Classification: Naïve Bayes

Nathan Schneider
(slides adapted from Chris Dyer, Noah Smith, et al.)
ENLP | 19 September 2016
Sentiment Analysis

• Recall the task:

Filled with horrific dialogue, laughable characters, a laughable plot, ad really no interesting stakes during this film, "Star Wars Episode I: The Phantom Menace" is not at all what I wanted from a film that is supposed to be the huge opening to the segue into the fantastic Original Trilogy. The positives include the score, the sound
Sentiment Analysis

• Recall the task:

Filled with horrific dialogue, laughable characters, a laughable plot, and really no interesting stakes during this film, "Star Wars Episode I: The Phantom Menace" is not at all what I wanted from a film that is supposed to be the huge opening to the segue into the fantastic Original Trilogy. The positives include the score, the sound.

• This is a classification task: we have open-ended text as input and a fixed set of discrete classes as output.

• By convention, the input/observed information is denoted $x$, and the output/predicted information is $y$. 
A Rule-based Classifier

good = {'yay', 'cool', ...}
bad = {'ugh', ':(', ...}

score = 0
for w in x:
    if w in good:
        score += 1
    elif w in bad:
        score -= 1
return int(score > 0)
Supervised Classification

• We can probably do better with data
  ‣ Our intuitions about word sentiment aren’t perfect

• **Supervised** = generalizations are *learned* from *labeled* data
  ‣ So, we need a **training** corpus of reviews with gold (correct) sentiment labels
  ‣ And a learning algorithm

• This course: **inductive** learning algorithms—collect statistics from training corpus, but the resulting classifier does not rely on the training corpus itself
A **Rule-based Classifier**

```
good = {...from training data...}
bad = {...from training data...}

score = 0
for w in x:
    if w in good:
        score += 1
    elif w in bad:
        score -= 1
return int(score>0)
```
Notation

- Training examples: \( X = (x_1, x_2, ..., x_N) \)

- Their categories: \( Y = (y_1, y_2, ..., y_N) \)

- A classifier \( C \) seeks to map \( x_i \) to \( y_i \): \( x \rightarrow C \rightarrow y \)

- A learner \( L \) infers \( C \) from \( (X, Y) \): \( X \rightarrow L \rightarrow C \rightarrow Y \)
from collections import Counter
scores = Counter()
for x,y in zip(X,Y):
    for w in x:
        if y==THUMBS_UP:
            scores[w] += 1
        elif y==THUMBS_DOWN:
            scores[w] -= 1

good, bad = set(), set()
for w,score in scores.items():
    if score>0: good.add(w)
    else: bad.add(w)

return good, bad
Limitations

• Our classifier doesn’t know that:
  ‣ Some words are more strongly indicative of sentiment than others
  ‣ The data may skew positive or negative (e.g., more or longer positive reviews than negative)
  ‣ Infrequent words may occur only in the positive examples or only in the negative examples by accident

• Instead of raw counts, we can use a probabilistic model
Review Questions: Conditional Probability

1. If \( p \) is a probability mass function, which is true by the definition of conditional probability:
\[
p(x \mid y, z) =
\]

 a. \( \frac{p(x)}{p(y,z)} \)

 b. \( \frac{p(y)p(z)}{p(x,y,z)} \)

 c. \( \frac{p(x,y,z)}{p(y,z)} \)

 d. \( p(x)p(x \mid y)p(x \mid z) \)
2. Which is/are guaranteed to be true?

a. \( \forall y \forall z, \sum_x p(x \mid y, z) = 1 \)

b. \( \forall x, \sum_y \sum_z p(x \mid y, z) = 1 \)

c. \( \sum_x p(x) = 1 \)

d. \( \forall y \forall z, \sum_x p(x)p(y \mid x)p(z \mid x, y) = 1 \)
Probabilistic Classifiers

\[
\text{return } \arg \max_{y'} p(y' \mid x)
\]
Probabilistic Classifiers

\[
\text{return } \arg \max_{y'} p(y' \mid x)
\]
Filled with horrific dialogue, laughable characters, a laughable plot, and really no interesting stakes during this film, "Star Wars Episode I: The Phantom Menace" is not at all what I wanted from a film that is supposed to be the huge opening to the segue into the fantastic Original Trilogy. The positives include the score, the sound
Probabilistic Classifiers

return
arg max_y' p(y' | x)

= p(y' | Filled, with, horrific, …)

How can we compute this?

