Chapter 9: Applications to Computing

9.3: The Naive Bayes Classifier

Slides (Google Drive) Starter Code (GitHub)

9.3.1 Motivation

Have you ever wondered how Gmail knows whether or not an email should be marked as spam? Or how Alexa/Google Home can your answer free-form questions? How self-driving cars actually work? How social media platforms recommend friends and people you should follow? How a computer program DeepBlue beat the chess champion Garry Kasparov? The answer to all of these questions is: **machine learning (ML)**!

After learning just a tiny bit of probability, we are ready to discover one way to solve one extremely important type of ML task: **classification**. In particular, we'll learn how to take in an email (a string/text), and predict whether it is "Spam" or "Ham". We will discuss this further shortly!

9.3.2 The Machine Learning Framework

Suppose you are given the following four examples in the table below. Could you use the information from these four rows to predict the label of the last row?

Number	Shape	"Label"
3		12
5		15
-2		-8
7		21
-4		???

It's okay if you didn't see the pattern, but we should predict -16; can you figure out why? It seems that the pattern is to take the number and multiply it by the number of sides in the shape! So for our last row, we take -4 and multiply by 4 (the number of sides of the square) to get -16. Sure, there is a possibility that this isn't the right function: this is only the most simple explanation we could give. The function could be some complex polynomial in which case we would be completely wrong.

This is the idea of (supervised) **machine learning (ML)**: given some **training examples**, we want to *learn* the pattern between the input **features** and output **label** and be able to have a computer predict the label on new/unseen examples. Above, our input features were number and shape. We want the computer to "learn" just like how we do: with several examples.

Within supervised ML, two of the largest subcategories are **regression** and **classification**. Regression

refers to predicting a continuous (decimal) value. For example, when predicting house price given features of the house or predicting weight from height. Classification on the other hand refers to predicting one of a finite number of **classes**. For example, predicting whether an email is spam or ham, or whether an image of a handwritten digit is one of ten class: $0, 1, \ldots, 9$.

Example(s)

For each of the situations below with a desired output label, identify whether it would be a classification or regression task. Then, describe what input features may be useful in making a prediction.

1. Predicting the price of a house.
2. Predicting whether or not a PhD applicant will be admitted.
3. Predicting which of 50 menu items someone will order.

Solution

1. This is a regression task, since we are predicting a continuous number like $\$310,321.55$ or $\$1,235,998.23$. Some features which would be useful for prediction include: square footage, age, location, number of bedrooms/bathrooms, number of stories, etc.

2. This is a classification task, since we predicting one of two outcomes: admitted or not. Features which may be important are: GPA, SAT score, recommendation letter quality, number of papers published, number of internships, etc.

3. This is a classification task since we are choosing from one of 50 classes. Important features may include: past order history, favorite cuisine, dietary restrictions, income, etc.

□

9.3.3 The Naive Bayes Classifier

To summarize, our end goal is to write a function which takes in an email (a string type) and returns either that it is "SPAM" or "HAM". The function in Python may look something like this.

```
1 def classify(email:str):
2     # Some Code Here
3     if some_condition:
4         return SPAM
5     else:
6         return HAM
```

So how do we write the code to make the decision for us? In the past, people tried writing these classifiers with a set of rules that they came up themselves. For example, if it is over 1000 words, predict "SPAM". Or if it contains the word 'Viagra', predict that it is "SPAM". This leads to code which looks like a ton of if-else statements, and is also not very accurate. In machine learning, we come up with a model that learns a decision-making rule for us! This may not make sense now, but I promise it will soon.

9.3.3.1 Preprocessing the Emails

Handling text can be very messy. People misspell words, use slang that isn't in the vocabulary, have bad grammar, use tons of punctuation, and so on. When we process our emails, we will employ the following approach:

1. Ignore Duplicate Words.
2. Ignore Punctuation.
3. Ignore Casing.

That is, we will reduce an email into a *Set* of lowercase words and nothing else! We'll see a potential drawback to this later, but despite these strong assumptions, the classifier still does a really good job!

Here are some examples of how we take the input string (email) to a Set of standardized words.

