COMP 6721 Applied Artificial Intelligence (Fall 2023)

Lab Exercise #4: Naïve Bayes Classification

Solutions

Question 1 Assume Jim is foraging for wild mushrooms and decides to use the mobile AI app *"MushDoom"* to help identify his findings. After scanning a mushroom, the app categorizes them as *Premium Grade*, *Standard Grade*, or *Caution Advised*. The app has the following accuracy rates based on its database and algorithms: 25% of the mushrooms it scans are labeled as *Premium Grade*, 50% are labeled as *Standard Grade*, and 25% are labeled as *Caution Advised*.

However, based on historical data and user feedback, mushrooms labeled as *Premium Grade* by the app have a 5% chance of being poisonous, a mushroom labeled as *Standard Grade* has a 15% chance of being poisonous, and a mushroom labeled as *Caution Advised* has a 25% chance.

If Jim consumes a mushroom identified by the app and falls ill from poisoning:

(a) What is the probability that the app had labeled the mushroom as *Premium Grade*?

Given:

$P(premium) = 0.25$
$P(standard) = 0.5$
$P(caution) = 0.25$
$P(poisonous|premium) = 0.05$
$P(poisonous|standard) = 0.15$
$P(poisonous|caution) = 0.25$

$$\begin{aligned} C_i &\in \{premium,\ standard,\ caution\} \\ P(poisonous) &= \sum P(poisonous|C_i) * P(C_i) \\ &= 0.05 * 0.25 + 0.15 * 0.5 + 0.25 * 0.25 \\ &= 0.15 \end{aligned}$$

$$\begin{aligned} P(premium|poisonous) &= \frac{P(poisonous|premium) * P(premium)}{P(poisonous)} \\ &= \frac{0.05 * 0.25}{0.15} \\ &= 0.083 \\ &= 8.3\% \end{aligned}$$

(b) What is the probability that the app had labeled the mushroom as *Standard Grade*?

$$\begin{aligned} P(standard|poisonous) &= \frac{P(poisonous|standard) * P(standard)}{P(poisonous)} \\ &= \frac{0.15 * 0.5}{0.15} \\ &= 0.5 \\ &= 50\% \end{aligned}$$

(c) What is the probability that the app had labeled the mushroom as *Caution Advised*?

$$\begin{aligned} P(caution|poisonous) &= \frac{P(poisonous|caution) * P(caution)}{P(poisonous)} \\ &= \frac{0.25 * 0.25}{0.15} \\ &= 0.417 \\ &= 41.67\% \end{aligned}$$

Question 2 Assume that Cecilia receives many e-mails from her home town in Klinga, where people speak Klinish. If you do not know Klinish, don't worry. It is a simple language made up of only 1,000 words that all start with the letter "k". A Klinish document may also contain words that do not start with "k", but these are considered out-of-vocabulary words (like a proper name, for example). Jack is trying to help Cecilia sort her Inbox into 3 mail folders (Personal, Work and Promotion). However, Jack does not speak Klinish, so all he has to work from are old e-mails that Cecilia has already sorted into the right folders. The table below shows a sample of the data that Jack has gathered from Cecilia's previous e-mails. The table indicates the frequency of each Klinish word in each folder (to be complete, the table should contain 1,000 rows, corresponding to each word in Klinish). For example, the word kiki appeared 30 times in e-mails labelled Personal, 50 times in e-mails about Work, and 9 times in Promotion e-mails.

		Folder		
		Personal	Work	Promotion
Word	kami	45	12	17
	kawa	78	1	67
	keke	0	5	80
	kiki	30	50	9
	koko	6	10	10
	kotuku	5	27	20
	koula	17	56	3
	...			
Total Nb of Words		20,000	25,000	17,000

The table above corresponds to data collected from 50 e-mails labeled *Personal*, 65 e-mails labeled *Work* and 45 e-mails labeled *Promotion*.

Based on the data above, Jack is trying to classify the following two e-mails (note that upper and lower cases should not be distinguished):

Email 1:	Koko kami kawa koula keke
Email 2:	Keke kawa, koko Google koula keke!

