CS 124/LINGUIST 180
From Languages to Information

Dan Jurafsky
Stanford University

Logistic Regression
Logistic Regression

- Important analytic tool in natural and social sciences
- Baseline supervised machine learning tool for classification
- Is also the foundation of a neural network
Logistic Regression: two formats

- Binary logistic regression
  - Two output classes
- Multinomial logistic regression
  - More than 2 output classes
Generative and Discriminative Classifiers

Naïve Bayes is a **generative** classifier

by contrast:

Logistic regression is a **discriminative** classifier
Generative and Discriminative Classifiers

Suppose we're distinguishing cat from dog images
Generative Classifier:

• Build a model of what's in a cat image
  • Knows about whiskers, ears, eyes
  • Assigns a probability to any image:
    • how cat-y is this image?

Also build a model for dog images

Now given a new image:

Run both models and see which one fits better
Discriminative Classifier

Just try to distinguish dogs from cats

Oh look, dogs have collars!
Let's ignore everything else
Finding the correct class \( c \) from a document \( d \) in Generative vs Discriminative Classifiers

Naive Bayes

\[
\hat{c} = \arg\max_{c \in C} \left( \text{likelihood} \ P(d|c) \right) \left( \text{prior} \ P(c) \right)
\]

Logistic Regression

\[
\hat{c} = \arg\max_{c \in C} \left( \text{posterior} \ P(c|d) \right)
\]
Components of a probabilistic (supervised) machine learning classifier

A **corpus** of \( M \) observation input/output pairs, \((x^{(i)}, y^{(i)})\)

For each input observation \( x^{(i)} \)
- a vector of **features** \([x_1, x_2, ..., x_n]\)

A **classification function** computing \( \hat{y} \), via \( p(y|x) \)
- **sigmoid**
- **softmax**

For learning
- A **loss function** (cross-entropy loss)
- An **optimization algorithm** (stochastic gradient descent)
CS 124/LINGUIST 180
From Languages to Information
Dan Jurafsky
Stanford University
Classification in Logistic Regression
Logistic regression: two pieces

**Training**: given a training set of $M$ observations $(x,y)$
- Learn the parameters of the model

**Test**: Given a test example $x$ and class $y \in \{0,1\}$
- return the higher probability class
Classification in the test phase

Consider input observation \( x \)

- vector of features \([x_1, x_2, \ldots, x_n]\)
- Maybe they are counts of words

Goal: what is class \( y \)?

- We want to know the probability \( P(y = 1|x) \) and \( P(y=0|x) \)
- \( y=1 \) “positive sentiment”
- \( y=0 \) “negative sentiment”
Features in logistic regression

For each feature $x_i$, we'll have a weight $w_i$

Weight $w_i$ tells us how important feature $x_i$ is the classification

$x_i = "awesome": w_i$ very positive $+5$

$x_i = "abysmal": w_i$ very negative $-5$
How to do classification

For each feature $x_i$, we'll have a weight $w_i$

Plus we'll have a bias $b$

We'll sum up all the weighted features and the bias

$$z = \left( \sum_{i=1}^{n} w_i x_i \right) + b$$

$$z = w \cdot x + b$$
But $z$ isn't a probability, it's just a number!

$$z = w \cdot x + b$$

Solution: use a function of $z$ that goes from 0 to 1

$$y = \sigma(z) = \frac{1}{1 + e^{-z}}$$
The very useful sigmoid function

\[ y = \sigma(z) = \frac{1}{1 + e^{-z}} \]
Making probabilities with sigmoids

\[
P(y = 1) = \sigma(w \cdot x + b)
\]

\[
= \frac{1}{1 + e^{-(w \cdot x + b)}}
\]

\[
P(y = 0) = 1 - \sigma(w \cdot x + b)
\]

\[
= 1 - \frac{1}{1 + e^{-(w \cdot x + b)}}
\]

\[
= \frac{e^{-(w \cdot x + b)}}{1 + e^{-(w \cdot x + b)}}
\]
The sigmoid has a number of advantages; it takes a real-valued number and maps it into the range $[0,1]$, which is just what we want for a probability. Because it is nearly linear around 0 but has a sharp slope toward the ends, it tends to squash outlier values toward 0 or 1. And it's differentiable, which as we'll see in Section 5.8 will be handy for learning.

We're almost there. If we apply the sigmoid to the sum of the weighted features, we get a number between 0 and 1. To make it a probability, we just need to make sure that the two cases, $P(y=1)$ and $P(y=0)$, sum to 1. We can do this as follows:

$$P(y=1) = \frac{1}{1 + e^{-(w \cdot x + b)}}$$

$$P(y=0) = \frac{1}{1 + e^{-(w \cdot x + b)}}$$

Now we have an algorithm that given an instance $x$ computes the probability $P(y=1| x)$. How do we make a decision? For a test instance $x$, we say yes if the probability $P(y=1| x)$ is more than .5, and no otherwise. We call .5 the decision boundary:

$$\hat{y} = \begin{cases} 
1 & \text{if } P(y=1| x) > 0.5 \\
0 & \text{otherwise}
\end{cases}$$

5.1.1 Example: sentiment classification

Let's have an example. Suppose we are doing binary sentiment classification on movie review text, and we would like to know whether to assign the sentiment class + or - to a review document $doc$. We'll represent each input observation by the following 6 features $x_1 \ldots x_6$ of the input; Fig. 5.2 shows the features in a sample mini test document.

