# Mp3: classify_email

discussion posted 6 days ago by **avonmoll**

Are we allowed to know how well the true solution performs on the local tests?

I'm getting:

```
You correctly classified 48 out of 49 spam emails, and
47 out of 51 ham emails.
```

locally but still failing the autograder test for `classify_emails`

Related to: Mini-project 3 / Mini-project: Email Spam Detection

This post is visible to everyone.

Add a Response

4 responses

RADUGROSU Community TA

6 days ago

Interesting! I have a much higher error rate and my solution passes.

> You correctly classified 40 out of 49 spam emails, and 48 out of 51 ham emails.

Because I'm passing all but the last test, I see two possible sources of error:

1. My default value for the log conditional probability (i.e. the log probability of a word that was NOT in the training set)

   I'm using $\log(\frac{1}{\text{total of all word counts for this class} + 2})$

   I tried all the different combinations I can think of for the denominator:

   - vocabulary size (+ 2)
   - # of messages for this class (+ 2)
2. My "vocabulary"

   I'm using the UNIQUE set of words from the union of words in the email and words in the `log_probabilities` for this class. Thus, each word that is either in the email or the probability dictionary is counted only once.

   If the word is in the email I just add the log_probability, otherwise, I add $\log(1 - e^{\text{log\_prob}})$

I can't seem to figure out what I'm doing wrong though.

posted 6 days ago by **avonmoll**

> I'm using the UNIQUE set of words from the union of words in the email and words in the log_probabilities for this class. Thus, each word that is either in the email or the probability dictionary is counted only once.

I'm not using unique words, I'm using every word in the email to be classified, as many times as it appears. (Edit: that's not correct)

> If the word is in the email I just add the log_probability, otherwise, I add $\log(1 - e^{\text{log\_prob}})$

For missing words I use $\log\left(\frac{1}{\#\text{number of emails in this class} + 2}\right)$ for non-missing words $\log\left(\frac{\text{number of emails of this class in which this word appeared} + 1}{\text{number of emails in this class} + 2}\right)$

posted 6 days ago by **RADUGROSU** Community TA

Are you handling words that are in the probability dictionary that AREN'T in the email?

Apologies, ~10 tests later I still can't reproduce the correct result.

--- Edit ---

I've now tried summing in the contributions to ALL words in the email (including duplicates, which seems incorrect) and summing (or not summing) in the contributions of words in the probability dictionary NOT in the email (i.e. the term $\log(1 - \hat{p}_j)^{1-y_j}$ from the lecture notes). Still no success.

--- Edit 2 ---

I've tried the above combinations with most of the combinations for the default (unseen word) probability. Still no success.

posted 6 days ago by **avonmoll**

> I've now tried summing in the contributions to ALL words in the email (including duplicates, which seems incorrect)

Why would that be incorrect? (Edit: again, I was wrong)

> and summing (or not summing) in the contributions of words in the probability dictionary NOT in the email

When computing the probability of an email, why would you be interested in anything else other than the words in the email? (Edit: because an email is not a list of words, it's a 0-1 vector as with the dimensionality of the vocabulary). Just add the log of the prior to the sum of the logs of the conditional probabilities of the words in the email, nothing more, nothing less. This is the model, for better or for worse; we'll critique it after it runs correctly.

posted 6 days ago by **RADUGROSU** Community TA

The model presented in the lectures takes into account the words not in the email, does it not? That's what the (1-p) terms are for.

posted 6 days ago by **dfannius**

To be more concrete: if you have determined that 99% of all spam emails have the word "banana" in them, and you are then asked to classify an email that does not contain the word "banana", does that information tell you anything useful?

posted 6 days ago by **dfannius**

You have a point. The length of the email (shorter emails would have higher probability this way) is a related critique. I'm clearly in error. Thank you!

posted 6 days ago by **RADUGROSU** Community TA

I've corrected my error - and now it passes the grader, but with the same number of false positives (3) and false negatives (9) as before.

