Text Models

Previously on CSCI 4622

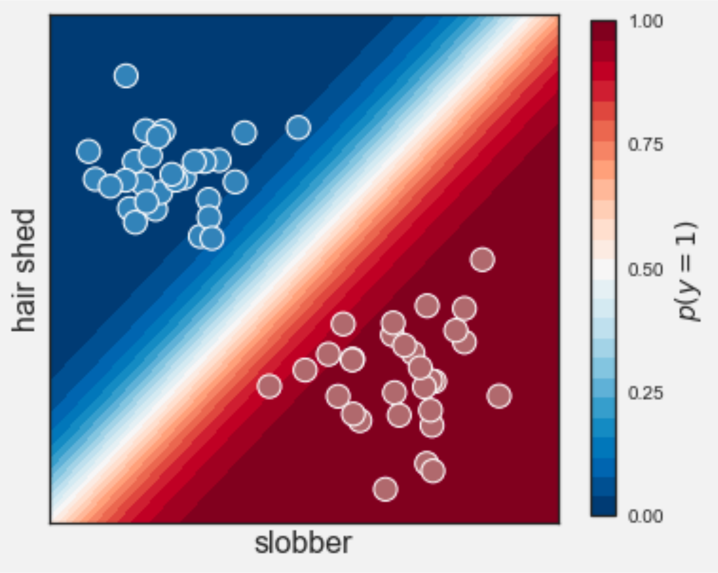
Logistic Regression:

- Learn parameters $\beta_0, \beta_1, \ldots, \beta_p$ to model probability that example is in class by

$$p(y = 1 \mid \mathbf{x}) = \text{sigm}\left(\beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p\right)$$

- Decision rule for two-feature binary classification:

$$\hat{y} = \begin{cases} 1 & \text{if } \text{sigm}(\hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2) \geq 0.5 \\ 0 & \text{if } \text{sigm}(\hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2) < 0.5 \end{cases}$$



Logistic Regression for Spam vs Ham

Example: How would you classify each of the following emails?

- **Email 1:** Mom, I got ~~a~~ job ~~in~~ Nigeria.

HAMMY

- **Email 2:** Jobs in Nigeria! Money, money, money, money!

SPAMMY

⇒ lowercase

⇒ REMOVE 's

Vector Space Models of Text

Before we can use Logistic Regression, have to define what the features are.

$$p(y = 1 \mid \mathbf{x}) = \operatorname{sigm}\left(\underline{\beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p}\right)$$

Most text models are what we call **vector space models**:

$$\text{“}\cancel{\text{the}}\text{ quick brown fox”} \rightarrow \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \begin{array}{l} \\ \leftarrow \text{QUICK} \\ \\ \leftarrow \text{BROWN} \\ \leftarrow \text{FOX} \\ \\ \end{array}$$

Vector Space Models of Text

Before we can use Logistic Regression, have to define what the features are.

$$p(y = 1 \mid \mathbf{x}) = \mathrm{sigm}\left(\beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p\right)$$

Most text models are what we call **vector space models**:

$$\text{“the quick brown fox”} \rightarrow \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \qquad \text{“}\cancel{\text{the}}\text{ fox jumps over }\cancel{\text{the}}\text{ log”} \rightarrow \begin{array}{c} \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \end{bmatrix} \end{array} \begin{array}{l} \leftarrow \text{jumps} \\ \\ \leftarrow \text{over} \\ \\ \leftarrow \text{fox} \\ \leftarrow \text{log} \end{array}$$

Vector Space Models of Text

- Suppose we have a training set comprised of many documents (articles, tweets, reviews)
- From the training set we extract a vocabulary V of distinct words
- Each document is represented by a feature vector $\mathbf{x}$ of length $p = |V|$
- Each feature x_k corresponds to a particular word in the vocabulary

Example: For the quick brown fox example, our vocabulary might be:

$$V = \{\ \underset{1}{\text{quick}},\ \underset{2}{\text{brown}},\ \underset{3}{\text{fox}},\ \underset{4}{\text{jump}},\ \underset{5}{\text{over}},\ \underset{6}{\text{log}}\}$$

$|V| = 6$

Vector Space Models of Text

Example: For the quick brown fox example, our vocabulary might be:

$$V = \{\text{ quick, brown, fox, jump, over, log}\}$$

- Represent a mapping from vocabulary to feature indices by a hash table

$$V = \{\text{quick} : 2,\ \text{brown} : 4,\ \text{fox} : 5,\ \text{jump} : 1,\ \text{over} : 3,\ \text{log} : 6\}$$

Binary Text Models encode a document as a binary feature vector, where feature x_k is 1 if term k appears anywhere in the document, and 0 otherwise.