<table>
<thead>
<tr>
<th>Var Definition</th>
<th>Value in Fig. 5.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>count(positive lexicon)</td>
<td>2</td>
</tr>
<tr>
<td>count(negative lexicon)</td>
<td>2</td>
</tr>
<tr>
<td>count(1st and 2nd pronouns)</td>
<td>3</td>
</tr>
<tr>
<td>$\implies$ 1 if &quot;no&quot;</td>
<td>0</td>
</tr>
<tr>
<td>$\implies$ 1 if &quot;!&quot;</td>
<td>1</td>
</tr>
<tr>
<td>log(word count of $doc$)</td>
<td>$\ln(64) = 4.15$</td>
</tr>
</tbody>
</table>

Let's assume for the moment that we've already learned a real-valued weight for each of these features, and that the 6 weights corresponding to the 6 features are $[2.5, 5.0, 1.2, 0.5, 2.0, 0.7]$, while $b = 0.1$. (We'll discuss in the next section how the weights are learned.) The weight $w_1$, for example indicates how important

$$\hat{y} = \begin{cases} 
1 & \text{if } P(y=1| x) > 0.5 \\
0 & \text{otherwise}
\end{cases}$$


Sentiment example: does $y=1$ or $y=0$?

It's hokey, there are virtually no surprises, and the writing is second-rate. So why was it so enjoyable? For one thing, the cast is great. Another nice touch is the music. I was overcome with the urge to get off the couch and start dancing. It sucked me in, and it'll do the same to you.
It's hokey, there are virtually no surprises, and the writing is second-rate. So why was it so enjoyable? For one thing, the cast is great. Another nice touch is the music. I was overcome with the urge to get off the couch and start dancing. It sucked me in, and it'll do the same to you.

Positive sentiment words
Negative sentiment words
"no"
i/me/mine/you/your/yours
Let's have an example. Suppose we are doing binary sentiment classification on a movie review text, and we would like to know whether to assign the sentiment class positive or negative.

5.1.1 Example: sentiment classification

We're almost there. If we apply the sigmoid to the sum of the weighted features, we will get a probability between 0 and 1 of belonging to the positive class.

\[ P(y=1) = \frac{1}{1 + e^{-\sum w_i x_i}} \]

The sigmoid has a number of advantages; it takes a real-valued number and maps it to a probability. It is nearly linear around 0 but has a sharp slope toward the ends, it tends to squash outlier probabilities, and it will provide insights into features.

Given these 6 features and the input review text, we can compute the probability of a positive sentiment decision, while the class is EOS.

![Graph showing features and their values](image)

<table>
<thead>
<tr>
<th>Var</th>
<th>Definition</th>
<th>Value in Fig. 5.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x_1)</td>
<td>count(positive lexicon) (\in) doc)</td>
<td>3</td>
</tr>
<tr>
<td>(x_2)</td>
<td>count(negative lexicon) (\in) doc)</td>
<td>2</td>
</tr>
<tr>
<td>(x_3)</td>
<td>(\begin{cases} 1 &amp; \text{if &quot;no&quot; } \in \text{doc} \ 0 &amp; \text{otherwise} \end{cases})</td>
<td>1</td>
</tr>
<tr>
<td>(x_4)</td>
<td>count(1st and 2nd pronouns (\in) doc)</td>
<td>3</td>
</tr>
<tr>
<td>(x_5)</td>
<td>(\begin{cases} 1 &amp; \text{if &quot;!&quot; } \in \text{doc} \ 0 &amp; \text{otherwise} \end{cases})</td>
<td>0</td>
</tr>
<tr>
<td>(x_6)</td>
<td>(\log(\text{word count of doc}))</td>
<td>(\ln(64) = 4.15)</td>
</tr>
</tbody>
</table>
Classifying sentiment for input $x$

<table>
<thead>
<tr>
<th>Var</th>
<th>Definition</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>count(positive lexicon) $\in$ doc</td>
<td>3</td>
</tr>
<tr>
<td>$x_2$</td>
<td>count(negative lexicon) $\in$ doc</td>
<td>2</td>
</tr>
<tr>
<td>$x_3$</td>
<td>$\begin{cases} 1 &amp; \text{if &quot;no&quot; } \in \text{doc} \ 0 &amp; \text{otherwise} \end{cases}$</td>
<td>1</td>
</tr>
<tr>
<td>$x_4$</td>
<td>count(1st and 2nd pronouns $\in$ doc)</td>
<td>3</td>
</tr>
<tr>
<td>$x_5$</td>
<td>$\begin{cases} 1 &amp; \text{if &quot;!&quot; } \in \text{doc} \ 0 &amp; \text{otherwise} \end{cases}$</td>
<td>0</td>
</tr>
<tr>
<td>$x_6$</td>
<td>log(word count of doc)</td>
<td>$\ln(64) = 4.15$</td>
</tr>
</tbody>
</table>

Suppose $w = [2.5, -5.0, -1.2, 0.5, 2.0, 0.7]$ 

$b = 0.1$
\[
p(+|x) = P(Y = 1|x) = \sigma(w \cdot x + b) \\
= \sigma([2.5, -5.0, -1.2, 0.5, 2.0, 0.7] \cdot [3, 2, 1, 3, 0, 4.15] + 0.1) \\
= \sigma(1.805) \\
= 0.86
\]

\[
p(-|x) = P(Y = 0|x) = 1 - \sigma(w \cdot x + b) \\
= 0.14
\]
We can build features for logistic regression for any classification task: period disambiguation

This ends in a period.
The house at 465 Main St. is new.