We can’t compute the usual MLE unless this exact document appeared in the training data!
A probabilistic model that generalizes

- Instead of estimating $p(y' | \text{Filled, with, horrific, ...})$ directly, we make two **modeling assumptions**:

  1. The **Bag of Words (BoW) assumption**: Assume the order of the words in the document is irrelevant to the task. I.e., stipulate that $p(y' | \text{Filled, with, horrific}) = p(y' | \text{Filled, horrific, with})$
Art installation in CMU’s Machine Learning Department
I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!

Figure 7.1 Intuition of the multinomial naive Bayes classifier applied to a movie review. The position of the words is ignored (the bag of words assumption) and we make use of the frequency of each word.

Figure from J&M 3rd ed. draft, sec 7.1
A probabilistic model that generalizes

• Instead of estimating $p(y' \mid \text{Filled, with, horrific, ...})$ directly, we make two modeling assumptions:

1. The Bag of Words (BoW) assumption: Assume the order of the words in the document is irrelevant to the task. I.e., stipulate that

   $$p(y' \mid \text{Filled, with, horrific}) = p(y' \mid \text{Filled, horrific, with})$$

So called because a bag or multiset is a data structure that stores counts of elements, but not their order.
A probabilistic model that generalizes

- The BoW assumption isn’t enough, though, unless documents with all the same words occurred in the training data. Hence:

2. The naïve Bayes assumption: Assume the words are independent conditioned on the class $y'$

$$p(\text{Filled, with, horrific } | \ y') = p(\text{Filled } | \ y') \times p(\text{with } | \ y') \times p(\text{horrific } | \ y')$$

Hang on, we actually wanted:

$$p(y' | \text{Filled, with, horrific})$$

How to reverse the order?
Bayes’ Rule

\[ p(B \mid A) = \frac{p(B) \times p(A \mid B)}{p(A)} \]
\[ p(B \mid A) = \frac{p(B \times p(A \mid B))}{p(A)} \]

multiply both sides by \( p(A) \)

\[ p(A) \times p(B \mid A) = p(B) \times p(A \mid B) \]

Chain Rule

\[ p(A, B) = p(B, A) \]

...which is true by definition of joint probability
Bayes’ Rule

\[ p(B \mid A) = \frac{p(B) \times p(A \mid B)}{p(A)} \]

\[ p(B \mid A) \propto p(B) \times p(A \mid B) \]

posterior \quad prior \quad likelihood
A probabilistic model that generalizes

- The BoW assumption isn’t enough, though, unless documents with all the same words occurred in the training data. Hence:

2. The **naïve Bayes assumption**: Assume the words are **independent** conditioned on the class $y'$

$$p(\text{Filled, with, horrific} \mid y') = p(\text{Filled} \mid y') \times p(\text{with} \mid y') \times p(\text{horrific} \mid y')$$

Hang on, we actually wanted:

$$p(y' \mid \text{Filled, with, horrific})$$

How to reverse the order?
A probabilistic model that generalizes

- The BoW assumption isn’t enough, though, unless documents with all the same words occurred in the training data. Hence:

2. The **naïve Bayes assumption**: Assume the words are **independent** conditioned on the class $y'$

$$p(\text{Filled, with, horrific} \mid y')$$

$$= p(\text{Filled} \mid y') \times p(\text{with} \mid y') \times p(\text{horrific} \mid y')$$

$$p(y' \mid \text{Filled, with, horrific})$$

$$\propto p(y') \times p(\text{Filled, with, horrific} \mid y')$$

$$= p(y') \times p(\text{Filled} \mid y') \times p(\text{with} \mid y') \times p(\text{horrific} \mid y')$$
Is this a good model?

- What is wrong with these assumptions?
Is this a good model?