Raw Email (string)	**Processed Email** (Set)
Hello hello hello there.	{hello, there}
You buy Viagra!!!!!	{you, buy, viagra}
Hello sir, I must ask that you keep this confidential...	{hello, sir, i, must, ask, that, you, keep, this, confidential}

9.3.3.2 The Decision Rule

For this section, we'll use the example of classifying the email "You buy Viagra!". The representation we have after processing is {you, buy, viagra}. Here's the approach of the Naive Bayes classifier. We will compute and compare the following two quantities (which must add to 1):

$$\mathbb{P}\left(\text{spam} \mid \{\text{you, buy, viagra}\}\right) \qquad \text{and} \qquad \mathbb{P}\left(\text{ham} \mid \{\text{you, buy, viagra}\}\right)$$

This is because, for a particular email, it is either spam or ham, and so the probabilities must sum to 1. In fact, because they both sum to 1, we can just compute one of them (let's say the first), and predict SPAM if $\mathbb{P}\left(\text{spam} \mid \{\text{you, buy, viagra}\}\right) > 0.5$ and HAM otherwise. Note that if it is exactly equal to 0.5, we will predict HAM (this is arbitrary - you can break ties however you want).

9.3.3.3 Learning from Data

WARNING: This is the heaviest math section, which draws from all ideas of Chapter 2.

Above all sounds nice and all, but how do we even begin to compute such a quantity? Let's try Bayes Theorem with the Law of Total Probability and see where that gets us!

$$\begin{aligned}
\mathbb{P}\left(\text{spam} \mid \{\text{you, buy, viagra}\}\right) &= \frac{\mathbb{P}\left(\{\text{you, buy, viagra}\} \mid \text{spam}\right)\mathbb{P}\left(\text{spam}\right)}{\mathbb{P}\left(\{\text{you, buy, viagra}\}\right)} && \text{[Bayes]} \\
&= \frac{\mathbb{P}\left(\{\text{you, buy, viagra}\} \mid \text{spam}\right)\mathbb{P}\left(\text{spam}\right)}{\mathbb{P}\left(\{\text{you, buy, viagra}\} \mid \text{spam}\right)\mathbb{P}\left(\text{spam}\right) + \mathbb{P}\left(\{\text{you, buy, viagra}\} \mid \text{ham}\right)\mathbb{P}\left(\text{ham}\right)} && \text{[LTP]}
\end{aligned}$$

How does this even help?? This looks way worse than before... Let's see if we can't start by figuring out the "easier" terms, like $\mathbb{P}(\text{spam})$. Remember we haven't even touched our data yet. Let's assume we were given five examples of emails with their labels to learn from:

Email	**Label**
Buy Viagra!	Spam
You good?	Ham
Viagra help you.	Spam
Good Viagra help.	Spam
I need Viagra for my health condition.	Ham

Based on the data only, what would you estimate $\mathbb{P}(\text{spam})$ to be? I might guess 3/5, and hope that you matched that! That is,

$$\mathbb{P}(\text{spam}) \approx \frac{\#\text{ of spam emails}}{\#\text{ of total emails}}$$

Similarly, we might estimate

$$\mathbb{P}(\text{ham}) \approx \frac{\#\text{ of ham emails}}{\#\text{ of total emails}}$$

to be 2/5 in our case. Great, so we've figured out two our of the four terms we needed after using Bayes/LTP. Now, we might try to similarly guess that

$$\mathbb{P}(\{\text{you, buy, viagra}\} \mid \text{spam}) \approx \frac{\#\text{ of spam emails with \{you, buy, viagra\}}}{\#\text{ of spam emails}}$$

because our definition of conditional probability came intuitively with equally likely outcomes in 2.1 as

$$\mathbb{P}(A \mid B) = \frac{|A \cap B|}{|B|} = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(B)}$$

But how many spam emails are we going to get that contain all three words? Probably none, or very few. In general, most emails will be much longer, so there's almost no chance that an email you are given to learn from has ALL of the words. This is a problem because it makes this probability 0, which isn't good for our model.

The Naive Bayes name comes from two parts. We've seen the Bayes part because we used Bayes Theorem to (attempt to) compute our desired probability. We are at a roadblock now, and now we will **make the "naive" assumption** that: words are conditionally independent GIVEN the label. That is,

$$\mathbb{P}(\{\text{you, buy, viagra}\} \mid \text{spam}) \approx \mathbb{P}(\text{you} \mid \text{spam})\,\mathbb{P}(\text{buy} \mid \text{spam})\,\mathbb{P}(\text{viagra} \mid \text{spam})$$

This should look like what we learned in 2.3:

$$\mathbb{P}(A, B, C \mid D) = \mathbb{P}(A \mid D)\,\mathbb{P}(B \mid D)\,\mathbb{P}(C \mid D)$$

So now, we might estimate

$$\mathbb{P}(\text{“you”} \mid \text{spam}) \approx \frac{\#\text{ of spam emails with “you”}}{\#\text{ of spam emails}}$$

which is most likely nonzero if we have a lot of emails! What should this quantity be? It is 1/3: there is just one spam email out of three which contains the word "you". In general,

$$\mathbb{P}(\text{word} \mid \text{spam}) \approx \frac{\#\text{ of spam emails with word}}{\#\text{ of spam emails}}$$

Now we're ready to put all of this together!