\[ x_1 = \begin{cases} 1 & \text{if “Case}(w_i) = \text{Lower”} \\ 0 & \text{otherwise} \end{cases} \]
\[ x_2 = \begin{cases} 1 & \text{if “} w_i \in \text{AcronymDict”} \\ 0 & \text{otherwise} \end{cases} \]
\[ x_3 = \begin{cases} 1 & \text{if “} w_i = \text{St.} & \text{Case}(w_{i-1}) = \text{Cap”} \\ 0 & \text{otherwise} \end{cases} \]
CS 124/LINGUIST 180
From Languages to Information
Dan Jurafsky
Stanford University
Multinomial Logistic Regression
Multinomial Logistic Regression

Often we need more than 2 classes
- Positive/negative/neutral
- Parts of speech (noun, verb, adjective, adverb, preposition, etc.)
- Classify emergency SMSs into different actionable classes

If >2 classes we use **multinomial logistic regression**
- "logistic regression" will just mean binary (2 output classes)
  - = Softmax regression
  - = Maximum entropy modeling
  - = Maxent
  - = Multinomial logit
Multinomial Logistic Regression

The probably of everything must still sum to 1

\[ P(\text{positive}|\text{doc}) + P(\text{negative}|\text{doc}) + P(\text{neutral}|\text{doc}) = 1 \]

Need a generalization of the sigmoid called the softmax

- Takes a vector \( z = [z_1, z_2, ..., z_k] \) of \( k \) arbitrary values
- Outputs a probability distribution
  - each value in the range \([0,1]\)
  - all the values summing to 1
The **softmax** function

Turns a vector \( z = [z_1, z_2, \ldots, z_k] \) of \( k \) arbitrary values into probabilities

\[
\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}} \quad 1 \leq i \leq k
\]

The denominator \( \sum_{i=1}^{k} e^{z_i} \) is used to normalize all the values into probabilities.

\[
\text{softmax}(z) = \left[ \frac{e^{z_1}}{\sum_{i=1}^{k} e^{z_i}}, \frac{e^{z_2}}{\sum_{i=1}^{k} e^{z_i}}, \ldots, \frac{e^{z_k}}{\sum_{i=1}^{k} e^{z_i}} \right]
\]
The **softmax** classifier

- Turns a vector $z = [z_1, z_2, \ldots, z_k]$ of $k$ arbitrary values into probabilities

  \[
  \text{softmax}(z) = \left[ \frac{e^{z_1}}{\sum_{i=1}^{k} e^{z_i}}, \frac{e^{z_2}}{\sum_{i=1}^{k} e^{z_i}}, \ldots, \frac{e^{z_k}}{\sum_{i=1}^{k} e^{z_i}} \right]
  \]

  $z = [0.6, 1.1, -1.5, 1.2, 3.2, -1.1]$

  $[0.055, 0.090, 0.0067, 0.10, 0.74, 0.010]$
Softmax in multinomial logistic regression

For a vector \( z \) of dimensionality \( k \), the softmax is defined as:

\[
\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{k=1}^{k} e^{z_j}}
\]  

The softmax of an input vector \( z = [z_1, z_2, \ldots, z_k] \) is thus a vector itself:

\[
\text{softmax}(z) = [e^{z_1}/\sum_{k=1}^{k} e^{z_j}, e^{z_2}/\sum_{k=1}^{k} e^{z_j}, \ldots, e^{z_k}/\sum_{k=1}^{k} e^{z_j}]
\]

The denominator \( \sum_{k} e^{z_j} \) is used to normalize all the values into probabilities.

Thus for example given a vector:

\( z = [0.6, 1.1, 1.5, 1.2, 3.2, 1.1] \)

the result softmax(\( z \)) is

\( [0.055, 0.090, 0.0067, 0.10, 0.74, 0.010] \)

Again like the sigmoid, the input to the softmax will be the dot product between a weight vector \( w \) and an input vector \( x \) (plus a bias). But now we'll need separate weight vectors (and bias) for each of the \( K \) classes.

\[
p(y = c|x) = \frac{e^{w_c \cdot x + b_c}}{\sum_{j=1}^{k} e^{w_j \cdot x + b_j}}
\]

Like the sigmoid, the softmax has the property of squashing values toward 0 or 1. Thus if one of the inputs is larger than the others, it will tend to push its probability toward 1, and suppress the probabilities of the smaller inputs.

5.6.1 Features in Multinomial Logistic Regression

For multiclass classification the input features need to be a function of both the observation \( x \) and the candidate output class \( c \). Thus instead of the notation \( x_i, f_i \) or \( f_i(x) \), when we're discussing features we will use the notation \( f_i(c, x) \), meaning feature \( i \) for a particular class \( c \) for a given observation \( x \).

In binary classification, a positive weight on a feature pointed toward \( y=1 \) and a negative weight toward \( y=0 \)... but in multiclass a feature could be evidence for or against an individual class.

Let's look at some sample features for a few NLP tasks to help understand this perhaps unintuitive use of features that are functions of both the observation \( x \) and the class \( c \),

Suppose we are doing text classification, and instead of binary classification our task is to assign one of the 3 classes +, −, or 0 (neutral) to a document. Now a feature related to exclamation marks might have a negative weight for 0 documents, and a positive weight for + or − documents.
Features in (binary) logistic regression

\[ f_1 = \begin{cases} 1 & \text{if "!" in doc} \\ 0 & \text{otherwise} \end{cases} \]

\[ w_1 = 2.5 \]
Features in softmax regression
distinct weights for each value!