@avonmoll, @dfannius Thank you both for your help.

posted 6 days ago by **RADUGROSU** Community TA

What I used was (and was okayed by the grader): 1) unique words from an email (python sets), and 2) when classifying if a word was not seen, say, in any of the ham emails I use a probability of 1/(#of ham emails in training set + 2). Similarly for words not seen in any of the training set spam emails 1/(#of spam emails in training set + 2).

posted 6 days ago by **Ash**

How long does it take for the test to run?

posted 5 days ago by **inigojpuente**

ash - how do you know the number of ham/spam emails in the training set from within the scope of the *classify_email* function?

posted 5 days ago by **inigojpuente**

I added a couple of globals and set them in the other function (can't remember the name). Feels a little hokey but the original idea made sense so I went with it. The classify_email function is not tested independently so the grader testing should be ok.

posted 5 days ago by **Ash**

I correctly classified 49 out of 49 spam emails, and 43 out of 51 ham emails and getting the autograder wrong.

I combined the word dictionaries for both ham and spam (edit: getting a total of ~528 based on first two training sets for each class) features and using 1/(number of total files + 2) for non-ham word for ham dict and non-spam word for spam dict. This method seems to be what it says in the lecture. Not sure what is wrong.

Which are the parameters that you all used to get the autograder correct?

posted 4 days ago by **Seph1201**

I have taken the similar approach in saying that we want to consider all the words from the training both spam and ham cases along with words from the e-mail of concern. However, in lieu of previous implementations, I loop through the union of these words separately for the case of spam and ham. For missing words, using 1/(num of case(spam/ham)+2), the algorithm correctly classified 45 out of 49 spam emails, and 34 out of 51 ham emails. Ofc, this misses the last question in the autograder, more specifically the second case. I am still trying to figure out if this is due to the difference in feature space or some mathematical error along the way. But from my previous experience with usual ML packages, we should definitely consider the entire feature space; however, there seems to be a different approach in mind for the staff of this course. I would really appreciate a bit more clarification on how to handle these.

posted 4 days ago by **JoonhoPark**

@Seph the number of features you are quoting seems very low. I am seeing 36874 words in spam dictionary and 13242 words in ham dictionary. Combined they give total of 42406 unique words in both dictionaries. How are you getting at ~528 words?

posted 3 days ago by **JoonhoPark**

@Ash, interesting. I used the same strategy as you based on the course video. When running on my computer, I got:

"correctly classified 43 out of 49 spam emails, and 46 out of 51 ham emails."

However, I failed the Auto-grader on the Test of classify_emails. I passed all the other 3 tests. Specifically, in theclassify_emails test, all match except for the 2nd item of the returned list. Still can't figure out why.

posted 3 days ago by **llmercury**

@JoonhoPark my bad i was referring to the first two cases of ham/spam which i used for testing my script... pl ignore the figure

posted 3 days ago by **Seph1201**

Add a comment

avonmoll

5 days ago

The reason I was not passing the final check in the autograder was actually an error in my `get_counts` function. Somehow, I was still passing the `get_counts` check even with this error, but it was causing a flip in some of the predicted classes later on -- totally wacky! I'm wondering if there is something up with the autograder, because I never should have gotten past that first check.

**Really, truly appreciate everyone's input on this** -- it helped me to debug and isolate some of the issues in my assumptions.

P.S.
Interestingly I still do not exactly match @RADUGROSU's results for the local test:

```
You correctly classified 38 out of 49 spam emails, and 48 out of
51 ham emails.
```

i.e. I had two additional false negatives

There autograder should use more extensive checks. It's incredible that my first (completely flawed implementation) passed.
Anyway, I'm curious what results other people have gotten. Who knows what bugs might still be lurking?

posted 5 days ago by **RADUGROSU** Community TA

@avonmoll, are you using unique set of words for `get_counts`? My `classify_emails` is wrong according to the grader if I'm using unique set of words for my`naivebayes.py`, but it gives me 100% if I'm counting according to the total counts of the word among all files.

posted 4 days ago by **chalupa**

@chalupa -- I was the exact opposite. Originally I was including duplicates in `get_counts` so I was computing the probabilities as $\frac{\#\text{ of occurrences in all files}}{\text{total }\#\text{ of words in all files}}$ which **does not** quite match the doc string definition, which says we should compute the fraction of documents a word occurs in. This approach was passing the first check but causing me to fail the final autograder test. One could argue that this approach still *sort of* approximates the fraction of files the word occurs in.

