Naïve Bayes in Numpy

CS114B Lab 1

Kenneth Lai

January 20, 2023

Goals of HW1

- You already implemented Naïve Bayes in CS114A
- Goals of HW1:
  - 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-α smoothing instead, change the $+1$ to $+\alpha$ for loglikelihood counts in training.

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
  - You will need to fill these in 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}`
- `self.feature_dict = {'predictable': 0,`

```
                       'no':          1,
                       'fun':         2,
                       'very':        3,
                       ⋮}
```

Training Naïve Bayes

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

Training Naïve Bayes

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

Testing Naïve Bayes

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

Testing Naïve Bayes

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

- $c_{NB} = \operatorname*{argmax}_{c \in C} \log(P(c)) + \sum_{i=1}^{n} \log(P(w_i|c))$
- 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"
- `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
  - Features are words, values are counts
- vector = $\begin{bmatrix} 0 & 0 & 1 & 2 & \dots \end{bmatrix}$

Testing Naïve Bayes

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

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

Testing Naïve Bayes

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

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

Testing Naïve Bayes

- $\left[\log(1/34) + 2\log(1/17) \quad 3\log(2/29)\right] + \left[\log(3/5) \quad \log(2/5)\right]$
  $=$
  $\left[\log(1/34) + 2\log(1/17) + \log(3/5) \quad 3\log(2/29) + \log(2/5)\right]$
- Take the argmax: positive