<table>
<thead>
<tr>
<th>Var</th>
<th>Definition</th>
<th>Wt</th>
</tr>
</thead>
</table>
| $f_1(0,x)$ | \[
|         | \begin{cases}
|         | 1 \text{ if } "!" \in \text{doc} \\
|         | 0 \text{ otherwise} \end{cases} \] | -4.5 |
| $f_1(+,x)$ | \[
|         | \begin{cases}
|         | 1 \text{ if } "!" \in \text{doc} \\
|         | 0 \text{ otherwise} \end{cases} \] | 2.6  |
| $f_1(-,x)$ | \[
|         | \begin{cases}
|         | 1 \text{ if } "!" \in \text{doc} \\
|         | 0 \text{ otherwise} \end{cases} \] | 1.3  |
Binary versus multinomial logistic regression

Multinomial is obviously applicable to many more classification tasks
  • Softmax is important for neural net classifiers

Binary logistic regression is simpler to model

We'll stick to binary logistic regression in PA3
Multinomial Logistic Regression
Summary: Classification in Logistic Regression
Classification in (binary) logistic regression: summary

Given:

- a set of classes: (+ sentiment, - sentiment)
- a vector $x$ of features $[x_1, x_2, \ldots, x_n]$
  - $x_1 =$ count( "awesome"")
  - $x_2 =$ log(number of words in review)
- A vector $w$ of weights $[w_1, w_2, \ldots, w_n]$
  - $w_i$ for each feature $f_i$

$$P(y = 1) = \sigma(w \cdot x + b) = \frac{1}{1 + e^{-(w \cdot x + b)}}$$
Overfitting

+ This movie drew me in, and it'll do the same to you.

- I can't tell you how much I hated this movie. It sucked.

Useful or harmless features

X1 = "this"
X2 = "movie"
X3 = "hated"
X4 = "drew me in"

4gram features that just "memorize" training set and might cause problems

X5 = "the same to you"
X7 = "tell you how much"
Overfitting

4-gram or 5-gram model on tiny data will just memorize the data
  ◦ 100% accuracy on the training set
It will be surprised by all the novel 4-grams in the test data
  ◦ Low accuracy on test set

Models that are too powerful can **overfit** the data
  ◦ Fitting the details of the training data so exactly that the model doesn't generalize well to the test set
  ◦ There are various ways to avoid overfitting
    ◦ Regularization in logistic regression (see chapter 5)
    ◦ Dropout in neural networks
CS 124/LINGUIST 180
From Languages to Information

Dan Jurafsky
Stanford University

Summary: Classification in Logistic Regression
Lexicons as Features for Logistic Regression

Reminder: Sentiment Lexicons
The General Inquirer


- Home page: http://www.wjh.harvard.edu/~inquirer
- List of Categories: http://www.wjh.harvard.edu/~inquirer/homecat.htm
- Spreadsheet: http://www.wjh.harvard.edu/~inquirer/inquirerbasic.xls
- Categories:
  - Positiv (1915 words) and Negativ (2291 words)
  - Strong vs Weak, Active vs Passive, Overstated versus Understated
  - Pleasure, Pain, Virtue, Vice, Motivation, Cognitive Orientation, etc
- Free for Research Use
LIWC (Linguistic Inquiry and Word Count)


- [http://www.liwc.net/](http://www.liwc.net/) 2300 words, >70 classes

- **Affective Processes**
  - negative emotion (bad, weird, hate, problem, tough)
  - positive emotion (love, nice, sweet)

- **Cognitive Processes**
  - Tentative (maybe, perhaps, guess),
  - Inhibition (block, constraint)

- **Pronouns** (I/me/you),

- **Negation** (no, never), **Quantifiers** (few, many)
MPQA Subjectivity Cues Lexicon


- 6885 words from 8221 lemmas
  - 2718 positive
  - 4912 negative
- Each word annotated for intensity (strong, weak)
- GNU GPL
type=weaksubj  len=1  word1=abandoned  pos1=adj  stemmed1=n  priorpolarity=negative

type=weaksubj  len=1  word1=abandonment  pos1=noun  stemmed1=n  priorpolarity=negative

type=weaksubj  len=1  word1=abandon  pos1=verb  stemmed1=y  priorpolarity=negative

type=strongsubj  len=1  word1=abase  pos1=verb  stemmed1=y  priorpolarity=negative

type=strongsubj  len=1  word1=abasement  pos1=anypos  stemmed1=y  priorpolarity=negative

type=strongsubj  len=1  word1=abash  pos1=verb  stemmed1=y  priorpolarity=negative

type=weaksubj  len=1  word1=abate  pos1=verb  stemmed1=y  priorpolarity=negative
Bing Liu Opinion Lexicon


- Bing Liu's Page on Opinion Mining
- http://www.cs.uic.edu/~liub/FBS/opinion-lexicon-English.rar

- 6786 words
  - 2006 positive
  - 4783 negative
Lexicons as Features for Logistic Regression

Sentiment Lexicons
Lexicons as Features for Logistic Regression

Emotion Lexicons
Scherer’s typology of affective states

**Emotion**: relatively brief episode of synchronized response of all or most organismic subsystems in response to the evaluation of an event as being of major significance

- angry, sad, joyful, fearful, ashamed, proud, desperate

**Mood**: diffuse affect state…change in subjective feeling, of low intensity but relatively long duration, often without apparent cause

- cheerful, gloomy, irritable, listless, depressed, buoyant

**Interpersonal stance**: affective stance taken toward another person in a specific interaction, coloring the interpersonal exchange

- distant, cold, warm, supportive, contemptuous

**Attitudes**: relatively enduring, affectively colored beliefs, preferences predispositions towards objects or persons

- liking, loving, hating, valuing, desiring

**Personality traits**: emotionally laden, stable personality dispositions and behavior tendencies, typical for a person

- nervous, anxious, reckless, morose, hostile, envious, jealous
Two families of theories of emotion

• Atomic basic emotions
  • A finite list of 6 or 8, from which others are generated

• Dimensions of emotion
  • Valence (positive negative)
  • Arousal (strong, weak)
  • Control
Ekman’s 6 basic emotions:
Surprise, happiness, anger, fear, disgust, sadness

Ekman & Matsumoto
1989
Valence/Arousal Dimensions

- High arousal, low pleasure: anger
- Low arousal, low pleasure: sadness
- High arousal, high pleasure: excitement
- Low arousal, high pleasure: relaxation
Atomic units vs. Dimensions

Distinctive
• Emotions are units.
• Limited number of basic emotions.
• Basic emotions are innate and universal

Dimensional
• Emotions are dimensions.
• Limited # of labels but unlimited number of emotions.
• Emotions are culturally learned.

