# Naïve Bayes in Numpy

CS114B Lab 1

Kenneth Lai

January 27, 2022

# Goals of PA1

- You already implemented Naïve Bayes in CS114A

# Goals of PA1

- You already implemented Naïve Bayes in CS114A
- Goals of PA1:

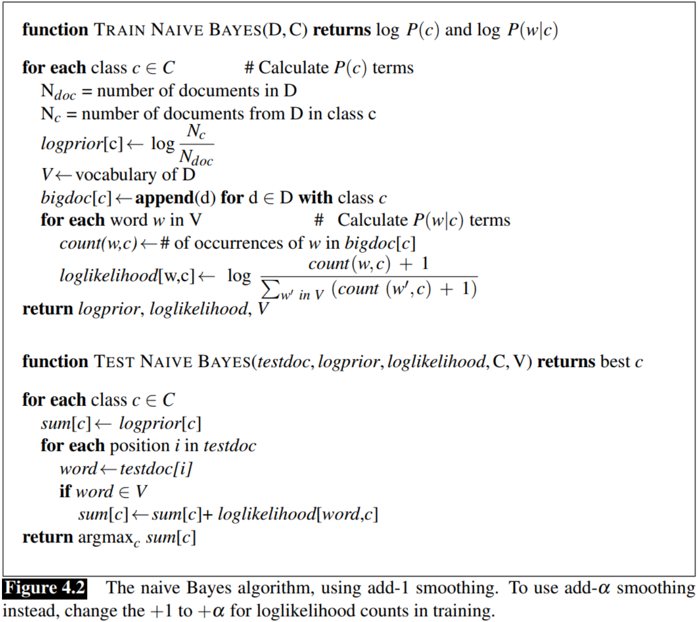
# Goals of PA1

- You already implemented Naïve Bayes in CS114A
- Goals of PA1:
  - Learn about Numpy arrays and operations

# Goals of PA1

- You already implemented Naïve Bayes in CS114A
- Goals of PA1:
  - Learn about Numpy arrays and operations
  - Learn how to represent documents/machine learning inputs in general as vectors

# Naïve Bayes



```
function TRAIN NAIVE BAYES(D, C) returns log $P(c)$ and log $P(w|c)$

for each class $c \in C$            # Calculate $P(c)$ terms
   $N_{doc}$ = number of documents in D
   $N_c$ = number of documents from D in class c
   $logprior[c] \leftarrow \log \frac{N_c}{N_{doc}}$
   $V \leftarrow$ vocabulary of D
   $bigdoc[c] \leftarrow$ append(d) for d $\in$ D with class $c$
   for each word $w$ in V                 #  Calculate $P(w|c)$ terms
      $count(w,c) \leftarrow$ # of occurrences of $w$ in $bigdoc[c]$
      $loglikelihood[w,c] \leftarrow \log \frac{count(w,c)\ +\ 1}{\sum_{w' \text{ in } V} (count\ (w',c)\ +\ 1)}$
return $logprior$, $loglikelihood$, $V$


function TEST NAIVE BAYES($testdoc$, $logprior$, $loglikelihood$, C, V) returns best $c$

for each class $c \in C$
   $sum[c] \leftarrow logprior[c]$
   for each position $i$ in $testdoc$
      $word \leftarrow testdoc[i]$
      if $word \in V$
         $sum[c] \leftarrow sum[c] + loglikelihood[word,c]$
return $\text{argmax}_c\ sum[c]$
```

**Figure 4.2** The naive Bayes algorithm, using add-1 smoothing. To use add-$\alpha$ smoothing instead, change the $+1$ to $+\alpha$ for loglikelihood counts in training.

