Text classification

We might want to categorize the content of the text:

- Spam detection (binary classification: spam/not spam)
- Sentiment analysis (binary or multiway)
  - movie, restaurant, product reviews (pos/neg, or 1-5 stars)
  - political argument (pro/con, or pro/con/neutral)
- Topic classification (multiway: sport/finance/travel/etc)

Text classification

Or we might want to categorize the author of the text (authorship attribution):

- Native language identification (e.g., to tailor language tutoring)
- Diagnosis of disease (psychiatric or cognitive impairments)
- Identification of gender, dialect, educational background (e.g., in forensics [legal matters], advertising/marketing)
N-gram models for classification?

N-gram models can sometimes be used for classification (see Lab 2: identifying specific authors). But

- For many tasks, sequential relationships between words are largely irrelevant: we can just consider the document as a **bag of words**.

- On the other hand, we may want to include other kinds of features (e.g., part of speech tags) that N-gram models don’t include.

Here we consider two alternative models for classification:

- **Naive Bayes** (this should be review)
- **Maximum Entropy** (aka **multinomial logistic regression**).

Naive Bayes: high-level formulation

- Given document $d$ and set of categories $C$ (say, spam/not-spam), we want to assign $d$ to the most probable category $\hat{c}$.

  $\hat{c} = \arg\max_{c \in C} P(c|d)$

Naive Bayes: high-level formulation

- Given document $d$ and set of categories $C$ (say, spam/not-spam), we want to assign $d$ to the most probable category $\hat{c}$.

  $\hat{c} = \arg\max_{c \in C} P(c|d)$

  $= \arg\max_{c \in C} \frac{P(d|c)P(c)}{P(d)}$

  $= \arg\max_{c \in C} P(d|c)P(c)$

- Just as in spelling correction, we need to define $P(d|c)$ and $P(c)$. 

Figure from J&M 3rd ed. draft, sec 7.1
How to model $P(d|c)$?

- First, define a set of features that might help classify docs.
  - Here we’ll assume these are all the words in the vocabulary.
  - But, we could just use some words (more on this later...).
  - Or, use other info, like parts of speech, if available.

- We then represent each document $d$ as the set of features (words) it contains: $f_1, f_2, \ldots f_n$. So

$$P(d|c) = P(f_1, f_2, \ldots f_n|c)$$

Naive Bayes assumption

- Effectively, we only care about the count of each feature in each document.
- For example, in spam detection:

<table>
<thead>
<tr>
<th></th>
<th>the</th>
<th>your</th>
<th>model</th>
<th>cash</th>
<th>Viagra</th>
<th>class</th>
<th>account</th>
<th>orderz</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc 1</td>
<td>12</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>doc 2</td>
<td>10</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>doc 3</td>
<td>25</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>doc 4</td>
<td>14</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>doc 5</td>
<td>17</td>
<td>5</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Naive Bayes classifier

Putting together the pieces, our complete classifier definition:

- Given a document with features $f_1, f_2, \ldots f_n$ and set of categories $C$, choose the class $\hat{c}$ where

$$\hat{c} = \arg\max_{c \in C} P(c) \prod_{i=1}^{n} P(f_i|c)$$

- $P(c)$ is the prior probability of class $c$ before observing any data.
- $P(f_i|c)$ is the probability of seeing feature $f_i$ in class $c$. 

Naive Bayes assumption

- As in LMs, we can’t accurately estimate $P(f_1, f_2, \ldots f_n|c)$ due to sparse data.
- So, make a naive Bayes assumption: features are conditionally independent given the class.

$$P(f_1, f_2, \ldots f_n|c) \approx P(f_1|c)P(f_2|c)\ldots P(f_n|c)$$

- That is, the prob. of a word occurring depends only on the class.
  - Not on which words occurred before or after (as in N-grams)
  - Or even which other words occurred at all
Estimating the class priors

- $P(c)$ normally estimated with MLE:
  \[
  \hat{P}(c) = \frac{N_c}{N}
  \]
  
  - $N_c$ = the number of training documents in class $c$
  - $N$ = the total number of training documents

- So, $\hat{P}(c)$ is simply the proportion of training documents belonging to class $c$.