Adapted from Julia Braverman
One emotion lexicon from each paradigm!

1. 8 basic emotions:
   - NRC Word-Emotion Association Lexicon (Mohammad and Turney 2011)

2. Dimensions of valence/arousal/dominance
   - Warriner, A. B., Kuperman, V., and Brysbaert, M. (2013)

   • Both built using Amazon Mechanical Turk
Plutchick’s wheel of emotion

- 8 basic emotions
- in four opposing pairs:
  - joy–sadness
  - anger–fear
  - trust–disgust
  - anticipation–surprise
NRC Word-Emotion Association Lexicon

Mohammad and Turney 2011

- 10,000 words chosen mainly from earlier lexicons
- Labeled by Amazon Mechanical Turk
- 5 Turkers per hit
- Give Turkers an idea of the relevant sense of the word
- Result:

<table>
<thead>
<tr>
<th>Word</th>
<th>Emotion</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>amazingly</td>
<td>anger</td>
<td>0</td>
</tr>
<tr>
<td>amazingly</td>
<td>anticipation</td>
<td>0</td>
</tr>
<tr>
<td>amazingly</td>
<td>disgust</td>
<td>0</td>
</tr>
<tr>
<td>amazingly</td>
<td>fear</td>
<td>0</td>
</tr>
<tr>
<td>amazingly</td>
<td>joy</td>
<td>1</td>
</tr>
<tr>
<td>amazingly</td>
<td>sadness</td>
<td>0</td>
</tr>
<tr>
<td>amazingly</td>
<td>surprise</td>
<td>1</td>
</tr>
<tr>
<td>amazingly</td>
<td>trust</td>
<td>0</td>
</tr>
<tr>
<td>amazingly</td>
<td>negative</td>
<td>0</td>
</tr>
<tr>
<td>amazingly</td>
<td>positive</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EmoLex</th>
<th># of terms</th>
</tr>
</thead>
<tbody>
<tr>
<td>EmoLex-Uni:</td>
<td></td>
</tr>
<tr>
<td>Unigrams from Macquarie Thesaurus</td>
<td></td>
</tr>
<tr>
<td>adjectives</td>
<td>200</td>
</tr>
<tr>
<td>adverbs</td>
<td>200</td>
</tr>
<tr>
<td>nouns</td>
<td>200</td>
</tr>
<tr>
<td>verbs</td>
<td>200</td>
</tr>
<tr>
<td>EmoLex-Bi:</td>
<td></td>
</tr>
<tr>
<td>Bigrams from Macquarie Thesaurus</td>
<td></td>
</tr>
<tr>
<td>adjectives</td>
<td>200</td>
</tr>
<tr>
<td>adverbs</td>
<td>187</td>
</tr>
<tr>
<td>nouns</td>
<td>200</td>
</tr>
<tr>
<td>verbs</td>
<td>200</td>
</tr>
<tr>
<td>EmoLex-GI:</td>
<td></td>
</tr>
<tr>
<td>Terms from General Inquirer</td>
<td></td>
</tr>
<tr>
<td>negative terms</td>
<td>2119</td>
</tr>
<tr>
<td>neutral terms</td>
<td>4226</td>
</tr>
<tr>
<td>positive terms</td>
<td>1787</td>
</tr>
<tr>
<td>EmoLex-WAL:</td>
<td></td>
</tr>
<tr>
<td>Terms from WordNet Affect Lexicon</td>
<td></td>
</tr>
<tr>
<td>anger terms</td>
<td>165</td>
</tr>
<tr>
<td>disgust terms</td>
<td>37</td>
</tr>
<tr>
<td>fear terms</td>
<td>100</td>
</tr>
<tr>
<td>joy terms</td>
<td>165</td>
</tr>
<tr>
<td>sadness terms</td>
<td>120</td>
</tr>
<tr>
<td>surprise terms</td>
<td>53</td>
</tr>
<tr>
<td>Union</td>
<td>10170</td>
</tr>
</tbody>
</table>
The AMT Hit

Prompt word: startle

Q1. Which word is closest in meaning (most related) to startle?
- automobile
- shake
- honesty
- entertain

Q2. How positive (good, praising) is the word startle?
- startle is not positive
- startle is weakly positive
- startle is moderately positive
- startle is strongly positive

Q3. How negative (bad, criticizing) is the word startle?
- startle is not negative
- startle is weakly negative
- startle is moderately negative
- startle is strongly negative

Q4. How much is startle associated with the emotion joy? (For example, happy and fun are strongly associated with joy.)
- startle is not associated with joy
- startle is weakly associated with joy
- startle is moderately associated with joy
- startle is strongly associated with joy

Q5. How much is startle associated with the emotion sadness? (For example, failure and heartbreak are strongly associated with sadness.)
- startle is not associated with sadness
- startle is weakly associated with sadness
- startle is moderately associated with sadness
- startle is strongly associated with sadness

Q6. How much is startle associated with the emotion fear? (For example, horror and scary are strongly associated with fear.)
- Similar choices as in 4 and 5 above

Q7. How much is startle associated with the emotion anger? (For example, rage and shouting are strongly associated with anger.)
- Similar choices as in 4 and 5 above

Q8. How much is startle associated with the emotion trust? (For example, faith and integrity are strongly associated with trust.)
- Similar choices as in 4 and 5 above

Q9. How much is startle associated with the emotion disgust? (For example, gross and cruelty are strongly associated with disgust.)
- Similar choices as in 4 and 5 above
Lexicon of valence, arousal, and dominance

- Supplementary data: This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License.