Example(s)

The emails are given again here for convenience:

Email	**Label**
Buy Viagra!	Spam
You good?	Ham
Viagra help you.	Spam
Good Viagra help.	Spam
I need Viagra for my health condition.	Ham

Make a prediction as to whether this email is SPAM or HAM, using the Naive Bayes classifier! Do this by computing $\mathbb{P}(\text{spam} \mid \{\text{you, buy, viagra}\})$ and comparing it to 0.5. Don't forget to use the conditional independence assumption!

Solution Combining what we had earlier (Bayes+LTP) with the (naive) conditional independence assumption, we get

$$\mathbb{P}(\text{spam} \mid \{\text{you, buy, viagra}\}) = \frac{\mathbb{P}(\{\text{you, buy, viagra}\} \mid \text{spam})\,\mathbb{P}(\text{spam})}{\mathbb{P}(\{\text{you, buy, viagra}\} \mid \text{spam})\,\mathbb{P}(\text{spam}) + \mathbb{P}(\{\text{you, buy, viagra}\} \mid \text{ham})\,\mathbb{P}(\text{ham})}$$

$$= \frac{\mathbb{P}(\text{you} \mid \text{spam})\,\mathbb{P}(\text{buy} \mid \text{spam})\,\mathbb{P}(\text{viagra} \mid \text{spam})\,\mathbb{P}(\text{spam})}{\mathbb{P}(\text{you} \mid \text{spam})\,\mathbb{P}(\text{buy} \mid \text{spam})\,\mathbb{P}(\text{viagra} \mid \text{spam})\,\mathbb{P}(\text{spam}) + \mathbb{P}(\text{you} \mid \text{ham})\,\mathbb{P}(\text{buy} \mid \text{ham})\,\mathbb{P}(\text{viagra} \mid \text{ham})\,\mathbb{P}(\text{ham})}$$

We need to compute a bunch of quantities, but notice the left side of the denominator is the same as the numerator, so we need to compute 8 quantities, 3 of which we did earlier! I'll just skip to the solution:

$$\begin{aligned} \mathbb{P}(\text{spam}) &= \frac{3}{5} & \mathbb{P}(\text{ham}) &= \frac{2}{5} \\ \mathbb{P}(\text{you} \mid \text{spam}) &= \frac{1}{3} & \mathbb{P}(\text{you} \mid \text{ham}) &= \frac{1}{2} \\ \mathbb{P}(\text{buy} \mid \text{spam}) &= \frac{1}{3} & \mathbb{P}(\text{buy} \mid \text{ham}) &= \frac{0}{2} \\ \mathbb{P}(\text{viagra} \mid \text{spam}) &= \frac{3}{3} & \mathbb{P}(\text{viagra} \mid \text{ham}) &= \frac{1}{2} \end{aligned}$$

Once we plug in all these quantities, we end up with a probability of 1, because the $\mathbb{P}(\text{buy} \mid \text{ham}) = 0$ killed the entire right side of the denominator! It turns out then we should predict spam because $\mathbb{P}(\text{spam} \mid \{\text{you, buy, viagra}\}) = 1 > 0.5$, and this is correct! We still don't ever want zeros though, so we'll see how we can fix that soon! □

Notice how the **data** (example emails) completely dictated our decision rule, along with Bayes Theorem and Conditional Independence. That is, we **learned** from our data, and used it to make conclusions on new data!

One last final thing, to avoid zeros, we will apply the following trick called "Laplace Smoothing". Before, we had said that

$$\mathbb{P}(\text{word} \mid \text{spam}) \approx \frac{\#\text{ of spam emails with word}}{\#\text{ of spam emails}}$$

We will now *pretend* we saw TWO additional spam emails: one which contained the word, and one which did not. This means instead that we have

$$\mathbb{P}(\text{word} \mid \text{spam}) \approx \frac{\#\text{ of spam emails with word} + 1}{\#\text{ of spam emails} + 2}$$

This will ensure that we don't get any zeros! For example, $\mathbb{P}(\text{buy} \mid \text{ham})$ was $\frac{0}{2}$ previously (none of the two ham emails contained the word "buy"), but now it is $\frac{0+1}{2+2} = \frac{1}{4}$.
We do not usually apply Laplace smoothing to the label probabilities $\mathbb{P}(\text{spam})$ and $\mathbb{P}(\text{ham})$ since these will never be zero anyway (and it wouldn't make much difference if we did).