Vector Space Models of Text

The mapping for our quick brown fox example might be:

$$V = \{\text{quick} : 2,\ \text{brown} : 4,\ \text{fox} : 5,\ \text{jump} : 1,\ \text{over} : 3,\ \log : 6\}$$

Binary Text Models encode a document as a binary feature vector, where feature x_k is 1 if term k appears anywhere in the document, and 0 otherwise.

Example: Encode the document "~~That~~ quick brown fox ~~is~~ quick!" using the binary model

$$x = \begin{array}{cc} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} & \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \end{array}$$

Logistic Regression for Text

Suppose we've encoded a **training set** of documents and labels as $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ where the classes are represented as the labels $y_i \in \{0, 1\}$

We want to learn a Logistic Regression model of the form

$$p(y = 1 \mid \mathbf{x}) = \text{sigm}\left(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p\right) = \text{sigm}(\boldsymbol{\beta}^T \mathbf{x})$$

Need to make vector lengths work out, so prepend each feature vector $\mathbf{x}$ with a 1

$$V = \{\text{bias} : 0,\ \text{quick} : 2,\ \text{brown} : 4,\ \text{fox} : 5,\ \text{jump} : 1,\ \text{over} : 3,\ \text{log} : 6\}$$

Example: The document "That quick brown fox is quick!" then becomes:

$$\begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \in \mathbb{R}^{|V|+1}$$

← BIAS

Logistic Regression for Spam vs Ham

$\text{sigm}(\beta^T x)$

$\frac{1}{1+e^{-a}}$

feature	bias	*viagra*	*mom*	*job*	*nigeria*	*money*
parameter	β_0	β_1	β_2	β_3	β_4	β_5
learned value	0.1	3.0	-2.0	-1.0	2.0	0.5

Example: How would you classify the following email using the binary text model?

- **Email 1:** Mom, ~~I~~ ~~got~~ ~~a~~ job ~~in~~ Nigeria.

$$X = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

$$\beta^T \quad [.1 \;\; 3 \;\; -2 \;\; -1 \;\; 2 \;\; .5] \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

$$= .1 - 2 - 1 + 2 = -.9$$

$$P(y = \text{SPAM} \mid x) = \text{sigm}(-.9)$$

$$0.29 < .5 \Rightarrow \hat{y} = 0$$

HAM

Logistic Regression for Spam vs Ham

	1	0	0	1	1	1
feature	*bias*	*viagra*	*mom*	*job*	*nigeria*	*money*
parameter	β_0	β_1	β_2	β_3	β_4	β_5
learned value	0.1	3.0	-2.0	-1.0	2.0	0.5

Example: How would you classify the following email using the binary text model?

- **Email 2**: Jobs ~~in~~ Nigeria! Money, money, money, money!

$$y = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$\beta^T y = .1 - 1.0 + 2.0 + 0.5 = 1.6$$

$$p(y = \text{SPAM} \mid x) = \text{sign}(1.6) = 0.83$$

$$> .5 \Rightarrow \hat{y} = 1 \Rightarrow \text{SPAM}$$

Logistic Regression for Spam vs Ham

feature	bias	$viagra$	mom	job	$nigeria$	$money$
parameter	β_0	β_1	β_2	β_3	β_4	β_5
learned value	0.1	3.0	-2.0	-1.0	2.0	0.5

Example: How would you classify the following email using the binary text model?

- **Email 2**: Jobs in Nigeria! Money, money, money, money!

Does something seem off about this?

The term "money" appeared 4 times in the document, but only gets a 1 in the feature vector

Vector Space Models of Text

Recall our mapping for the quick brown fox example:

$$V = \{\text{bias} : 0, \text{jump} : 1, \text{quick} : 2, \text{over} : 3, \text{brown} : 4, \text{fox} : 5, \text{log} : 6\}$$

The **Bag-of-Words** Model takes into account the frequency of a term in a document

$$x_k = \#\ \text{times term } k \text{ appears in document}$$

Example: Encode the document "~~That~~ quick brown fox ~~is~~ quick!" using **Bag-of-Words**

$$\bar{x} = \left[\begin{array}{c} 1 \\ \hline 0 \\ 2 \\ 0 \\ 1 \\ 1 \\ 0 \end{array}\right]$$

Logistic Regression for Spam vs Ham

Email: { } $\beta^T x = 0.1 \geq 0 \Rightarrow$ SPAM

feature	bias	*viagra*	*mom*	*job*	*nigeria*	*money*
parameter	β_0	β_1	β_2	β_3	β_4	β_5
learned value	0.1	3.0	-2.0	-1.0	2.0	0.5

Example: How would you classify the following email using the **Bag-of-Words** model?