Ratings for 14,000 words for emotional dimensions:
- **valence** (the pleasantness of the stimulus)
- **arousal** (the intensity of emotion provoked by the stimulus)
- **dominance** (the degree of control exerted by the stimulus)
Lexicon of valence, arousal, and dominance

- **valence** (the pleasantness of the stimulus)
  - 9: happy, pleased, satisfied, contented, hopeful
  - 1: unhappy, annoyed, unsatisfied, melancholic, despaired, or bored

- **arousal** (the intensity of emotion provoked by the stimulus)
  - 9: stimulated, excited, frenzied, jittery, wide-awake, or aroused
  - 1: relaxed, calm, sluggish, dull, sleepy, or unaroused;

- **dominance** (the degree of control exerted by the stimulus)
  - 9: in control, influential, important, dominant, autonomous, or controlling
  - 1: controlled, influenced, cared-for, awed, submissive, or guided

  Again produced by AMT
Lexicon of valence, arousal, and dominance: Examples

<table>
<thead>
<tr>
<th>Valence</th>
<th>Arousal</th>
<th>Dominance</th>
</tr>
</thead>
<tbody>
<tr>
<td>vacation</td>
<td>8.53</td>
<td>rampage</td>
</tr>
<tr>
<td>happy</td>
<td>8.47</td>
<td>tornado</td>
</tr>
<tr>
<td>whistle</td>
<td>5.7</td>
<td>zucchini</td>
</tr>
<tr>
<td>conscious</td>
<td>5.53</td>
<td>dressy</td>
</tr>
<tr>
<td>torture</td>
<td>1.4</td>
<td>dull</td>
</tr>
</tbody>
</table>
Lexicons as features for logistic regression

Other Useful Lexicons
Concreteness versus abstractness

- The degree to which the concept denoted by a word refers to a perceptible entity.
  - Do concrete and abstract words differ in connotation?
  - Storage and retrieval?
  - Bilingual processing?
  - Relevant for embodied view of cognition (Barsalou 1999 inter alia)
    - Do concrete words activate brain regions involved in relevant perception

  - Supplementary data: This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License.
  - 37,058 English words and 2,896 two-word expressions ("zebra crossing" and "zoom in"),
  - Rating from 1 (abstract) to 5 (concrete)
  - Calibrator words:
    - shirt, infinity, gas, grasshopper, marriage, kick, polite, whistle, theory, and sugar
Concreteness versus abstractness

- Supplementary data: This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License.
- Some example ratings from the final dataset of 40,000 words and phrases
  
  banana 5
  bathrobe 5
  bagel 5
  brisk 2.5
  badass 2.5
  basically 1.32
  belief 1.19
  although 1.07
Lexicons as features for logistic regression

Using the lexicons to detect affect
Lexicons for detecting document affect: Simplest unsupervised method

• Sentiment:
  • Sum the weights of each positive word in the document
  • Sum the weights of each negative word in the document
  • Choose whichever value (positive or negative) has higher sum

• Emotion:
  • Do the same for each emotion lexicon
Lexicons for detecting document affect:
Simplest unsupervised method

\[ f^+ = \sum_{w \text{ s.t. } w \in \text{positivelexicon}} \theta^+_w \text{count}(w) \]

\[ f^- = \sum_{w \text{ s.t. } w \in \text{negativelexicon}} \theta^-_w \text{count}(w) \]

\[ \text{Sentiment} = + \quad \text{if} \quad f^+ > f^- \]
Lexicons for detecting document affect: Slightly more complex unsupervised method

\[ f^+ = \sum_{w \text{ s.t. } w \in \text{positivelexicon}} \theta_w^+ \text{count}(w) \]

\[ f^- = \sum_{w \text{ s.t. } w \in \text{negativelexicon}} \theta_w^- \text{count}(w) \]

\[
\text{sentiment} = \begin{cases} 
+ & \text{if } \frac{f^+}{f^-} > \lambda \\
- & \text{if } \frac{f^-}{f^+} > \lambda \\
0 & \text{otherwise.}
\end{cases}
\]
Lexicons for detecting document affect: Simplest supervised method

• Build a classifier
  • Predict sentiment (or emotion, or etc) given features
  • Use “counts of lexicon categories” as a features
    • LIWC category “cognition” had count of 7
    • NRC Emotion category “anticipation” had count of 2

• Baseline
  • Use counts of all the words and bigrams in the training set
    • Like the naïve bayes algorithm
  • This is hard to beat
  • But only works if the training and test sets are very similar
Lexicons as features for logistic regression

Using the lexicons to detect affect
CS 124/LINGUIST 180
From Languages to Information
Dan Jurafsky
Stanford University

Learning in Logistic Regression
Learning in logistic regression

OK, how does logistic regression set the values for \( w \) and \( b \)?