Once I switched my `get_counts` to only iterate through the unique set of words in each message, I passed.

posted 4 days ago by **avonmoll**

I am currently using unique words for computing get_counts and using the proposed laplace smoothing for get_log_probabilities. During the prediction phase, I am looping through the union of unique words from the e-mail and the keys from the corresponding class (spam/ham). During the loop iteration, I check to see if the word is in the e-mail then check if the word is in the training set, essentially resulting in 4 unique case scenarios. 1) word is in e-mail and in training, 2) word is in e-mail but not in training, 3) word is not in e-mail but in training, 4) word is not in e-mail and not in training. Admittedly case 4) seems unnecessary. I would like some critique on this method because locally it gives a decent result.

```
You correctly classified 42 out of 49 spam emails, and 46 out of 51 ham emails
```

But it fails the autograder for second email in the testing set. The basic mathematics is to add up the log probabilities for each case and lastly add the corresponding prior for each case, after-which, I take difference(ratio) to see if it is greater or equal to zero. Is my assumption that the denominators in both spam and ham cases are equal, correct? I would much appreciate some assistance in figuring out where I am going wrong.

posted 4 days ago by **JoonhoPark**

> During the loop iteration, I check to see if the word is in the e-mail then check if the word is in the training set, essentially resulting in 4 unique case scenarios

I use two scenarios: word is in the email or not. Each conditional distribution further gives the correct (smoothed) probability for missing/non-missing words.

> Is my assumption that the denominators in both spam and ham cases are equal, correct?

Yes, that would be the marginal probability of the email, so it's the same number.

There have been mentions here on the forum that flawed implementations of `get_counts` (for example) had passed the grader, so, unfortunately, no part should be considered correct just because it got green checks.

posted 3 days ago by **RADUGROSU** Community TA

---

I use one scenario: for every unique word in email I compute p_spam and p_ham and compare them at the end.

posted 3 days ago by **Mark_B2** Community TA

---

@RAD Thanks again for answering. I don't know if this is obvious from the lectures but I am having hard time figuring out the feature space of $Y_j^{(i)}$. For each e-mail we are testing, what does $J$ equal? In other words, are we only considering the unique features/words seen in that particular e-mail? or are we to consider all words in both dictionaries + e-mail or some combination of these? We keep talking about this $J$ throughout the lectures but I am not entirely sure if they represent the same number or it varies depending upon situation. Thanks in advance for help!

posted 3 days ago by **JoonhoPark**

This is my current understanding: the vocabulary is the set of *all* possible words. In practice this will shrink to the *set* of all the words ever seen (in ham, spam, or the current email), having, say, $J$ (unique) words. To compute the conditional probability of an email - that is of a binary vector of length $J$ - I loop over all the words in the vocabulary and consider the contribution of the slot $j$ in the email - i.e. of the presence/absence of that particular word.

posted 3 days ago by **RADUGROSU** Community TA

@RADUGROSU correct me if I misread you, but it seems you do methodological error. If during reading test set your model somehow changes (in other words you incorporate test distribution knowledge into model) it's no longer test set, but training one. You'll get overfitting.

posted 3 days ago by **Mark_B2** Community TA

No, I don't think it's an error. I don't incorporate probabilistic knowledge from the test set. I just acknowledge that the word "xxx", which wasn't seen before anywhere, is actually a word. The same results will be obtained in this case by simply ignoring brand-new words, but not so for training sets where spam and ham emails come in different numbers. In that case, a word that hasn't been seen before will come out to be less likely to be produced by the more numerous category - which in theory is sound, but in practice probably isn't.

posted 3 days ago by **RADUGROSU** Community TA

Then you should say: the vocabulary is the set of all unique spam and ham words and the reserved missing_word.

posted 3 days ago by **Mark_B2** Community TA

I'm not sure what you mean by "reserved".

posted 3 days ago by **RADUGROSU** Community TA

A placeholder.

posted 3 days ago by **Mark_B2** Community TA

@RAD & Mark Thank both for replies. So for a given testing e-mail, the "vocabulary" is a set union of keys from both spam and ham + unique words in that particular e-mail to account for all the possibilities. Is this correct? I am not entirely sure if I understand how the implementation of "placeholders" would work other than to serve as the special case where word in the e-mail is absent in either spam or ham dictionaries. Another possibilities is to have different loops for spam and ham and use a set union of unique e-mail words and keys from the corresponding dictionaries. Based on observations, number of keys in spams is more than double that of keys in hams; thus, leading to some bias if I were to loop through the a set union of all words instead of using the case specific, spam + e-mail and ham + e-mail. One last thing, I read earlier post by @Seph mentioning around 530 words in his vocabulary, are we supposed to clean up the words in count_words by removing punctuation and stemming words and such, as is the usual practice in NLP?

