AIM 5011-1: Natural Language Processing (3 credits)
Assignment 1

Spring 26

adam.faulkner@yu.edu
Course Github site

Class hours: Tuesdays 5:30-7:30pm
Class Room: 215 Lexington Avenue LX312

---

1 Instructions

Complete the following *by hand* and show all of your work in a separate scratch sheet or in the question itself. Use a calculator for complex calculations such as square roots and sigmoid. Scan or take a picture of the completed sheet and upload it to Canvas. The first two problems are variants of problems presented in Jurafsky & Martin so refer to the relevant chapters for guidance. **Due: Tuesday, Feb 3rd**

2 Ngram-based Language Modeling

Refer to Jurafsky & Martin Chapter 3 section 3.1.

You are given a training set of 90 instances of the token *they*, and 1 each of the tokens *to, sieve, a, went, sea, in, did, jumblies, blue*, and *green* for a total of 100 tokens. Now we see the following 10 token test set: *they they went to sea in a sieve they did*. What is the unigram perplexity of the test set?

3 Naive Bayes

Refer to Jurafsky & Martin Appendix B

Assume the following likelihoods for each word being part of a positive or negative movie review, and equal prior probabilities for each class.

Word	Pos	Neg
I	0.09	0.16
always	0.07	0.06
like	0.29	0.06
foreign	0.04	0.15
films	0.08	0.11

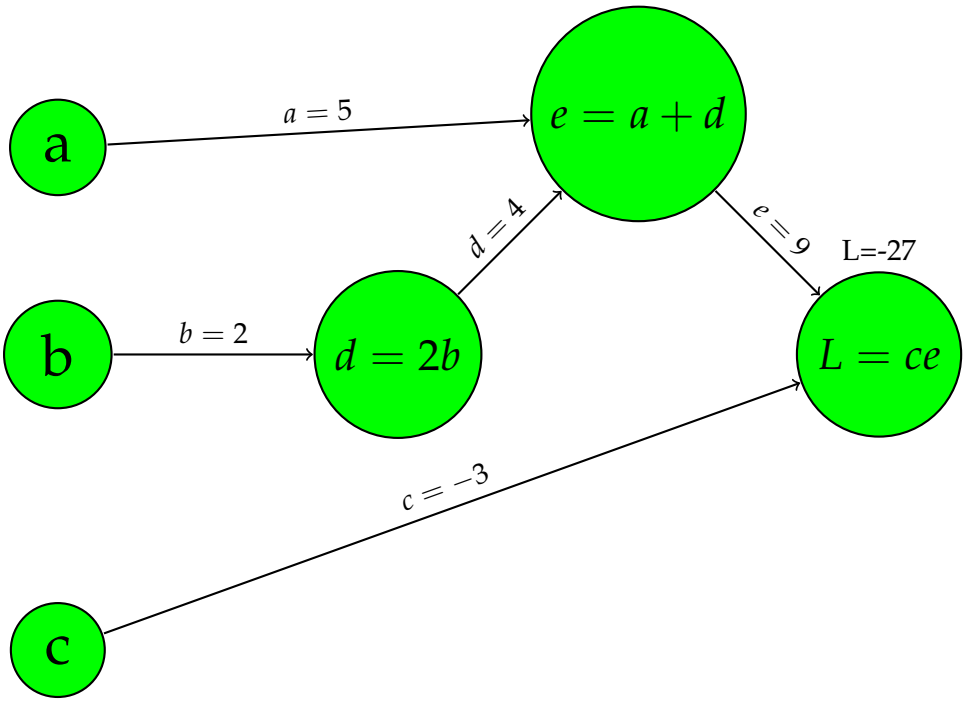
What class will Naive Bayes assign to the sentence "I always like foreign films."?

4 Backpropagation

For this problem, we'll be utilizing the computation graph formalism (see Jurafsky & Martin, Chapter 4).
Computation graphs can be used to visual mathematical expressions by breaking down the computation into separate operations, each of which is modeled as a node in a graph. For this exercise we've graphed the function $L(a, b, c) = c(a + 2b)$ by breaking the intermediate steps into:

- $d = 2 \times b$
- $e = a + d$
- $L = c \times e$

The graph for this is given below. We assume we're working with the inputs $a = 5$, $b = 2$, $c = -3$, and we have shown the result of the *forward* pass to calculate the result $L(5, 2, -3)$ = -27.



In this exercise, we're going to calculate the *backwards* pass of the above graph by computing the derivatives that we'll need for a weight update. The backwards pass computes the derivative of the output function L with respect to each of the input variables,i.e., $\frac{\sigma L}{\sigma a}$, $\frac{\sigma L}{\sigma b}$, and $\frac{\sigma L}{\sigma c}$. The results of these calculations tells us how much a small change in these input variables affects L.
In backward differentiation, we pass the gradients back from the final node to all nodes in the graph using the chain rule. Compute the backward pass gradients of the above graph by solving each of the following using the chain rule:

1. $\frac{\sigma L}{\sigma e} =$
2. $\frac{\sigma L}{\sigma c} =$
3. $\frac{\sigma e}{\sigma a} =$
4. $\frac{\sigma e}{\sigma d} =$
5. $\frac{\sigma L}{\sigma d} =$
6. $\frac{\sigma L}{\sigma a} =$
7. $\frac{\sigma d}{\sigma b} =$
8. $\frac{\sigma L}{\sigma c} =$
9. $\frac{\sigma L}{\sigma b} =$

5 Vector Semantics

As given in Jurafsky & Martin, Chapter 6, equation 6.10, the cosine similarity metric between two vectors $\mathbf{v}$ and $\mathbf{w}$ is computed as:

$$cosine(\mathbf{v}, \mathbf{w}) = \frac{\mathbf{v} \cdot \mathbf{w}}{|\mathbf{v}||\mathbf{w}|} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2} \sqrt{\sum_{i=1}^{N} w_i^2}}$$

Suppose we have 4 documents with the following counts for words *president*, *prime minister*, and *coconut*

Word	Doc1	Doc2	Doc3	Doc4
president	2	367	3424	1089
prime minister	7	788	415	231
coconut	545	7	6	23

Consider each row as an embedding for the word in the first column and calculate the following cosines:

1. cosine(president, prime minister) =

2. cosine(president, coconut) =

6 Term-weighting

Suppose we are given the following corpus (inspired by Dr Seuss's Green Eggs & Ham):

Doc #	Document
1	sam does not like green eggs and ham
2	jane hates green eggs and ham
3	does sam like green eggs and ham

These documents have the vocabulary *sam, does, not, like green, eggs, and, ham, Jane, hates*
Now calculate the Inverse-Document-Frequency (IDF) weight for each term in the vocabulary using Jurafksy & Martin, Chapter 11, eq. 11.3:

$$IDF_t = log_{10}\frac{N}{df_t}$$

where N is the total number of documents in the corpus, and df_t is the number of documents in which term t occurs.
Which term has the highest IDF weight? Interpret this result.