Just a quick intuition of the algorithm

You won't need this for CS124
- But take cs221, cs229, cs224n, etc. for more details!
Logistic regression is supervised ML

We have a training set that has the correct y for each x!
- \( x^{(1)}, 0 \)
- \( x^{(2)}, 1 \)
- \( x^{(3)}, 1 \ldots \)

But our classifier gives us the estimate \( \hat{y} \) not the true y

\[
P(y = 1) = \sigma(w \cdot x + b)
\]

So we want to find \( w \) and \( b \) that make \( \hat{y} \) closest to \( y \)
How to find $w$ and $b$ that make $\hat{y}$ closest to $y$

Need a metric: how close is current $\hat{y}$ is to the true gold label $y$

- Instead of similarity, it's conventional to measure distance
- And call it the **loss function** or cost function.
  - $\rightarrow$ cross-entropy loss

And we need an algorithm for minimizing this loss.

- $\rightarrow$ stochastic gradient descent
The cross-entropy loss function

\[ L(\hat{y}, y) = \text{How much } \hat{y} \text{ differs from the true } y \]

Want correct class labels of training examples to be more likely.
Choose \( w, b \) that maximize log probability
- of the true \( y \) labels in the training data
- given the observations \( x \).

The resulting loss function is the negative log likelihood loss
- Usually called cross entropy loss
Getting to cross entropy loss
Formally modeling the likelihood $p(y|x)$

Only two discrete outcomes, $y=0$ or $y=1$:

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

If $y=1$: $\hat{y}$

If $y=0$: $1 - \hat{y}$
Getting to cross entropy loss: log likelihood

Our probability:
\[ p(y|x) = \hat{y}^y (1 - \hat{y})^{1-y} \]

Taking log of both sides
- (values that maximize \( p(.) \) will also maximize \( \log p(.) \))
\[
\log p(y|x) = \log [\hat{y}^y (1 - \hat{y})^{1-y}]
= y \log \hat{y} + (1 - y) \log(1 - \hat{y})
\]

This is called the log likelihood of the observation
Getting to cross entropy loss: negative log likelihood

Log likelihood

\[
\log p(y|x) = \log [\hat{y}^y (1 - \hat{y})^{1-y}]
\]

\[
= y \log \hat{y} + (1 - y) \log (1 - \hat{y})
\]

But instead of maximizing log likelihood, we want to minimize a loss:

\[
L_{CE}(\hat{y}, y) = -\log p(y|x) = -[y \log \hat{y} + (1 - y) \log (1 - \hat{y})]
\]

Plugging in \( \hat{y} = \sigma(w \cdot x + b) \)

\[
L_{CE}(w, b) = -[y \log \sigma(w \cdot x + b) + (1 - y) \log (1 - \sigma(w \cdot x + b))]
\]
The cross entropy loss

\[ L_{CE}(\hat{y}, y) = -\log p(y|x) = - [y \log \hat{y} + (1 - y) \log (1 - \hat{y})] \]

\[ L_{CE}(w, b) = - [y \log \sigma(w \cdot x + b) + (1 - y) \log (1 - \sigma(w \cdot x + b))] \]

Why it does the right thing:
If \( y=1 \): \(-\log \hat{y}\)
If \( y=0 \): \(-\log (1-\hat{y})\)

The negative log goes from
- \(0\): neg log of 1, no loss
- \(\infty\): neg log of 0, infinite loss
Extending it to the whole dataset

\[
\log p(\text{training labels}) = \log \prod_{i=1}^{m} p(y^{(i)}|x^{(i)}) \\
= \sum_{i=1}^{m} \log p(y^{(i)}|x^{(i)}) \\
= -\sum_{i=1}^{m} L_{CE}(\hat{y}^{(i)}, y^{(i)}),
\]

\[
\text{Cost}(w, b) = \frac{1}{m} \sum_{i=1}^{m} L_{CE}(\hat{y}^{(i)}, y^{(i)}) \\
= -\frac{1}{m} \sum_{i=1}^{m} y^{(i)} \log \sigma(w \cdot x^{(i)} + b) + (1 - y^{(i)}) \log \left(1 - \sigma(w \cdot x^{(i)} + b)\right)
\]
We want to find parameters that minimize loss

We'll use \( \theta \) to refer to parameters \( w, b \)

\[
\hat{\theta} = \arg\min_{\theta} \frac{1}{m} \sum_{i=1}^{m} L_{CE}(y^{(i)}, x^{(i)}; \theta)
\]
How to find the parameters that minimize a function

Gradient descent
Gradient descent for one scalar variable w

Loss

slope of loss at \( w^1 \) is negative

one step of gradient descent

\( w^1 \)

\( w^{\min} \)

\( 0 \)

(goal)
How much to move

\[
\frac{d}{dw} f(x; w) \text{ weighted by a learning rate } \eta.
\]

\[
w^{t+1} = w^t - \eta \frac{d}{dw} f(x; w)
\]
Now in N dimensions

The gradient is just such a vector; it expresses the directional components of the sharpest slope along each of those $N$ dimensions. If we're just imagining two weight dimensions (say for one weight $w$ and one bias $b$), the gradient might be a vector with two orthogonal components, each of which tells us how much the ground slopes in the $w$ dimension and in the $b$ dimension. Fig. 5.4 shows a visualization:

In an actual logistic regression, the parameter vector $w$ is much longer than 1 or 2, since the input feature vector $x$ can be quite long, and we need a weight $w_i$ for each $x_i$. For each dimension/variable $w_i$ in $w$ (plus the bias $b$), the gradient will have a component that tells us the slope with respect to that variable. Essentially we're asking: "How much would a small change in that variable $w_i$ influence the total loss function $L$?"