Example(s)

Redo the example from earlier, but now apply Laplace smoothing to ensure no zero probabilities. Do not apply it to the label probabilities.

Solution Basically, we just take the same numbers from above and add 1 to the numerator and 2 to the denominator!

$$\mathbb{P}(\text{spam}) = \frac{3}{5} \qquad \mathbb{P}(\text{ham}) = \frac{2}{5}$$
$$\mathbb{P}(\text{you} \mid \text{spam}) = \frac{1+1}{3+2} = \frac{2}{5} \qquad \mathbb{P}(\text{you} \mid \text{ham}) = \frac{1+1}{2+2} = \frac{2}{4}$$
$$\mathbb{P}(\text{buy} \mid \text{spam}) = \frac{1+1}{3+2} = \frac{2}{5} \qquad \mathbb{P}(\text{buy} \mid \text{ham}) = \frac{0+1}{2+2} = \frac{1}{4}$$
$$\mathbb{P}(\text{viagra} \mid \text{spam}) = \frac{3+1}{3+2} = \frac{4}{5} \qquad \mathbb{P}(\text{viagra} \mid \text{ham}) = \frac{1+1}{2+2} = \frac{2}{4}$$

Plugging these in gives $\mathbb{P}(\text{spam} \mid \{\text{you, buy, viagra}\}) \approx 0.7544 > 0.5$, so our prediction is unchanged! But it is better to not have probabilities ever being exactly one or zero, so this solution is preferred! □

That's it for the main idea! We're almost there now, just some logistics.

9.3.3.4 Evaluating Performance

Let's say we are given 1000 emails for learning our spam filter using Naive Bayes. How should we measure performance? We could check the **accuracy**, which is exactly what you think it is:

$$\text{accuracy} = \frac{\#\text{ of emails classified correctly}}{\#\text{ of total emails}}$$

However, if we trained/learned from these 1000 emails, and measure the accuracy, surely it will be very good right? It's like taking a practice test and then using that as your actual test - of course you'll do well! What we care about is how well the spam filter works on NEW or UNSEEN emails. Emails that the spam filter

was not allowed to see/use when estimating those probabilities. This is fair and more realistic now right? You get practice exams, as many as you want, but you are only evaluated once on an exam you (hopefully) haven't seen before!

Where do we get these new/unseen emails? We actually take our initial 1000 emails and do a **train/test split** (usually around 80/20 split). That means, we will use 800 emails to estimate those quantities, and measure the accuracy on the remaining 200 emails. The 800 emails we learn from are collectively called the **training set**, and the 200 emails we test on are collectively called the **test set**.

This is good because we care how our classifier does on new examples, and so when doing machine learning, we ALWAYS split our data into separate training/testing sets!

Disclaimer: Accuracy is typically not a good measure of performance for classification. Look into F1-Score and AUROC instead if you are interested! Since this isn't a ML class, we will stick with plain accuracy for simplicity.

9.3.3.5 Summary

Here's a summary of everything we just learned:

Definition 9.3.1: Naive Bayes Algorithm for Spam Filtering

Suppose we are given a set of emails WITH their labels (of spam or ham). We split into a training set with around 80% of the data, and a test set with the remaining 20%.

Suppose we are given an email with wordset $\{w_1, \ldots, w_k\}$ and want to make a prediction. We compute using Bayes Theorem, the law of total probability, and our naive assumption that words are conditionally independent given their label to get:

$$\mathbb{P}(\text{spam} \mid \{w_1, \ldots, w_k\}) = \frac{\mathbb{P}(\text{spam}) \prod_{i=1}^{k} \mathbb{P}(w_i \mid \text{spam})}{\mathbb{P}(\text{spam}) \prod_{i=1}^{k} \mathbb{P}(w_i \mid \text{spam}) + \mathbb{P}(\text{ham}) \prod_{i=1}^{k} \mathbb{P}(w_i \mid \text{ham})}$$

We estimate the quantities using the TRAINING SET ONLY as follows:

$$\mathbb{P}(\text{spam}) \approx \frac{\#\text{ of TRAINING spam emails}}{\#\text{ of total TRAINING emails}}$$

$$\mathbb{P}(\text{ham}) \approx \frac{\#\text{ of TRAINING ham emails}}{\#\text{ of total TRAINING emails}}$$

$$\mathbb{P}(w_i \mid \text{spam}) \approx \frac{\#\text{ of TRAINING spam emails with } w_i + 1}{\#\text{ of TRAINING spam emails} + 2}$$

$$\mathbb{P}(w_i \mid \text{ham}) \approx \frac{\#\text{ of TRAINING ham emails with } w_i + 1}{\#\text{ of TRAINING ham emails} + 2}$$

If $\mathbb{P}(\text{spam} \mid \{w_1, \ldots, w_k\}) > 0.5$, predict that the email is SPAM, and otherwise, predict it is HAM.