# Training Naïve Bayes

- Follow the pseudo-code/what you did in CS114A

# Training Naïve Bayes

- Follow the pseudo-code/what you did in CS114A
    - By the end, the Numpy arrays `self.prior` and `self.likelihood` should be filled in

# Training Naïve Bayes

- Follow the pseudo-code/what you did in CS114A
  - By the end, the Numpy arrays `self.prior` and `self.likelihood` should be filled in
- Use dictionaries (`self.class_dict` and `self.feature_dict`) to translate between class/feature names and indices

# Training Naïve Bayes

- Follow the pseudo-code/what you did in CS114A
  - By the end, the Numpy arrays `self.prior` and `self.likelihood` should be filled in
- Use dictionaries (`self.class_dict` and `self.feature_dict`) to translate between class/feature names and indices
  - `self.class_dict` is given to you, while you will need to fill in `self.feature_dict` yourself

# Training Naïve Bayes

- Training data:

| document | class |
|---|---|
| just plain boring | negative |
| entirely predictable and lacks energy | negative |
| no surprises and very few laughs | negative |
| very powerful | positive |
| the most fun film of the summer | positive |

# Training Naïve Bayes

- `self.class_dict = {'neg': 0, 'pos': 1}`

# Training Naïve Bayes

- `self.class_dict = {‘neg’:  0, ‘pos’:  1}`
- self.feature_dict =

```
self.feature_dict = {‘predictable’:  0,
                     ‘no’:           1,
                     ‘fun’:          2,
                     ‘very’:         3
                     ⋮}
```

# Training Naïve Bayes

- `self.prior =` $\begin{bmatrix} \log(3/5) \\ \log(2/5) \end{bmatrix}$

# Training Naïve Bayes

- `self.prior =` $\begin{bmatrix} \log(3/5) \\ \log(2/5) \end{bmatrix}$
- `self.likelihood =`
  $\begin{bmatrix} \log(1/17) & \log(1/17) & \log(1/34) & \log(1/17) & \dots \\ \log(1/29) & \log(1/29) & \log(2/29) & \log(2/29) & \dots \end{bmatrix}$

# Training Naïve Bayes

- `self.prior` = $\begin{bmatrix} \log(3/5) \\ \log(2/5) \end{bmatrix}$
- `self.likelihood` =
  $\begin{bmatrix} \log(1/17) & \log(1/17) & \log(1/34) & \log(1/17) & \ldots \\ \log(1/29) & \log(1/29) & \log(2/29) & \log(2/29) & \ldots \end{bmatrix}$

# Testing Naïve Bayes

- Suppose we observe a movie review $d =$"very very fun". Is the review positive or negative?

# Testing Naïve Bayes

- $c_{NB} = \underset{c \in C}{\operatorname{argmax}} \log(P(c)) + \sum_{i=1}^{n} \log(P(w_i|c))$

# Testing Naïve Bayes

- $c_{NB} = \underset{c \in C}{\operatorname{argmax}} \log(P(c)) + \sum_{i=1}^{n} \log(P(w_i|c))$
- Compare:
  - $\log(P(-)) + \log(P(\text{very}|-)) + \log(P(\text{very}|-)) + \log(P(\text{fun}|-))$
  - $\log(P(+)) + \log(P(\text{very}|+)) + \log(P(\text{very}|+)) + \log(P(\text{fun}|+))$

# Testing Naïve Bayes

- $c_{NB} = \underset{c \in C}{\operatorname{argmax}} \log(P(c)) + \sum_{i=1}^{n} \log(P(w_i|c))$
- Compare:
    - $\log(3/5) + \log(1/17) + \log(1/17) + \log(1/34)$
    - $\log(2/5) + \log(2/29) + \log(2/29) + \log(2/29)$

# Testing Naïve Bayes

- $d =$ “very very fun”

# Testing Naïve Bayes

- $d =$ "very very fun"
- `self.feature_dict = {'predictable': 0,`

```
self.feature_dict = {'predictable':  0,
                     'no':           1,
                     'fun':          2,
                     'very':         3
                     ⋮}
```

# Testing Naïve Bayes

- $d =$ “very very fun”
- 
  ```
  self.feature_dict = {‘predictable’:  0,
                       ‘no’:           1,
                       ‘fun’:          2,
                       ‘very’:         3
                       ⋮}
  ```
- Create a feature vector

# Testing Naïve Bayes

- $d =$ “very very fun”
- 
  ```
  self.feature_dict = {‘predictable’:  0,
                       ‘no’:           1,
                       ‘fun’:          2,
                       ‘very’:         3
                       ⋮}
  ```
- Create a feature vector
  - Features are words, values are counts