(a) Use a Naïve Bayes classifier without any smoothing, to classify the two e-mails above. Use the sum of logs (base 10), and show the score of each of the 3 classes (Personal, Work and Promotion) and the most likely class.

Priors:
P(Personal) = 50 / (50 + 65 + 45)
P(Work) = 65 / (50 + 65 + 45)
P(Promotion) = 45 / (50 + 65 + 45)

Email 1: Koko kami kawa koula keke

$score(personal) =$
$\log(P(personal)) + \log(P(koko|personal)) + \log(P(kami|personal)) +$
$\log(P(kawa|personal)) + \log(P(koula|personal)) + \log(P(keke|personal))$
$= \log(50/160) + \log(6/20{,}000) + \log(45/20{,}000) + \log(78/20{,}000) + \log(17/20{,}000) +$
$\log(0/20{,}000)$
$= -\infty$

$score(work) = \log(P(work)) + \log(P(koko|work)) + \log(P(kami|work)) +$
$\log(P(kawa|work)) + \log(P(koula|work)) + \log(P(keke|work))$
$= \log(65/160) + \log(10/25{,}000) + \log(12/25{,}000) + \log(1/25{,}000) + \log(56/25{,}000) +$
$\log(5/25{,}000)$
$= -17.8546$

$score(promotion) = \log(P(promotion)) + \log(P(koko|promotion)) +$
$\log(P(kami|promotion)) + \log(P(kawa|promotion)) + \log(P(koula|promotion)) +$
$\log(P(keke|promotion))$
$= \log(45/160) + \log(10/17{,}000) + \log(17/17{,}000) + \log(67/17{,}000) + \log(3/17{,}000) +$
$\log(80/17{,}000)$
$= -15.2664$

The highest score is -15.2664 $\implies$ the most likely class is promotion.

Email 2: Keke kawa, koko Google koula keke!

Note:

- ignore the word Google
- keke counts twice

$score(personal) = \log(P(personal)) + \log(P(keke|personal)) + \log(P(kawa|personal)) +$
$\log(P(koko|personal)) + \log(P(koula|personal)) + \log(P(keke|personal))$
$= \log(50/160) + \log(0/20{,}000) + \log(78/20{,}000) + \log(6/20{,}000) + \log(17/20{,}000) +$
$\log(0/20{,}000)$
$= -\infty$

$score(work) = \log(65/160) + \log(5/25{,}000) + \log(1/25{,}000) + \log(10/25{,}000) +$
$\log(56/25{,}000) + \log(5/25{,}000)$

$= -18.2348$

$score(promotion) = \log(45/160) + \log(80/17{,}000) + \log(67/17{,}000) + \log(10/17{,}000) + \log(3/17{,}000) + \log(80/17{,}000)$
$= -14.5938$

The highest score is -14.5938 $\implies$ the most likely class is promotion.

(b) Do the same as part A above, but this time use "add 0.5 smoothing" (i.e., instead of adding the value 1 to each word frequency, add $\frac{1}{2}$ to each word frequency). Adjust the smoothing formula accordingly, and show all your work. Again, use the sum of logs (base 10), and show the score of each of the 3 classes and the most likely class.

Solution:

		Folder		
		Personal	Work	Promotion
Word	kami	45.5	12.5	17.5
	kawa	78.5	1.5	67.5
	keke	0.5	5.5	80.5
	kiki	30.5	50.5	9.5
	koko	6.5	10.5	10.5
	kotuku	5.5	27.5	20.5
	koula	17.5	56.5	3.5
	...			
Total Nb of Words		20,000 +0.5 x 1,000 = 20,500	25,000 +0.5 x 1,000 = 25,500	17,000 +0.5 x 1,000 = 17,500

Email 1: Koko kami kawa koula keke

$score(personal) = \log(P(personal)) + \log(P(koko|personal)) + \log(P(kami|personal)) + \log(P(kawa|personal)) + \log(P(koula|personal)) + \log(P(keke|personal))$
$= \log(50/160) + \log(6.5/20,500) + \log(45.5/20,500) + \log(78.5/20,500) + \log(17.5/20,500) + \log(0.5/20,500)$
$= -16.7561$

