NAĪVE BAYES CONTINUED

David Kauchak
CS159 Fall 2020

Admin

Assignment 6b

No class Tuesday

Assignment 7 out Monday

Final project

1. Your project should relate to something involving NLP

2. Your project must include a solid experimental evaluation

3. Your project should be in a group of 2-4. If you’d like to do it solo, please come talk to me.

Final project

<table>
<thead>
<tr>
<th>date</th>
<th>time</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>11/5</td>
<td>in-class</td>
<td>Project proposal presentation</td>
</tr>
<tr>
<td>11/11</td>
<td>11:50pm</td>
<td>Project proposal write-up</td>
</tr>
<tr>
<td></td>
<td>11:50pm</td>
<td>Status report</td>
</tr>
<tr>
<td>11/23</td>
<td>11:50pm</td>
<td>Paper draft</td>
</tr>
<tr>
<td>11/24</td>
<td>in-class</td>
<td>Presentation</td>
</tr>
<tr>
<td>11/25</td>
<td>11:50pm</td>
<td>Final paper and code</td>
</tr>
</tbody>
</table>

Read the final project handout ASAP!

Start forming groups and thinking about what you want to do
Final project ideas

- pick a text classification task
  - evaluate different machine learning methods
  - implement a machine learning method
  - analyze different feature categories

- n-gram language modeling
  - implement and compare alternative smoothing techniques
  - implement alternative models

- parsing
  - lexicalized PCFG (with smoothing)
  - n-best list generation
  - parse output reranking
  - implement another parsing approach and compare
  - parsing non-traditional domains (e.g., n-iter)

- EM
  - try and implement EM model 2
  - word level translation models

Final project application areas

- spelling correction
- part of speech tagger
- text chunker
- dialogue generation
- pronoun resolution
- compare word similarity measures (more than the ones we looked at)
- word sense disambiguation
- machine translation
- information retrieval
- information extraction
- question answering
- summarization
- speech recognition

Basic steps for probabilistic modeling

- Step 1: pick a model
- Step 2: figure out how to estimate the probabilities for the model
- Step 3 (optional): deal with overfitting

Probabilistic models

Which model do we use, i.e., how do we calculate \( p(\text{feature, label}) \)?

How do we train the model, i.e., how do we estimate the probabilities for the model?

How do we deal with overfitting?

Naive Bayes assumption

\[
p(\text{features, label}) = p(y) \prod_{i=1}^{m} p(x_i | y, x_1, ..., x_{i-1})
\]

\[
p(x_i | y, x_1, x_2, ..., x_{i-1}) = p(x_i | y)
\]

Assumes feature \( i \) is independent of the other features given the label
Generative Story

To classify with a model, we're given an example and we obtain the probability

We can also ask how a given model would generate an example

This is the “generative story” for a model

Looking at the generative story can help understand the model

We also can use generative stories to help develop a model

Bernoulli NB generative story

\[ p(y) \prod_{j=1}^{m} p(x_j \mid y) \]

1. Pick a label according to \( p(y) \)
   - roll a biased, \( \text{num\_labels} \)-sided die
2. For each feature:
   - Flip a biased coin:
     - if heads, include the feature
     - if tails, don't include the feature

What does this mean for text classification, assuming unigram features?

Bernoulli NB

Pros
- Easy to implement
- Fast!
- Can be done on large data sets

Cons
- Naive Bayes assumption is generally not true
- Performance isn't as good as other models
- For text classification (and other sparse feature domains) the \( p(x_i=0 \mid y) \) can be problematic
Another generative story
Randomly draw words from a “bag of words” until document length is reached

Draw words from a fixed distribution
Selected:

Draw words from a fixed distribution
Selected:
Put a copy of w1 back

Draw words from a fixed distribution
Selected:

Draw words from a fixed distribution
Selected:
Draw words from a fixed distribution

Selected: \( w_1 \), \( w_1 \), \( w_1 \)

Put a copy of \( w_1 \) back

Draw words from a fixed distribution

Selected: \( w_1 \), \( w_1 \), \( w_2 \)

Draw words from a fixed distribution

Selected: \( w_1 \), \( w_1 \), \( w_2 \)

Put a copy of \( w_2 \) back

Draw words from a fixed distribution

Selected: \( w_1 \), \( w_1 \), \( w_2 \)

Put a copy of \( w_2 \) back

Draw words from a fixed distribution

Selected: \( w_1 \), \( w_1 \), \( w_2 \), \( w_2 \), ...

