INFO371 Problem Set 6: Naïve Bayes

May 12, 2022

Introduction

This is perhaps the hardest problem set in this class. Sorry about that. And you will again work with spam emails. Hope you are not completely bored yet... The good news is that you already know this dataset somewhat. However, now you will implement your own brand new shiny Naive Bayes to categorize the emails into spam/non-spam, and find the optimal smoothing parameters.

Ling-Spam emails

This is the same dataset you used in PS5, *lingspam-emails.csv*. The corpus contains ~2700 emails from academic accounts talking about conferences, deadlines, papers etc, and peppered with wonderful offers of viagra, lottery millions and similar spam messages. The emails have been converted into a csv file that contains three variables:

spam true or false, this email is spam

files the original file name (not needed)

message the content of the email in a single line

Note that

1 (7pt) Load and clean data

1. (1pt) Load and clean data. Feel free to copy-paste from your PS05 solution.
2. (2pt) Vectorize emails so you have a DTM (I'll refer to this as the design matrix X) and the spam/non-spam indicator $\boldsymbol{y}$. If you don't know how to do it, you can just use the code below:

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(binary=True)
# define vectorizer
X = vectorizer.fit_transform(emails.message)
# vectorize your data.  Note: this creates a sparse matrix,
# use .toarray() if you run into trouble
vocabulary = vectorizer.get_feature_names()
# in case you want to see what are the actual words
```

   How many different documents (emails) and different tokens (words) do you have in these data?

3. (2pt) Split data into training/validation chunks.

4. (2pt) Design a scheme to name your variables so you can understand (and you grader can understand too!) which mathematical concept it refers to. You need variables like (see Lecture notes, Ch 7.3 for more examples/explanations):

   - $\Pr(S = 1)$: probability of spam
   - $\Pr(S = 0|W = 1)$: probability the email is not spam given it cointains the word W
   - $\log \Pr(W = 1)$: log probability of word present
   - $\ell(S = 1|\boldsymbol{W})$: log-likelihood of email being spam, given vector of words it contains.

   Explain how do you name these examples values.

---

2 (42pt) Naïve Bayes

Before you get into the business, here are two warming-up exercises:

1. (4pt) Here is a small excerpt from the initial DTM (before you split it into training/validation), corresponding to rows 983 to 985, and to columns 40,042–40,046:

```
X[982:985, 40041:40046].toarray()