- **Email 2**: Jobs ~~in~~ Nigeria! Money, money, money, money!

$$y = \begin{bmatrix} 1 \\ \hline 0 \\ 0 \\ 1 \\ 1 \\ 4 \end{bmatrix}$$

$$\beta^T x = .1 - 1.0 + 2.0 + 4 \times .5$$
$$= 3.1 \qquad \mathrm{sigm}(3.1) = 0.96$$

Practical Implementation

Typically we store our training set in a matrix, where each row corresponds to a training example and each column corresponds to a feature.

Define the **Document-Term Matrix** $\mathbf{X}_{dt}$ for a **Bag-of-Words** model as follows:

$$[\mathbf{X}_{dt}]_{i,k} = \#\ \text{times term } k \text{ appears in document } i$$

Example: Suppose you have the following documents and vocabulary map, find $\mathbf{X}_{dt}$

$$\begin{aligned}&\texttt{Training Set}:\\&\texttt{d1}:\texttt{new york new tribune}\\&\texttt{d2}:\texttt{new york times}\\&\texttt{d3}:\texttt{los angeles times}\end{aligned}$$

$$V = \{\text{new} : 3,\ \text{york} : 6,\ \text{tribune} : 5,\ \text{times} : 4,\ \text{los} : 2,\ \text{angeles} : 1\}$$

Practical Implementation

Example: Suppose you have the following documents and vocabulary map, find $\mathbf{X}_{dt}$

$$\begin{aligned}
&\texttt{Training Set:}\\
&\texttt{d1: new york new tribune}\\
&\texttt{d2: new york times}\\
&\texttt{d3: los angeles times}
\end{aligned}$$

$$V = \{\text{new} : 3,\ \text{york} : 6,\ \text{tribune} : 5,\ \text{times} : 4,\ \text{los} : 2,\ \text{angeles} : 1\}$$

$$X_{dt} = \begin{array}{c c} & \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \end{array} & \begin{bmatrix} 0 & 0 & 2 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \end{bmatrix} \end{array}$$

Practical Implementation

Example: Suppose you have the following documents and vocabulary map, find $\mathbf{X}_{dt}$

$$\begin{aligned}&\texttt{Training Set :}\\&\texttt{d1 : new york new tribune}\\&\texttt{d2 : new york times}\\&\texttt{d3 : los angeles times}\end{aligned}$$

$$V = \{\text{new} : 3,\ \text{york} : 6,\ \text{tribune} : 5,\ \text{times} : 4,\ \text{los} : 2,\ \text{angeles} : 1\}$$

Solution: We found

$$\mathbf{X}_{df} = \begin{bmatrix} 0 & 0 & 2 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}$$

SPARSE MATRIX

Question: Suppose our vocabulary is huge, what data structure should we use for $\mathbf{X}_{dt}$?

Practical Implementation

Question: What if you're have documents of widely varying lengths? Should these be treated differently?

Idea: Rescale feature vectors so that longer docs don't overpower shorter docs

- Scale rows of $\mathbf{X}_{dt}$ to have unit length

$$\mathbf{X}_{df} = \begin{bmatrix} 0 & 0 & 2 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \end{bmatrix} \Rightarrow \mathbf{X}_{ndf} = \begin{bmatrix} 0 & 0 & 0.82 & 0 & 0.41 & 0.41 \\ 0 & 0 & 0.58 & 0.58 & 0 & 0.58 \\ 0.58 & 0.58 & 0 & 0.58 & 0 & 0 \end{bmatrix}$$

Building Better Features

Question: What words do you think should be good for doing classification?