Learning the class priors: example

- Given training documents with correct labels:

<table>
<thead>
<tr>
<th></th>
<th>the</th>
<th>your</th>
<th>model</th>
<th>cash</th>
<th>Viagra</th>
<th>class</th>
<th>account</th>
<th>orderz</th>
<th>spam?</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc 1</td>
<td>12</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>doc 2</td>
<td>10</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>+</td>
</tr>
<tr>
<td>doc 3</td>
<td>25</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>doc 4</td>
<td>14</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>+</td>
</tr>
<tr>
<td>doc 5</td>
<td>17</td>
<td>5</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>+</td>
</tr>
</tbody>
</table>

- $\hat{P}(\text{spam}) = \frac{3}{5}$

Learning the feature probabilities

- $P(f_i|c)$ normally estimated with simple smoothing:
  \[
  \hat{P}(f_i|c) = \frac{\text{count}(f_i, c) + \alpha}{\sum_{f \in F}(\text{count}(f, c) + \alpha)}
  \]
  
  - $\text{count}(f_i, c)$ = the number of times $f_i$ occurs in class $c$
  - $F$ = the set of possible features
  - $\alpha$: the smoothing parameter, optimized on held-out data

Learning the feature probabilities: example

<table>
<thead>
<tr>
<th></th>
<th>the</th>
<th>your</th>
<th>model</th>
<th>cash</th>
<th>Viagra</th>
<th>class</th>
<th>account</th>
<th>orderz</th>
<th>spam?</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc 1</td>
<td>12</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>doc 2</td>
<td>10</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>+</td>
</tr>
<tr>
<td>doc 3</td>
<td>25</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>doc 4</td>
<td>14</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>+</td>
</tr>
<tr>
<td>doc 5</td>
<td>17</td>
<td>5</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>+</td>
</tr>
</tbody>
</table>
Learning the feature probabilities: example

<table>
<thead>
<tr>
<th>the</th>
<th>your</th>
<th>model</th>
<th>cash</th>
<th>Viagra</th>
<th>class</th>
<th>account</th>
<th>orderz</th>
<th>spam?</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc 1</td>
<td>12</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>doc 2</td>
<td>10</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>+</td>
</tr>
<tr>
<td>doc 3</td>
<td>25</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>doc 4</td>
<td>14</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>+</td>
</tr>
<tr>
<td>doc 5</td>
<td>17</td>
<td>5</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>+</td>
</tr>
</tbody>
</table>

\[ \hat{P}(\text{your}|+) = \frac{(4+2+5+\alpha)}{(\text{all words in + class})+\alpha F} = \frac{(11 + \alpha)}{(68 + \alpha F)} \]

Classifying a test document: example

- Test document \( d \):
  
  get your cash and your orderz

- Suppose all features not shown earlier have \( \hat{P}(f_i|+) = \frac{\alpha}{(68 + \alpha F)} \)

\[
P(+|d) \propto P(+) \prod_{i=1}^{n} P(f_i|+) \]
\[
\phantom{P(+|d)} = P(+) \cdot \frac{\alpha}{(68 + \alpha F)} \cdot \frac{11 + \alpha}{(68 + \alpha F)} \cdot \frac{7 + \alpha}{(68 + \alpha F)} \cdot \frac{\alpha}{(68 + \alpha F)} \cdot \frac{11 + \alpha}{(68 + \alpha F)} \cdot \frac{2 + \alpha}{(68 + \alpha F)} \]
Alternative feature values and feature sets

- Use only binary values for $f_i$: did this word occur in $d$ or not?
- Use only a subset of the vocabulary for $F$
  - Ignore stopwords (function words and others with little content)
  - Choose a small task-relevant set (e.g., using a sentiment lexicon)
- Use more complex features (bigrams, syntactic features, morphological features, ...)