• George Box, statistician: “essentially, all models are wrong, but some are useful”

• It turns out that naïve Bayes + BoW works pretty well for many text classification tasks, like spam detection.
Naïve Bayes Classifier

In other words: Loop over class labels, choose the one that makes the document most probable (prior \times likelihood)
Naïve Bayes Learner

\[ X \rightarrow \forall y, p(y) \leftarrow \frac{\text{count}(y)}{N} \]

\[ Y \rightarrow \forall y, \forall w, p(w \mid y) \leftarrow \frac{\text{count}(w, y)}{\text{count}(y)} \]

\[ p(\text{horrific} \mid \text{thumbs up}) \leftarrow \frac{\# \text{docs with horrorific}}{\# \text{docs}} \]
Parameters

• Each probability (or other value) that is \textit{learned} and used by the classifier is called a \textit{parameter}
  ‣ E.g., a single probability in a distribution

• Naïve Bayes has two kinds of distributions:
  ‣ the class prior distribution, \( p(y) \)
  ‣ the likelihood distribution, \( p(w \mid y) \)

• So how many parameters total, if there are \( K \) classes and \( V \) words in the training data?
Smoothing \( p(w \mid y) \)

- What if we encounter the word distraught in a test document, but it has never been seen in training?
  - Can’t estimate \( p(\text{distraught} \mid \text{👍}) \) or \( p(\text{distraught} \mid \text{👎}) \): numerator will be 0
  - Because the word probabilities are multiplied together for each document, the probability of the whole document will be 0
Smoothing $p(w | y)$

- Smoothing techniques adjust probabilities to avoid overfitting to the training data
  - Above: **Laplace (add-1) smoothing**
  - OOV (out-of-vocabulary/unseen) words now have small probability, which decreases the model’s confidence in the prediction without ignoring the other words
  - Probability of each seen word is reduced slightly to save probability mass for unseen words

$V$ is the size of the vocabulary of the training corpus

$$p(\text{horrific} | \text{👍}) \leftarrow \frac{(# \text{👍 docs with horrific}) + 1}{(# \text{👍 docs}) + V + 1}$$

$$p(\text{OOV} | \text{👍}) \leftarrow \frac{1}{(# \text{👍 docs}) + V + 1}$$
Naïve Bayes Classifier

\( w_j \leftarrow \text{[words}(x)\text{]}_j \)

return

arg max\( _{y'} \) \( p(y') \times \prod_j p(w_j \mid y') \)

In other words: Loop over class labels, choose the one that makes the document most probable (prior \( \times \) likelihood)

Can get very small

\( x \rightarrow y \)
Avoiding Underflow

- Multiplying 2 very small floating point numbers can yield a number that is too small for the computer to represent. This is called underflow.

- In implementing probabilistic models, we use log probabilities to get around this.
  - Instead of storing p(•), store log p(•)
  - p(•) × p′(•) → log p(•) + log p′(•)
  - p(•) + p′(•) → numpy.logaddexp(log p(•), log p′(•))
Noisy Channel Model

$p(y)$

source $\rightarrow y \rightarrow channel \rightarrow x$

$p(x \mid y)$

What proportion of emails are expected to be spam vs. not spam?

What proportion of product reviews are expected to get 1,2,3,4,5 stars?
Noisy Channel Classifiers

\[
\text{return } \arg \max_y p(y) \times p(x \mid y)
\]
Conclusions

- We have seen how labeled **training data** and **supervised learning** can produce a better-informed classifier

  ‣ **Classifier** takes an *input* (such as a text document) and predicts an *output* (such as a class label)

  ‣ **Learner** takes *training data* and produces (statistics necessary for) the classifier
Conclusions

• Because most pieces of text are unique, it’s not very practical to assume the one being classified has occurred in the training data
  ‣ We need to make **modeling assumptions** that help the learner to **generalize** to unseen inputs

• The **naïve Bayes** model + **bag-of-words** assumption are a simple, fast probabilistic approach to text classification
  ‣ Works well for many tasks, despite being a dumb naïve model of language: We know that
    * good, not as bad as expected ≠ bad, not as good as expected
    * \( p(\text{Star Wars} | ✏) \neq p(\text{Star} | ✏) \times p(\text{Wars} | ✏) \)
Conclusions

• In practice, we need smoothing to avoid assuming that everything that might come up at test time is in the training data

• Implementation trick: use log probabilities to avoid underflow
• Quiz 1 is graded in Canvas. ✓+, ✓, or ✓−. Answers posted. Ask James if you want yours back.

• Office hours: who can’t make Tu 3-4 (me) / Th 2-3 (James)?

• A0, A1

• We’ll drop your lowest quiz grade & lowest homework grade

• Language Lighting Presentations

• Readings