posted 3 days ago by **JoonhoPark**

Nevermind, I was making this harder than it had to be. I just never thought that the method leading to so many false spam classification could be the answer. We only work with features seen in training phase afterall. One thing that still bothers me is that the model is heavily biased toward spam. Here is the performance matrix I end up with: [[ 47. 2.] [ 18. 33.]] Obviously, the model favors spam classification a lot more than it does ham. Either way, thanks for all the help! You guys are the best!

posted 3 days ago by **JoonhoPark**

My participation in "vocabulary" discussion was to show that it inevitably leads to contradiction: either length of your binary vector $\boldsymbol{J}$ depends on test set or your $\boldsymbol{J_{missing}}$ is no longer binary.

posted 2 days ago by **Mark_B2** Community TA

Add a comment

avonmoll

4 days ago

A final word of mild frustration:

My code, which passed the autograder tests and seems correct and consistent with the instructions and assumptions of naive bayes, gave me incorrect results for part (c).

After the submission window closes, I'd really like to discuss this with somebody.

@avonmoll how did you increase and decrease $p(s)$? Did you take into account that the priors in classify_email are in log?

posted 3 days ago by **BradLee77**

It's still an increase in log. I threw in a print loop from 0.1 to 0.9 to override the learned s and observe the effect.

posted 3 days ago by **kiwitrader** Community TA

@kiwitrader what I meant was that log of a number between 0 and 1 is negative, and if simulating the increase/decrease by scaling, be careful of the sign.

posted 2 days ago by **BradLee77**

Right. In my log prior function I just returned `[log(0.999), log(0.001)]`, thus $s = 0.999$. However, when I printed the numerator for $p(s)$ I'm getting negative values in the hundreds in some cases. Thus the effect of the prior is washed out in all but a few cases considering $\log(0.001)$ is only -6.9.

In other words, because there are so many terms to take into account (i.e. the number of unique words between the email and dictionary) the prior is essentially inconsequential. As I was reading about this (sorry, didn't store the links), it turns out that many spam blockers that are based on naive bayes don't use ALL the words, only those with the most impact.

posted 2 days ago by **avonmoll**

Add a comment

kejriwall

3 days ago

I get this on local data

```
You correctly classified 38 out of 49 spam emails, and 48 out of 51 h
```

But not able to make the grader happy.

**I must say this is a sloppily designed assignment -- a disappointment after mp2**

You spam classification accuracy is too low.

posted 2 days ago by **Mark_B2** Community TA

I'm getting You correctly classified 45 out of 49 spam emails, and 46 out of 51 ham emails and still failing part 4 by one classification, sad day for me :P

```
Test: classify_emails(['autograder_spam1', 'autograder_spam2', 'autograder_spa
Your output:
spam    ham ham spam    ham
Correct output:
spam    spam    ham spam    ham
```

posted 2 days ago by **seanedXacc**

I don't know autograders threshold, my result of 43 out of 49 spam emails, and 48 out of 51 ham emails is accepted.

posted 2 days ago by **Mark_B2** Community TA

I passed the autograder with the same local results, @kejriwall.

I agree this assignment seemed a bit wonky.

posted 2 days ago by **avonmoll**

I had to accept the very biased model to pass the autograder. The second spam classification seems to be the tough one in the bunch. Have you tried to print out your p and q sums to see how close they are for the autograder test? If they are close perhaps just a little bias will do unless your prediction model differs significantly from the one proposed.

posted a day ago by **JoonhoPark**

That's exactly what I did. I had identical results to the ones at the start of this thread. The second autograder comparison was a ham instead of a spam. I added an offset to my decision in favour of a spam decision and it passed. I still get good results in the local test. I assume this is acceptable as it is actually discussed in the final part of the project

posted a day ago by **jmoranrun**

Well I passed doing what you said (by manually adjusting the bias), but it feels like an unsatisfactory answer lol.

posted about 15 hours ago by **seanedXacc**

Add a comment

Showing all responses

## Add a response:

Preview

Submit

© 2016 edX Inc. All rights reserved except where noted. EdX, Open edX and the edX and Open EdX logos are registered trademarks or trademarks of edX Inc.