Choosing features can be tricky

- For example, sentiment analysis might need domain-specific non-sentiment words
  - Such as quiet, memory for computer product reviews.
- And for other tasks, stopwords might be very useful features
  - E.g., People with schizophrenia use more 2nd-person pronouns (?), those with depression use more 1st-person (?).
- Probably better to use too many irrelevant features than not enough relevant ones.

Task-specific features

Example words from a sentiment lexicon:

Positive:
- absolutely
- beaming
- calm
- celebrated
- champ
- charming
- cheery
- choice
- classic
- classical
- clean

Negative:
- abysmal
- bad
- callous
- can’t
- clumsy
- coarse
- cold
- collapse
- contradictory
- contrary
- corrosive
- corrupt

From http://www.enchantedlearning.com/wordlist/

Advantages of Naive Bayes

- Very easy to implement
- Very fast to train, and to classify new documents (good for huge datasets).
- Doesn’t require as much training data as some other methods (good for small datasets).
- Usually works reasonably well
- This should be your baseline method for any classification task
Problems with Naive Bayes

• Naive Bayes assumption is naive!

• Consider categories Travel, Finance, Sport.

• Are the following features independent given the category?
  beach, sun, ski, snow, pitch, palm, football, relax, ocean

– No! Ex: Given Travel, seeing beach makes sun more likely, but ski less likely.
– Defining finer-grained categories might help (beach travel vs ski travel), but we don’t usually want to.

Non-independent features

• Features are not usually independent given the class

• Adding multiple feature types (e.g., words and morphemes) often leads to even stronger correlations between features

• Accuracy of classifier can sometimes still be ok, but it will be highly overconfident in its decisions.
  – Ex: NB sees 5 features that all point to class 1, treats them as five independent sources of evidence.
  – Like asking 5 friends for an opinion when some got theirs from each other.

A less naive approach

• Although Naive Bayes is a good starting point, often we have enough training data for a better model (and not so much that slower performance is a problem).

• We may be able to get better performance using loads of features and a model that doesn’t assume features are conditionally independent.

• Namely, a Maximum Entropy model.
MaxEnt classifiers

- Used widely in many different fields, under many different names
- Most commonly, multinomial logistic regression
  - multinomial if more than two possible classes
  - otherwise (or if lazy) just logistic regression
- Also called: log-linear model, one-layer neural network, single neuron classifier, etc ...
- The mathematical formulation here (and in the text) looks slightly different from standard presentations of mult. logistic regression, but is ultimately equivalent.

Naive Bayes vs MaxEnt

- Like Naive Bayes, MaxEnt assigns a document \( d \) to class \( \hat{c} \), where
  \[
  \hat{c} = \arg\max_{c \in C} P(c|d)
  \]
- Unlike Naive Bayes, we do not apply Bayes’ Rule. Instead, we model \( P(c|d) \) directly.

Example: classify by topic

- Given a web page document, which topic does it belong to?
  - \( \vec{x} \) are the words in the document, plus info about headers and links.
  - \( c \) is the latent class. Assume three possibilities:
  \[
  c = \begin{array}{c}
  \text{class} \\
  \text{1} \quad \text{TRAVEL} \\
  \text{2} \quad \text{SPORT} \\
  \text{3} \quad \text{FINANCE}
  \end{array}
  \]

Feature functions

- Like Naive Bayes, MaxEnt models use features we think will be useful for classification.
- However, features are treated differently in the two models:
  - NB: features are directly observed (e.g., words in doc): no difference between features and data.
  - MaxEnt: we will use \( \vec{x} \) to represent the observed data. Features are functions that depend on both observations \( \vec{x} \) and class \( c \).

This way of treating features in MaxEnt is standard in NLP; in ML it’s often explained differently.
MaxEnt feature example

- If we have three classes, our features will always come in groups of three. For example, we could have three binary features:
  
  \[ f_1 : \text{contains('ski') \& \ c = 1} \]
  \[ f_2 : \text{contains('ski') \& \ c = 2} \]
  \[ f_3 : \text{contains('ski') \& \ c = 3} \]

  - training docs from class 1 that contain ski will have \( f_1 \) active;
  - training docs from class 2 that contain ski will have \( f_2 \) active;
  - etc.