In each dimension $w_i$, we express the slope as a partial derivative $\frac{\partial}{\partial w_i} L$ of the loss function. The gradient is then defined as a vector of these partials. We'll represent $\hat{y}$ as $f(x; q)$ to make the dependence on $q$ more obvious:

$$ L(f(x; q), y) = \frac{\partial}{\partial w_1} L(f(x; q), y) \partial w_2 L(f(x; q), y) \ldots \partial w_n L(f(x; q), y) $$

The final equation for updating $q$ based on the gradient is thus

$$ q^{t+1} = q^t - h - L(f(x; q), y) $$

5.4.1 The Gradient for Logistic Regression

In order to update $q$, we need a definition for the gradient $-L(f(x; q), y)$. Recall that for logistic regression, the cross-entropy loss function is:

$$ L_{CE}(w, b) = [y \log(s(w \cdot x + b)) + (1 - y) \log(1 - s(w \cdot x + b))] $$

It turns out that the derivative of this function for one observation vector $x$ is Eq. 5.21 (the interested reader can see Section 5.8 for the derivation of this equation):

$$ \frac{\partial L_{CE}(w, b)}{\partial w_j} = [s(w \cdot x + b) y] x_j $$

(5.21)
Gradient

\[
\nabla_{\theta} L(f(x; \theta), y) = \begin{bmatrix}
\frac{\partial}{\partial w_1} L(f(x; \theta), y) \\
\frac{\partial}{\partial w_2} L(f(x; \theta), y) \\
\vdots \\
\frac{\partial}{\partial w_n} L(f(x; \theta), y)
\end{bmatrix}
\]
What is this gradient? (see textbook for the proof)

\[ L_{CE}(w, b) = -[y \log \sigma(w \cdot x + b) + (1 - y) \log (1 - \sigma(w \cdot x + b))] \]

\[ \frac{\partial L_{CE}(w, b)}{\partial w_j} = [\sigma(w \cdot x + b) - y]x_j \]

Loss for one example is very intuitive!
Difference between predicted and true values, weighted by x
For an entire dataset

\[
Cost(w, b) = -\frac{1}{m} \sum_{i=1}^{m} y^{(i)} \log \sigma(w \cdot x^{(i)} + b) + (1 - y^{(i)}) \log \left(1 - \sigma(w \cdot x^{(i)} + b)\right)
\]

\[
\frac{\partial Cost(w, b)}{\partial w_j} = \sum_{i=1}^{m} \left[ \sigma(w \cdot x^{(i)} + b) - y^{(i)} \right] x_j^{(i)}
\]
Stochastic Gradient Descent

function **STOCHASTIC GRADIENT DESCENT**(*L*, *f*, *x*, *y*) returns θ

# where: *L* is the loss function
# *f* is a function parameterized by θ
# *x* is the set of training inputs *x*(1), *x*(2),..., *x*(n)
# *y* is the set of training outputs (labels) *y*(1), *y*(2),..., *y*(n)

θ ← 0

repeat T times

For each training tuple (*x*(i), *y*(i)) (in random order)

Compute ̂*y*(i) = *f*(*x*(i); θ)  # What is our estimated output ̂*y*?

Compute the loss *L*(*̂y*(i), *y*(i))  # How far off is ̂*y*(i) from the true output *y*(i)?

g ← ∇θ *L*(*f*(*x*(i); θ), *y*(i))  # How should we move θ to maximize loss ?

θ ← θ − η g  # go the other way instead

return θ
Working through an example

Assign sentiment (1 or 0) to an observation document $x$ with 2 features

\[
\begin{align*}
  x_1 &= 3 & \text{(count of positive lexicon words)} \\
  x_2 &= 2 & \text{(count of negative lexicon words)}
\end{align*}
\]

So we need to learn 3 parameters: $w_1$, $w_2$, and $b$.

Let's initialize all weights and bias to 0

\[
\begin{align*}
  w_1 &= w_2 &= b &= 0 \\
  \eta &= 0.1
\end{align*}
\]
An example...

\[ \theta^{t+1} = \theta^t - \eta \nabla_\theta L(f(x^{(i)}; \theta), y^{(i)}) \]

\[
\nabla_{w,b} = \begin{bmatrix}
\frac{\partial L_{CE}(w,b)}{\partial w_1} \\
\frac{\partial L_{CE}(w,b)}{\partial w_2} \\
\frac{\partial L_{CE}(w,b)}{\partial b}
\end{bmatrix} = \begin{bmatrix}
(\sigma(w \cdot x + b) - y)x_1 \\
(\sigma(w \cdot x + b) - y)x_2 \\
\sigma(w \cdot x + b) - y
\end{bmatrix} = \begin{bmatrix}
(\sigma(0) - 1)x_1 \\
(\sigma(0) - 1)x_2 \\
\sigma(0) - 1
\end{bmatrix} = \begin{bmatrix}
-0.5x_1 \\
-0.5x_2 \\
-0.5
\end{bmatrix} = \begin{bmatrix}
-1.5 \\
-1.0 \\
-0.5
\end{bmatrix}
\]

\[ \theta^2 = \begin{bmatrix}
w_1 \\
w_2 \\
b
\end{bmatrix} - \eta \begin{bmatrix}
-1.5 \\
-1.0 \\
-0.5
\end{bmatrix} = \begin{bmatrix}
.15 \\
.1 \\
.05
\end{bmatrix} \]

So after one step of gradient descent, the weights have shifted to be:

\[ w_1 = .15, w_2 = .1, \text{and } b = .05 \]
Learning in Logistic Regression