# Testing Naïve Bayes

- $d =$ "very very fun"
- self.feature_dict =

  ```
  {'predictable':  0,
   'no':           1,
   'fun':          2,
   'very':         3
   ⋮}
  ```
- Create a feature vector
  - Features are words, values are counts
- vector = $\begin{bmatrix} 0 \\ 0 \\ 1 \\ 2 \\ \vdots \end{bmatrix}$

# Testing Naïve Bayes

- Take the matrix (`numpy.dot`) product of `self.likelihood` and our feature vector

# Testing Naïve Bayes

- Take the matrix (`numpy.dot`) product of `self.likelihood` and our feature vector
- $$\begin{bmatrix} \log(1/17) & \log(1/17) & \log(1/34) & \log(1/17) & \ldots \\ \log(1/29) & \log(1/29) & \log(2/29) & \log(2/29) & \ldots \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \\ 2 \\ \vdots \end{bmatrix}$$

# Testing Naïve Bayes

- Take the matrix (`numpy.dot`) product of `self.likelihood` and our feature vector
- $$\begin{bmatrix} \log(1/17) & \log(1/17) & \log(1/34) & \log(1/17) & \dots \\ \log(1/29) & \log(1/29) & \log(2/29) & \log(2/29) & \dots \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \\ 2 \\ \vdots \end{bmatrix}$$

$$= \begin{bmatrix} 0\log(1/17) + 0\log(1/17) + 1\log(1/34) + 2\log(1/17) \\ 0\log(1/29) + 0\log(1/29) + 1\log(2/29) + 2\log(2/29) \end{bmatrix}$$

# Testing Naïve Bayes

- Take the matrix (`numpy.dot`) product of `self.likelihood` and our feature vector
- $$\begin{bmatrix} \log(1/17) & \log(1/17) & \log(1/34) & \log(1/17) & \dots \\ \log(1/29) & \log(1/29) & \log(2/29) & \log(2/29) & \dots \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \\ 2 \\ \vdots \end{bmatrix}$$

  $$= \begin{bmatrix} \log(1/34) + 2\log(1/17) \\ 3\log(2/29) \end{bmatrix}$$

# Testing Naïve Bayes

- Take the matrix (`numpy.dot`) product of `self.likelihood` and our feature vector
- $$\begin{bmatrix} \log(1/17) & \log(1/17) & \log(1/34) & \log(1/17) & \dots \\ \log(1/29) & \log(1/29) & \log(2/29) & \log(2/29) & \dots \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \\ 2 \\ \vdots \end{bmatrix}$$

  $$= \begin{bmatrix} \log(1/34) + 2\log(1/17) \\ 3\log(2/29) \end{bmatrix}$$
- This computes the log-likelihood of the document given each class

# Testing Naïve Bayes

- Take the matrix (`numpy.dot`) product of `self.likelihood` and our feature vector
- $$\begin{bmatrix} \log(1/17) & \log(1/17) & \log(1/34) & \log(1/17) & \dots \\ \log(1/29) & \log(1/29) & \log(2/29) & \log(2/29) & \dots \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \\ 2 \\ \vdots \end{bmatrix}$$

  $$= \begin{bmatrix} \log(1/34) + 2\log(1/17) \\ 3\log(2/29) \end{bmatrix}$$
- This computes the log-likelihood of the document given each class
  - All we need is `self.prior`

# Testing Naïve Bayes

- $\begin{bmatrix} \log(3/5) \\ \log(2/5) \end{bmatrix} + \begin{bmatrix} \log(1/34) + 2\log(1/17) \\ 3\log(2/29) \end{bmatrix}$

  $= \begin{bmatrix} \log(3/5) + \log(1/34) + 2\log(1/17) \\ \log(2/5) + 3\log(2/29) \end{bmatrix}$

# Testing Naïve Bayes

- $\begin{bmatrix}\log(3/5)\\ \log(2/5)\end{bmatrix} + \begin{bmatrix}\log(1/34) + 2\log(1/17)\\ 3\log(2/29)\end{bmatrix}$

  $= \begin{bmatrix}\log(3/5) + \log(1/34) + 2\log(1/17)\\ \log(2/5) + 3\log(2/29)\end{bmatrix}$

- Take the argmax: positive