$score(work) = \log(P(work)) + \log(P(koko|work)) + \log(P(kami|work)) + \log(P(kawa|work)) + \log(P(koula|work)) + \log(P(keke|work))$
$= \log(65/160) + \log(10.5/25,500) + \log(12.5/25,500) + \log(1.5/25,500) + \log(56.5/25,500) + \log(5.5/25,500)$
$= -17.6373$

$score(promotion) = \log(P(promotion)) + \log(P(koko|promotion)) + \log(P(kami|promotion)) + \log(P(kawa|promotion)) + \log(P(koula|promotion)) + \log(P(keke|promotion))$
$= \log(45/160) + \log(10.5/17,500) + \log(17.5/17,500) + \log(67.5/17,500) + \log(3.5/17,500) + \log(80.5/17,500)$
$= -15.2227$

The highest score is -15.2227 $\implies$ *the most likely class is promotion.*

		Folder		
		Personal	Work	Promotion
Word	kami	45.5	12.5	17.5
	kawa	78.5	1.5	67.5
	keke	0.5	5.5	80.5
	kiki	30.5	50.5	9.5
	koko	6.5	10.5	10.5
	kotuku	5.5	27.5	20.5
	koula	17.5	56.5	3.5
	...			
Total Nb of Words		20,500	25,500	17,500

Email 2: Keke kawa, koko Google koula keke!

Note:

- ignore the word Google
- keke counts twice

$score(Personal) = \log(P(personal)) + \log(P(keke|personal)) + \log(P(kawa|personal)) + \log(P(koko|personal)) + \log(P(koula|personal)) + \log(P(keke|personal))$
$= \log(50/160) + \log(0.5/20,500) + \log(78.5/20,500) + \log(6.5/20,500) + \log(17.5/20,500) + \log(0.5/20,500)$
$= -18.7152$

$score(work) = \log(65/160) + \log(5.5/25,500) + \log(1.5/25,500) + \log(10.5/25,500) + \log(56.5/25,500) + \log(5.5/25,500)$
$= -17.9939$

$score(promotion) = \log(45/160) + \log(80.5/17,500) + \log(67.5/17,500) + \log(10.5/17,500) + \log(3.5/17,500) + \log(80.5/17,500)$
$= -14.5599$

The highest score is -14.5599 $\implies$ *the most likely class is promotion*

Question 3 Let's write a *Python* program to train and run a model using the *Multinomial Naïve Bayes Classifier*. You can use the MNB implementation provided by *scikit-learn*, a popular machine learning library for Python.[1]

Let's first check if scikit-learn is already installed and if not, install it:

- Activate your desired Anaconda environment (use `comp6721` for the environment created in the first lab):

```
conda activate your_environment_name
```

- Check if scikit-learn is already installed:

```
python -c "import sklearn; print(sklearn.__version__)"
```

  If this command returns a version number, it means scikit-learn is already installed. If it raises an error, continue with the installation steps below.

- Install `scikit-learn`:

```
conda install scikit-learn
```

Here's how to get started with Naïve Bayes in scikit-learn:

```
import numpy as np
from sklearn.naive_bayes import MultinomialNB
from sklearn.feature_extraction.text import CountVectorizer
```

Start by implementing the *Email Spam Classifier* you've worked through on Worksheet #3. Create the training data:

```
corpus = np.array([
      'cheap meds for sale',
      'click here for the best meds',
      'book your trip',
      'cheap book sale, not meds',
      'here is the book for you'
])
```

To transform the text corpus into a feature vector, you can use scikit-learn's *CountVectorizer*:[2]

```
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(corpus)
```

You also need the target vector with the labels for the training data (here, SPAM is 0 and HAM is 1):

[1] See https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html

[2] See https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

```
y = np.array([0,0,0,1,1])
```

Get a classifier using the prior probabilities for each class (0.6 for SPAM, 0.4 for HAM):

```
classifier = MultinomialNB(class_prior=[0.6, 0.4])
```

Train a model using your classifier:

```
model = classifier.fit(X, y)
```

Now you can try to apply your model to classify a new email as SPAM or HAM. Here is the example email ('the cheap book') as a feature vector:

```
new_mail = [[0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0]]
```

(a) Create a complete, working Python program. Print out the intermediate variables to see the data you are working with. Predict the class for the `new_mail` using your `model` and print it out.