To get a fair measure of performance, make predictions using the above procedure on all the TEST emails and return the overall test accuracy.

9.3.3.6 Underflow Prevention

Computers are great, but sometimes they cause us problems. When we compute something like

$$\prod_{i=1}^{k} \mathbb{P}(w_i \mid \text{spam})$$

we are multiplying a bunch of numbers between 0 and 1, and so we will get some very very small number (close to zero). When numbers get too large on a computer (exceeding 2^{63} or something), it is called **overflow**, and results in weird and wrong arithmetic. Our problem is appropriately named **underflow**, as we can't handle the precision.

This is the last thing we need to figure out (I promise). Remember that our two probabilities $\mathbb{P}(\text{spam} \mid \{w_1, \ldots, w_k\})$ and $\mathbb{P}(\text{ham} \mid \{w_1, \ldots, w_k\})$ summed to 1, so we only needed to compute one of them. Let's go back to computing both, and just comparing which is larger:

$$\mathbb{P}(\text{spam} \mid \{w_1, \ldots, w_k\}) = \frac{\mathbb{P}(\text{spam}) \prod_{i=1}^{k} \mathbb{P}(w_i \mid \text{spam})}{\mathbb{P}(\text{spam}) \prod_{i=1}^{k} \mathbb{P}(w_i \mid \text{spam}) + \mathbb{P}(\text{ham}) \prod_{i=1}^{k} \mathbb{P}(w_i \mid \text{ham})}$$

$$\mathbb{P}(\text{ham} \mid \{w_1, \ldots, w_k\}) = \frac{\mathbb{P}(\text{ham}) \prod_{i=1}^{k} \mathbb{P}(w_i \mid \text{ham})}{\mathbb{P}(\text{spam}) \prod_{i=1}^{k} \mathbb{P}(w_i \mid \text{spam}) + \mathbb{P}(\text{ham}) \prod_{i=1}^{k} \mathbb{P}(w_i \mid \text{ham})}$$

Notice the denominators are equal: they are both just $\mathbb{P}(\{w_1, \ldots, w_k\})$. So, $\mathbb{P}(\text{spam} \mid \{w_1, \ldots, w_k\}) > \mathbb{P}(\text{ham} \mid \{w_1, \ldots, w_k\})$ if and only the corresponding numerator is greater:

$$\mathbb{P}(\text{spam}) \prod_{i=1}^{k} \mathbb{P}(w_i \mid \text{spam}) > \mathbb{P}(\text{ham}) \prod_{i=1}^{k} \mathbb{P}(w_i \mid \text{ham})$$

Recall the log properties:

$$\log(xy) = \log(x) + \log(y)$$

and that both sides are simply a product of $k+1$ terms. We can take logs on both sides and this preserves order because log is a monotone increasing function (if $x > y$ then $\log(x) > \log(y)$):

$$\log(\mathbb{P}(\text{spam})) + \sum_{i=1}^{k} \log(\mathbb{P}(w_i \mid \text{spam})) > \log(\mathbb{P}(\text{ham})) + \sum_{i=1}^{k} \log(\mathbb{P}(w_i \mid \text{ham}))$$

And that's it, problem solved! If our initial quantity (after multiplying 50 word probabilities) was something like $\mathbb{P}(\text{spam} \mid \{w_1, \ldots, w_k\}) \approx 10^{-81}$, then $\log \mathbb{P}(\text{spam} \mid \{w_1, \ldots, w_k\}) \approx -186.51$. There is no chance of underflow anymore!

9.3.3.7 After Finishing Chapter 7

We actually used some concepts of estimation from Chapter 7 that we just took for granted, as that was the "natural" thing to do. But it turns out, they have rigorous justifications for doing so (e.g., for estimating the probability of spam as just the number of spam emails over the total). As well as Laplace smoothing!

After reading Chapter 7: do you see how MLE/MAP were used here? We used MLE to estimate $\mathbb{P}(\text{spam})$ and $\mathbb{P}(\text{ham})$. We also used MAP to estimate all the $\mathbb{P}(w_i \mid \text{spam})$ as well, with a $\text{Beta}(2,2)$ prior: pretending we saw 1 of each success and failure. Naive Bayes actually required us to estimate all these different Bernoulli parameters, and it's great to come back and see!