## array([[0, 0, 0, 0, 0],
##        [0, 1, 0, 1, 0],
##        [0, 0, 0, 0, 0]])
```

   What do these numbers show:

   (a) which emails do the rows correspond to?
   (b) Which words do the columns correspond to?
   (c) What do the "1"-s in the middle of the table mean?
   (d) What do the zeros mean?

   Note: you should have exactly the same numbers in your analysis, this is not random.

2. (2pt) What is the accuracy of the naive model that predicts all emails into the majority category?

Now the data is in shape and it is time to get serious. But first things first. Ensure you are familiar with Naive Bayes. Consult the readings, available on canvas.

- Lecture notes Section 7.3 is written with this task in mind.
- Schutt & O'Neill is an easy and accessible (and long) introduction,
- Whitten & Frank is a lot shorter but still accessible introduction.

Good. Now you are ready with the preparatory work and it's time to dive into the real thing. Let's implement Naive Bayes. Use only training data in the fitting below. We also do it in a slightly simpler way and ignore the missing words when computing the probabilites, i.e. we only look for the effect of a word's presence ($W = 1$), not for the effect of its absence ($W = 0$).

3. (3pt) Compute the unconditional (log) probability that the email is spam/non-spam, $\log \Pr(S = 1)$, and $\log \Pr(S = 0)$. These probabilities are based on the values of $\mathtt{y}$ (i.e. *spam*) alone. They do not contain information about the words in emails.

4. (8pt) For each word w, compute the (log) probability that the word is present in spam emails, $\log \Pr(W = 1|S = 1)$, and (log) probability that the word is present in non-spam emails, $\log \Pr(W = 1|S = 0)$. These probabilities can easily be calculated from counts of how many times these words are present for each class.

   Hint: these computations are based on your BOW-s X. Look at ways to sum along columns in this matrix.

5. (1pt) What should be the dimension of your $\log \Pr(W = 1|S = 0)$ and $\log \Pr(W = 1|S = 1)$ vectors? Explain!

Now we are done with the estimator. Your fitted model is completely described by these four probability vectors: $\log \Pr(S = 1), \log \Pr(S = 1), \log \Pr(W = 1|S = 1), \log \Pr(W = 1|S = 0)$.

Let's now pull out your validation data and do some predictions.

6. (10pt) For both classes, $S = 1$ and $S = 0$, compute the *log-likelihood* that the email belongs to this class. Log-likelihood is given as (7.3.20 and 7.3.21, page 270 for now) in lecture notes, and the equations in Schutt "Doing Data Science", page 102.

   Computing the likelihoods involves sums of the previously computed probabilities, $\log \Pr(W = 1|S)$, and BOW elements x_{ij}. Start by doing this by whatever way you can get it done (e.g. loops). The most important thing is that you understand what you do!

   But if you want to write efficient code, use matrix product instead (it is $\sim 1000\times$ faster than loops). See Lecture Notes (7.30.30) for how to do it with matrix product. You can also check out `np.apply_along_axis` as an alternative way to do it.

7. (2pt) How many log-likelihoods you have to compute? Explain why do you have to have this many log-likelihoods.

8. (7pt) Based on the log-likelihoods, predict the class $S = 1$ or $S = 0$ for each email in the validation set.

9. (3pt) Print the resulting confusion matrix and accuracy (feel free to use existing libraries).

10. (5pt) If your results are like mine, you can see that the results are not impressive at all, your model works no better than the naive guess. Explain why do you get such mediocre results.

    Hint: this is related to infinites, where are those coming from, and why they make the model useless? See also the smoothing-related discussion in Lecture Notes at the end of the Naive Bayes (Section 7.3.3), before Example 7.3.

    Note: *just explain*, but do not do anything about it! We'll attack the problem in the next question with smoothing.

---

3 (32pt) Add smoothing

So, now you have your brand-new NB algorithm up and running. But the results are not impressive... As a next step, we add smoothing to the model.

1. (2pt) As you will be doing validation below, your first task is to mold what you did above into two functions: one for fitting and another one for predicting. You can mostly copy-paste your code from above.

2. (18pt) Add smoothing to the model. Smoothing amounts to assuming that we have "seen" every possible word $\alpha \geqslant 0$ times already, in both spam and non-spam emails. Note that α does not have to be an integer, and typically the best $\alpha < 1$.

   What you have to do is to re-compute the probabilities $\log \Pr(S = 1), \log \Pr(S = 0), \log \Pr(W = 1|S = 1), \log \Pr(W = 1|S = 1)$, the predictions part will remain unchanged. So you should update your fitting function by adding an additional argument α to it, and modify the probabilities accordingly. (And you use only training data for this.)

   See Lecture Notes 7.3.2; Example 7.3 (page 274); and Schutt p 103 and p 109 for more explanations.

3. (5pt) Use your updated model for predictions with a few different α-values (on validation data) and report the corresponding confusion matrix and accuracy.

   A well-implemented algorith should not spend more than a few seconds on both fitting and predicting. However, more important that you understand what you are doing!

4. (5pt) Use validation to find the best smoothing parameter α.

   You can just run a loop over different values, but start with very small values (10^{-8}, 10^{-7}, 10^{-6} up to perhaps 10).

   Note: this is fairly fast if your algorithm is fast. But even if your algorithm is slow, do the best you can!

   If your results are like mine, your best accuracy will be $> 99.5\%$. (But this result is random!)

5. (2pt) Plot how accuracy depends on α. Use log-scale for α!

---

4 (20pt) Interpretation

Naive Bayes is interpretable in a little similar fashion like linear regression. But in only a little similar fashion. Namely, we can find the words that are the best predictors that an email is spam, and the best predictors that email is non-spam. And we want to look at reasonably common words only, say more frequent than 10 times in the data.

1. (10pt) Which words are the best predictors that an email is spam? These are the word where $\Pr(S = 1|W = 1)$ is large and $\Pr(S = 0|W = 1)$ is small, or to put it differently, where $\log \Pr(S = 1|W = 1) - \log \Pr(S = 0|W = 1)$ is large.

   Explain why this is the case.

   Hint: you may re-check the concept of log-likelihood and how that is used for prediction.

   Hint 2: you may imagine you receive 60k 1-word emails (one for each word in your vocabulary). Which ones are most likely spam, and which ones are least likely spam?

2. (10pt) Find 10 best words to predict spam and 10 best words to predict non-spam. Comment your results.

How many hours did you spend on this PS?

5 Extra credit: implement cross-validation (10ec points)

Above we just did validation using testing/validation set approach. But cross-validation is better–it avoids much of the random noise when doing the split. Here your task is to implement CV yourself without using existing libraries!

- Implement k-fold CV. I recommend to implement it as a function that a) puts your data into random order; b) splits these into k chunks; c) selects one chunk for testing and the others for training; d) trains your NB model on the training chunks; e) computes accuracy on training chunk; f) returns mean accuracy over all these k trials. The function should also take α as an argument, this is the hyperparameter you are going to optimize.
- Find the optimal α by 5-fold CV using your own CV code. You have to find the cross-validated accuracies for a number of α-s between 0 and 1. Present the accuracy as a function of α on a plot and indicate which one is the best α.

  Hint: you may limit the vertical range of your plot, I find the vertical range $[0.95, 1]$ to make a nice plot.
