Module 18

By: Daniel Lindquist, Victoria Salido, and Kayla Torres

Naive Bayes Text Classification

Naive Bayes: "assume independence even when it's false" (i.e. that is why it is 'naive').

Component regression- is a perfect union of regression with principal component analysis. Provides a way of compression of D-dimensional vector to K-dimensional vector.

Idea of Naive Bayes

> Think about events A and B (which are data points) and H is some hypothesis (e.g. an email is spam) then the Naive Bayes approximation is the truth here.

$$\text{Truth: } P(H|A,B) = \frac{P(H) * P(A,B|H)}{P(A,B)}$$

$$\text{Naive Bayes Approximation: } P(A,B|H) = P(A|H)*P(B|H)$$

Recall bag of words:

Document j = {word 12, word 1001, word 10017}

- ↳ Just a bag of words
- ↳ This motivates the multinomial model for text.

Intuition this:

Assume we had a physical bag of words (think of pieces cut out of a magazine and each piece contains a different word). Reach into the bag and pull out a single word. Now, what is probability of this event?

Consider event A: "reach into bag and pull out word ('medal')"

If one copy of each word in the English language is in this bag:

Probability(A) = $\frac{1}{D}$

(In this case, you are no more likely to pick the word metal than the word wombat)

There is not a general way that documents are constructed, but we can use this model to gain a fair assumption. The "bag of words" model is good for classifying some documents.

*Remember: "All models are wrong but some models are useful."

But in our bag of words model we do not have this probability, instead we say that the probability of 'medal' :

P('medal') = W_{medal} = W_{101}

W being the 101 index of the vector represented

Bag of Words model is parameterized by a probability vector (like that of the Gaussian model):

W = ($W_1, W_2, W_3..., W_D$)

Where W_j = P(random word is jth word)

W_j = jth entry in vector

(Generalizes the 1/D case)

The individual probability vector will be associated with a corpus of documents. Intuition is that we'd have a separate probability vector for each corpora. DIfferent probability vectors are biasing towards themes, content, etc.

Suppose we have document with the following information:

Words {1, 2, 3} ← this is our data

Our data consists of 1st, 2nd, and 3rd word of the dictionary

In the generalized model we pull out the first word, then the second, and then the third. Think of this as being "the catch and release method". We can pull out the same word each time we reach into the bag.

What is the probability of our data under our bag of words model?

Key assumption: successive words are considered independent.

Basically this is asking, what is the probability of word 1, 2, 3, and how does it factorize if the three events are independent?

P(word1, word2, word3)=P(w_1) * P(w_2) * P(w_3)

This needs to be thought of as a catch and release method. Each word goes back into the bag and is independent of the next word chosen.

If the words are not assumed to be independent:

P(word1, word2, word3)= P(w_1)*P(w_2 / w_1) *P($w_3 | w_1, w_2$)

The difficulty here is that you're drastically cutting down the amount of data needed to assume probability.

One strength of Naive Bayes is that we can estimate these probabilities very effectively from a modest to a large size document. We can gain efficiency, even though we lose computational complexity.

Let's make this a little more generic:

Recall we have been using a vector of counts:

X_i = ($X_{i1}, X_{i2}.....X_{iD}$)

X_{ij} = number of words j in our document i

We go through element by element and look at how many of these words contribute to the overall document in our model.

W = our bag of words probability vector

W = (W_1 , W_2 ,..... W_D)

What is the conditional probability of X given W:

P(X|W) = W_1^{xi1} * W_2^{xi2} … W_D^{xiD}

.

How many times did we see the word, W_1 ? → its Xij .

At the end of this conditional probability we get a multinomial coefficient. It is a constant that effectively describes the number of different ways we could see Xij copies of each word.

What amount of probability does that amount of W_1 contribute to the whole document?

It is the number of copies of that word raised to the power of how many times that word appears in the document.

If we saw W1, 1 time = W_1^{Xi1}

Suppose we have 3 flips of a biased coin (W):

W = probability of heads

Let's assume we see 2 heads.

Assuming the coin flips are independent, what is the probability of the event:

P(2 heads) = P(H) * P(H) * P(T)
+P(H) * P(T) * P(H)
+P(T) * P(H) * P(H)

n the above example: P(H)= W and P(T)= (1-W)

We have to take three flips into account. We could've seen two heads then a tail or one head, one tail, then heads again.

We write this as: ($\frac{3}{2}$) * W^2 * $(1-W)^1$ ← all three sequences have the same probs.

binomial coef.

P(k heads in N flips) = N * W^k * $(1-W)^{(N-k)}$

Classification(how we would use this for classification):

Consider the situation with 2 bags of words: W^1 and W^2

Both bags have their own vector of probabilities.

We are told that a bunch of words are pulled from one of the two bags, which bag did the words come from?

Now we have a document from an unknown bag = X

2 bags:

P(X| W^1) = $W_1^{1^{X_1}}$ * $W_2^{1^{X_2}}$ …….. $W_D^{1^{X_D}}$

P(X| W^2) = $W_1^{2^{X_1}}$ * $W_2^{2^{X_2}}$ …….. $W_D^{2^{X_D}}$

Numerical Underflow:

The issue here is that these numbers tend to be extremely small. Each of these words probabilities are 1/100,000 roughly. When we take lots of these small numbers (to the power of how many times they occur) these numbers get extremely small quickly. Computer's don't deal with numbers this small easily - numbers get so small that a computer looks at it like 0. Numerical Overflow is the opposite, numbers are too large. Instead we deal with these numbers as such:

We have the number 10^{-10000} (computer can't handle this)

So instead we should work with logarithms of very small or very large numbers.

logP(X|W) = $X_1\log(W_1) + X_2\log(W_2) + \ldots + X_D\log(W_D)$

$=\sum X_j * \log(W_j)$