文本 → 预处理 → 提取特征 → 文本转化为数字.



将每个tweet的 $[1, x_1^{(i)}, x_2^{(i)}]$ 与组成 $X = \begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} \\ \vdots & \vdots & \vdots \\ 1 & x_1^{(m)} & x_2^{(m)} \end{bmatrix}$

```
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
```

Building and Visualizing word frequencies

将数据叠加，变成频率矩阵。

创建一个矩阵 表示 tweet的标签 (0, 1)

```
labels = np.append(np.ones(len(all_positive)), np.zeros(len(all_negative)))
```

```
def build_freqs(tweets, labels):
    label_list = np.squeeze(labels).tolist()
```

去掉列的维度

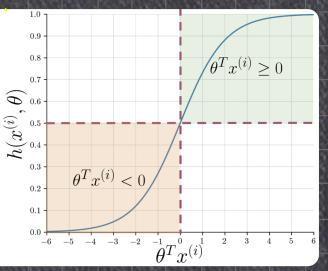
```
#转换化list()是为了迭代.
    freqs = { }
    for y, words in zip(label_list, tweets):
        for word in process_tweet(words):
            pair = (word, y)
            if pair in freqs:
                freqs[pair] += 1
            else:
                freqs[pair] = 1

    return freqs
```

Logistic Regression Overview

Sigmoid:

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



如何对tweet进行预测:

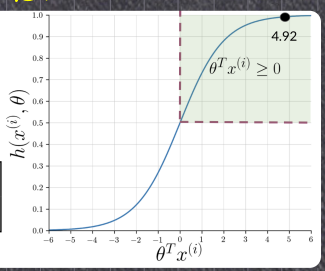
@YMourri and @AndrewYNg are tuning a GREAT AI model

↓

[tun, ai, great, model]

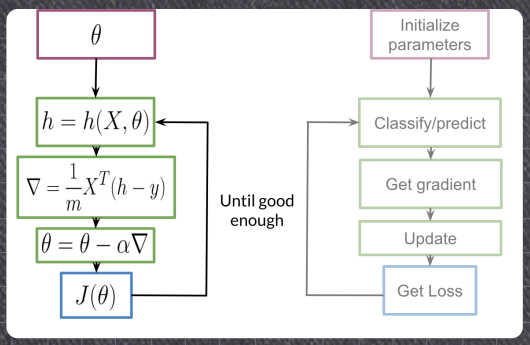
↓

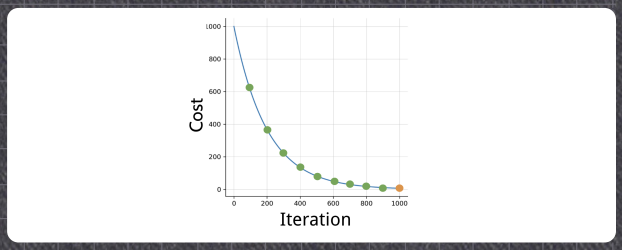
$$x^{(i)} = \begin{bmatrix} 1 \\ 3476 \\ 245 \end{bmatrix} \quad \theta = \begin{bmatrix} 0.00003 \\ 0.00150 \\ -0.00120 \end{bmatrix}$$



Training

初始化参数θ → 更新θ → 迭代降低 cost.





Testing

根据 Sigmoid 输出 pred ⇒ $\geq 0.5 \rightarrow$ positive, < 0.5 negative

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

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

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

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

与 y_{val} 对应的 y 真实结果比较，得到准确率

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

Cost Function

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

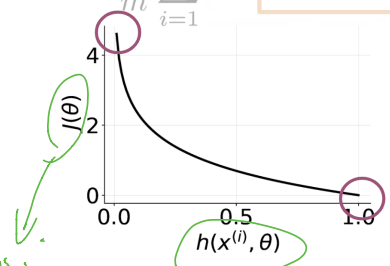
当 $y = 1$, $h(x^{(i)}, \theta) = 0$ 时, $J(\theta) = \infty$

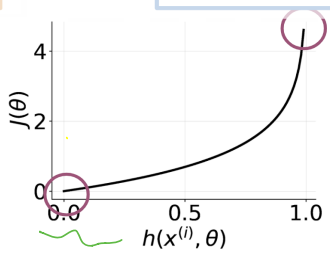
$y = 0$, $h(x^{(i)}, \theta) = 1$ 时, $J(\theta) = \infty$

当 $y = 1$, $h(x^{(i)}, \theta) = 1$ 时, $J(\theta) = 0$

$y = 0$, $h(x^{(i)}, \theta) = 0$ 时, $J(\theta) = 0$

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





cost减小

当y=1，预测为1时，

当y=0时，预测为0，cost最小。