- Each feature \( f_i \) has a real-valued weight \( w_i \) (learned in training).

Classification with MaxEnt

Choose the class that has highest probability according to

\[
P(c|\vec{x}) = \frac{1}{Z} \exp \left( \sum_i w_i f_i(\vec{x}, c) \right)
\]

where the normalization constant \( Z = \sum_{c'} \exp(\sum_i w_i f_i(\vec{x}, c')) \)

- Inside brackets is just a dot product: \( \vec{w} \cdot \vec{f} \).
- And \( P(c|\vec{x}) \) is a monotonic function of this dot product.
- So, we will end up choosing the class for which \( \vec{w} \cdot \vec{f} \) is highest.

Classification example

\[
\begin{align*}
f_1 & : \text{contains('ski') \& \ c = 1} & w_1 &= 1.2 \\
f_2 & : \text{contains('ski') \& \ c = 2} & w_2 &= 2.3 \\
f_3 & : \text{contains('ski') \& \ c = 3} & w_3 &= -0.5 \\
f_4 & : \text{link_to('expedia.com') \& \ c = 1} & w_4 &= 4.6 \\
f_5 & : \text{link_to('expedia.com') \& \ c = 2} & w_5 &= -0.2 \\
f_6 & : \text{link_to('expedia.com') \& \ c = 3} & w_6 &= 0.5 \\
f_7 & : \text{num_links \& \ c = 1} & w_7 &= 0.0 \\
f_8 & : \text{num_links \& \ c = 2} & w_8 &= 0.2 \\
f_9 & : \text{num_links \& \ c = 3} & w_9 &= -0.1
\end{align*}
\]

- \( f_7, f_8, f_9 \) are numeric features that count outgoing links.

• Suppose our test document contains ski and 6 outgoing links.

  - We don’t know \( c \) for this doc, so we try out each possible value.
    - Travel: \( \sum_i w_i f_i(\vec{x}, c = 1) = 1.2 + (0.0)(6) = 1.2 \).
    - Sport: \( \sum_i w_i f_i(\vec{x}, c = 2) = 2.3 + (0.2)(6) = 3.5 \).
    - Finance: \( \sum_i w_i f_i(\vec{x}, c = 3) = -0.5 + (-0.1)(6) = -1.1 \).

  • We’d need to do further work to compute the probability of each class, but we know already that Sport will be the most probable.
Feature templates

- In practice, features are usually defined using templates
  
  $\text{contains}(w) \land c$
  $\text{header} \text{contains}(w) \land c$
  $\text{header}_\text{contains}(w) \land \text{link}_\text{in}_\text{header} \land c$
  
  - instantiate with all possible words $w$ and classes $c$
  - usually filter out features occurring very few times

- NLP tasks often have a few templates, but 1000s or 10000s of features

Training the model

- Given annotated data, choose weights that make the class labels most probable under the model.

- That is, given items $x^{(1)} \ldots x^{(N)}$ with labels $c^{(1)} \ldots c^{(N)}$, choose

  $$\hat{w} = \arg\max_{\vec{w}} \sum_j \log P(c^{(j)}|x^{(j)})$$

- called conditional maximum likelihood estimation (CMLE)

- Like MLE, CMLE will overfit, so we use tricks (regularization) to avoid that.

The downside to MaxEnt models

- Supervised MLE in generative models is easy: compute counts and normalize.

- Supervised CMLE in MaxEnt model not so easy
  
  - requires multiple iterations over the data to gradually improve weights (using gradient ascent).
  
  - each iteration computes $P(c^{(j)}|x^{(j)})$ for all $j$, and each possible $c^{(j)}$.
  
  - this can be time-consuming, especially if there are a large number of classes and/or thousands of features to extract from each training example.
Summary

• Two methods for text classification: Naive Bayes, MaxEnt

• Make different independence assumptions, have different training requirements.

• Both are easily available in standard ML toolkits.

• Both require some work to figure out what features are good to use.