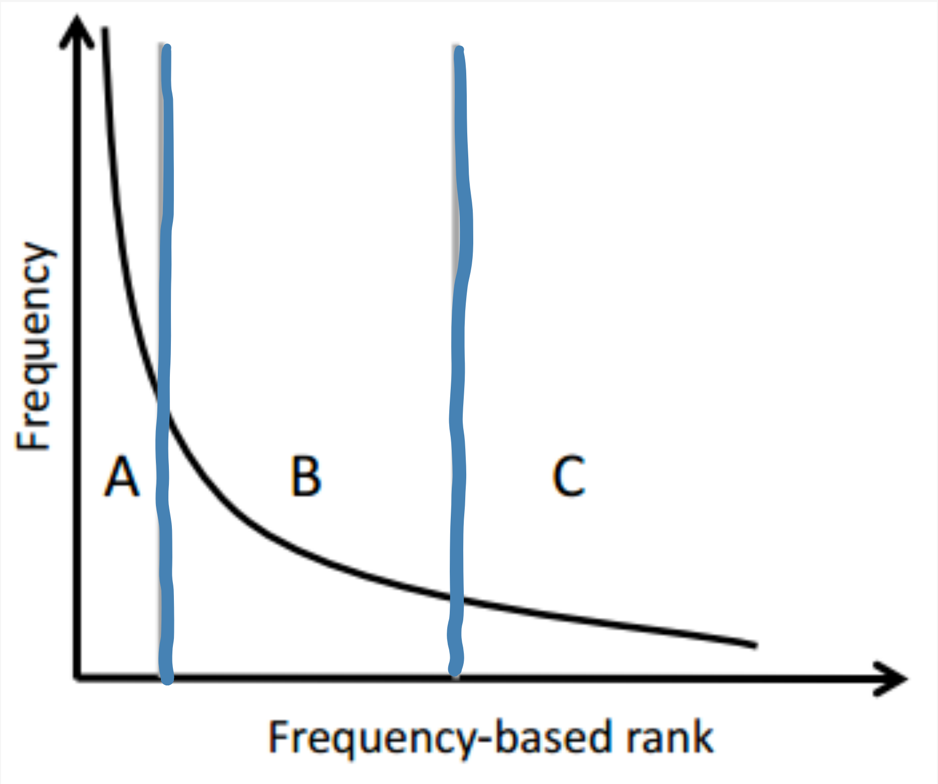
STOP WORDS = { IS, THE, THAT }

→ COMMON ⇒ LOW INFORMATION

Building Better Features

Question: What words do you think should be good for doing classification?

- Column A: Words that are too common **can't discriminate** between classes.
- Column C: Words that are too uncommon **can't generalize** to new data
- Column B: Good features are words that are common, but not too common.



Term Freq. – Inverse Document Freq.

Idea: Build term features based on how frequent a term is in a particular document and how many documents that term occurs in.

$$\mathtt{tfidf(d,t))} = \mathtt{tf(d,t)} \times \mathtt{idf(t)}$$

Term Freq. – Inverse Document Freq.

Idea: Build term features based on how frequent a term is in a particular document and how many documents that term occurs in.

$$\texttt{tfidf(d,t))} = \texttt{tf(d,t)} \times \texttt{idf(t)}$$

- The term frequency is the frequency of the term in the particular document

$$\texttt{tf(d,t)} = \texttt{\# times term t appears in document d}$$

Term Freq. - Inverse Document Freq.

Idea: Build term features based on how frequent a term is in a particular document and how many documents that term occurs in.

$$\texttt{tfidf(d,t))} = \texttt{tf(d,t)} \times \texttt{idf(t)}$$

- The term frequency is the frequency of the term in the particular document

$$\texttt{tf(d,t)} = \texttt{\# times term t appears in document d}$$

- The inverse document frequency is measure of how many documents the term appears in

$$\texttt{idf(t)} = \log\left(\frac{1+n_d}{1+\texttt{df(} \quad \texttt{t)}}\right)+1$$

Term Freq. – Inverse Document Freq.

The inverse document frequency is measure of how many documents the term appears in

- n_d is the total number of training documents
- `df(d,t)` is the number of documents that contain at least one instance of term t

$$\texttt{idf(t)} = \log\left(\frac{1 + n_d}{1 + \texttt{df}(\cancel{\texttt{d},}\,\texttt{t})}\right) + 1$$

Term Freq. - Inverse Document Freq.

Idea: Build term features based on how frequent a term is in a particular document and how many documents that term occurs in.

$$\mathtt{tfidf(d,t))} = \mathtt{tf(d,t)} \times \mathtt{idf(t)}$$

Term Freq. - Inverse Document Freq.

Example: Compute the tfidf score for the word "new" in the following document set

```
Training Set :
d1 : new york new tribune
d2 : new york times
d3 : los angeles times
```

WE'LL WORK THIS EXAMPLE OUT IN THE LOGISTIC REGRESSION HANDS ON NOTEBOOK.

Term Freq. – Inverse Document Freq.

Example: Compute the tfidf score for the word "new" in the following document set

```
Training Set :
d1 : new york new tribune
d2 : new york times
d3 : los angeles times
```

Text Models Wrap-Up

- Vector Space Models allow us to represent documents and their terms as feature vectors
- Vector Space Models come in many flavors, from simple binary to TF-IDF.
- Much fancier text models exist, like Google's **Word2Vec** Model

Next Time:

- Learn to do text encoding in Scikit-Learn
- Explore text models and logistic regression for predicting sentiment in movie reviews