Put a copy of \( w_2 \) back
Is this a NB model, i.e. does it assume each individual word occurrence is independent?

Yes! Doesn’t matter what words were drawn previously, still the same probability of getting any particular word.

Does this model handle multiple word occurrences?

Selected: w1, w2, w3, ...

w1, w2, w3, w1, w1, w3, w1
NB generative story

Bernoulli NB
1. Pick a label according to $p(y)$
2. Roll a biased, num_labels-sided die
3. For each word in your vocabulary:
   - Flip a biased coin:
     - If heads, include the word in the text
     - If tails, don’t include the word

Multinomial NB
1. Pick a label according to $p(y)$
2. Roll a biased, num_labels-sided die
3. Keep drawing words from $p(\text{words} | y)$ until text length has been reached.

Probabilities

Bernoulli NB
1. Pick a label according to $p(y)$
2. Roll a biased, num_labels-sided die
3. For each word in your vocabulary:
   - Flip a biased coin:
     - If heads, include the word in the text
     - If tails, don’t include the word

Multinomial NB
1. Pick a label according to $p(y)$
2. Roll a biased, num_labels-sided die
3. Keep drawing words from $p(\text{words} | y)$ until document length has been reached

A digression: rolling dice
What’s the probability of getting a 3 for a single roll of this dice?
1/6

A digression: rolling dice
What is the probability distribution over possible single rolls?

<table>
<thead>
<tr>
<th>1</th>
<th>1/6</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1/6</td>
</tr>
<tr>
<td>3</td>
<td>1/6</td>
</tr>
<tr>
<td>4</td>
<td>1/6</td>
</tr>
<tr>
<td>5</td>
<td>1/6</td>
</tr>
<tr>
<td>6</td>
<td>1/6</td>
</tr>
</tbody>
</table>
A digression: rolling dice

What if I told you 1 was twice as likely as the others?

\[
\begin{array}{cccccc}
\text{2/7} & \text{1/7} & \text{1/7} & \text{1/7} & \text{1/7} & \text{1/7} \\
1 & 2 & 3 & 4 & 5 & 6
\end{array}
\]

What if I rolled 400 times and got the following number?

\[
\begin{array}{cccccc}
\text{1: 100} & \text{2: 50} & \text{3: 50} & \text{4: 100} & \text{5: 50} & \text{6: 50} \\
1/4 & 1/8 & 1/8 & 1/4 & 1/8 & 1/8 \\
1 & 2 & 3 & 4 & 5 & 6
\end{array}
\]

1. What is the probability of rolling a 1 and a 5 (in any order)?
2. Two 1s and a 5 (in any order)?
3. Five 1s and two 5s (in any order)?

\[
\begin{array}{cccccc}
\text{1/4} & \text{1/8} & \text{1/8} & \text{1/4} & \text{1/8} & \text{1/8} \\
1 & 2 & 3 & 4 & 5 & 6
\end{array}
\]

\[
\left(\frac{1}{4}\right)^2 \times \frac{1}{8} \times 2 = \frac{1}{16}
\]
Number of ways that can happen

\[
\frac{1}{6} \times \frac{1}{6} = \frac{1}{36}
\]

\[
\left(\frac{1}{4}\right)^5 \times \left(\frac{1}{8}\right)^3 \times 21 = \frac{21}{524,288} = 0.00004
\]

General formula?
Multinomial distribution: independent draws over $m$ possible categories

If we have frequency counts $x_1, x_2, \ldots, x_m$ over each of the categories, the probability is:

$$p(x_1, x_2, \ldots, x_m | \theta_1, \theta_2, \ldots, \theta_m) = \frac{n!}{x_1! \cdots x_m!} \prod_{j=1}^{m} \theta_j^{x_j}$$

What are $\theta_j$?

Are there any constraints on the values that they can take?

$$\theta_j \geq 0$$

$$\sum_{j=1}^{m} \theta_j = 1$$

Why the digression?

$$p(x_1, x_2, \ldots, x_m | \theta_1, \theta_2, \ldots, \theta_m) = \frac{n!}{x_1! \cdots x_m!} \prod_{j=1}^{m} \theta_j^{x_j}$$

Drawing words from a bag is the same as rolling a die!

number of sides = number of words in the vocabulary
Back to words...