Here is a possible solution:

```
import numpy as np
from sklearn.naive_bayes import MultinomialNB
from sklearn.feature_extraction.text import CountVectorizer

corpus = np.array([
      'cheap meds for sale',
      'click here for the best meds',
      'book your trip',
      'cheap book sale, not meds',
      'here is the book for you'
])

# Create training data
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(corpus)
print(vectorizer.get_feature_names()) # Printing the vocabulary
print(X.toarray()) # Printing the token count vectors for the corpus

# Create target vector
y = np.array([0,0,0,1,1])
print("Target vector = ", y)

# Create multinomial naive Bayes classifier
# with prior probabilities of each class
classifier = MultinomialNB(class_prior=[0.6, 0.4])

# Train a model
```

```
model = classifier.fit(X, y)

# New email: 'the cheap book'
new_mail = [[0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0]]

# Predict new observation's class
predict = model.predict(new_mail)
print('Predicted class = ', predict)
```

Note that it prints the numerical values for the predicted class, if you want to use labels, you can add code like the following:

```
labels = ["SPAM", "HAM"]
print(f'Predicted class = {predict} ({labels[predict[0]]})')
```

(b) Inspect the scikit-learn documentation to understand how *smoothing* is handled for this algorithm.

First, remember that smoothing is introduced to handle the problem of zero probability. In the context of text classification, if a word is not present in the training data for a particular class, it will have a zero probability, which will affect the posterior probability. To avoid this, smoothing ensures that every word has a non-zero probability. As you can see from the scikit-learn documentation,

```
class sklearn.naive_bayes.MultinomialNB(alpha=1.0,
   fit_prior=True, class_prior=None)
```

the parameter `alpha` controls smoothing and uses 1.0 by default (this is what we also did in the lecture and is called *Laplace smoothing*[3]). Try experimenting with switching it off by setting the value to `0`.

(c) Change the code to transform the `new_mail` automatically from a string into a feature vector.

When using machine learning models, raw data (like a text email) usually cannot be processed directly. It has to be transformed into a numerical format that the model can understand. This is what the `CountVectorizer` does, turning text data into a vector of numbers:

```
T = vectorizer.transform(np.array(['the cheap book']))
```

A common mistake when using `CountVectorizer` (or any similar tool) is to fit a *new* vectorizer to the test data, which usually leads to inconsistent features between the training and test data. It's important to use the same `CountVectorizer` that you used for your training data. This ensures that the same word-to-index mapping is maintained. That's why we use `transform` instead of `fit_transform` (look up the difference in the documentation). If you were to fit a new vectorizer to the test data, the

[3] See https://en.wikipedia.org/wiki/Additive_smoothing

resulting vectors will not align with what the model expects, leading to errors or inaccurate predictions.

(d) Change the code to automatically compute the prior probabilities using the training data. Print out the priors for the model to verify that they are indeed correct.

If you do not explicitly provide the class priors, like we did above, they will be automatically computed from the training data. So unless you already know the priors for a specific problem, you would simply create the classifier with:

```
classifier = MultinomialNB()
```

You can verify them by printing out the `model` priors with:

```
print("Class priors ln = ", model.class_log_prior_)
```

which will show you the *natural logarithm* (ln) values of the computed class priors. The library provides the logarithm of the probabilities to avoid underflow issues that can occur when multiplying many small probabilities together, which is a common operation in Naïve Bayes. To get the actual probabilities from the logged values, you can use the exponential function. Here's how to convert the logged values back to regular probabilities:

```
print("Class priors = ", np.exp(model.class_log_prior_))
```