Why the digression?

\[ p(x_1, x_2, \ldots, x_m | \theta_1, \theta_2, \ldots, \theta_m) = \frac{n!}{\prod_{j=1}^{m} x_j!} \prod_{j=1}^{m} \theta_j^{x_j} \]

\[ p(\text{features, label}) = p(y) \frac{n!}{\prod_{j=1}^{m} (\theta_j)^{y_j}} \]

θ_j for class y

Basic steps for probabilistic modeling

Model each class as a multinomial:

\[ p(\text{features, label}) = p(y) \frac{n!}{\prod_{j=1}^{m} (\theta_j)^{y_j}} \]

Step 2: figure out how to estimate the probabilities for the model

How do we train the model, i.e. estimate θ_j for each class?

A digression: rolling dice

What if I rolled 400 times and got the following number?

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>50</td>
<td>50</td>
<td>100</td>
<td>50</td>
<td>50</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1/4</th>
<th>1/8</th>
<th>1/8</th>
<th>1/4</th>
<th>1/8</th>
<th>1/8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>

Training a multinomial

<table>
<thead>
<tr>
<th>1/4</th>
<th>1/8</th>
<th>1/8</th>
<th>1/4</th>
<th>1/8</th>
<th>1/8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>
Training a multinomial

For each label, y:

- w1: 100 times
- w2: 50 times
- w3: 10 times
- w4: ...

θ_j = \frac{\text{count}(w_j, y)}{\sum_{k} \text{count}(w_k, y)}

\text{number of times word } w_j \text{ occurs in label } y \text{ docs}

\text{total number of words in label } y \text{ docs}

p(y=1) = \prod_{j=1}^{n} \theta_j^{x_j} \prod_{j} \theta_j^{x_j}

\prod_{j} \theta_j^{x_j} \text{ is a constant!}

Classifying with a multinomial

(10, 2, 6, 0, 0, 1, 0, 0, ...)

\text{pick largest}

Multinomial finalized

Training:
- Calculate p(label)
- For each label, calculate θ_s

\theta_j = \frac{\text{count}(w_j, y)}{\sum_{k} \text{count}(w_k, y)}

Classification:
- Get word counts
- For each label you had in training, calculate:

p(y) \prod \theta_j

and pick the largest
Multinomial vs. Bernoulli?

- Handles word frequency
- Given enough data, tends to perform better

Maximum likelihood estimation

- Intuitive
- Sets the probabilities so as to maximize the probability of the training data

**Problems?**
- Overfitting!
- Amount of data
  - Particularly problematic for rare events
- Is our training data representative
Basic steps for probabilistic modeling

Step 1: Pick a model

Step 2: Figure out how to estimate the probabilities for the model

Step 3 (optional): Deal with overfitting

Probabilistic models

Which model do we use, i.e., how do we calculate \( p(\text{feature, label}) \)?

How do we train the model, i.e., how do we estimate the probabilities for the model?

How do we deal with overfitting?

Unseen events

\[ \theta_j = \frac{\text{count}(w_j, y)}{\sum_{i \in \text{categories}} \text{count}(w_i, y)} \]

What will \( \theta_{\text{banana}} \) be for the negative class?

Unseen events

\[ p(\text{"I ate a bad banana", negative}) = ? \]
Unseen events

Training data
positive
banana: 2
negative
banana: 0

\[ p("I ate a bad banana", \text{negative}) = 0 \]
\[ p("\ldots banana \ldots", \text{negative}) = 0 \]

Add lambda smoothing

\[ \theta_j = \frac{\text{count}(w_j, y)}{\sum_{k=1}^{m} \text{count}(w_k, y)} \]

for each label, pretend like we've seen each feature/word occur in \(\lambda\) additional examples

Different than...

How is this problem different?

Different than...

\[ p("I ate a bad banana", \text{positive}) \]
\[ p("I ate a bad", \text{positive}) \]
\[ p("I ate a bad banana", \text{negative}) \]
\[ p("I ate a bad", \text{negative}) \]

Out of vocabulary. Many ways to solve... for our implementation, we'll just ignore them.