CS 188 Spring 2022 Introduction to Artificial Intelligence

# Written HW 9 Sol.

Solutions for HW 9 (Written)

# Q1. [40 pts] Probabilistic Language Modeling

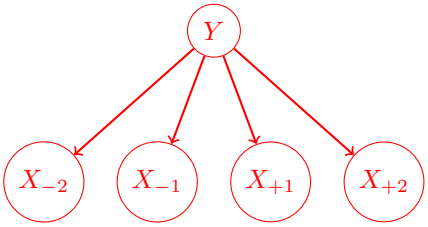
In lecture, you saw an example of supervised learning where we used Naive Bayes for a binary classification problem: to predict whether an email was ham or spam. To do so, we needed a labeled (i.e., ham or spam) dataset of emails. To avoid this requirement for labeled datasets, let's instead explore the area of unsupervised learning, where we don't need a labeled dataset. In this problem, let's consider the setting of language modeling.

Language modeling is a field of Natural Language Processing (NLP) that tries to model the probability of the next word, given the previous words. Here, instead of predicting a binary label of "yes" or "no," we instead need to predict a multiclass label, where the label is the word (from all possible words of the vocabulary) that is the correct word for the blank that we want to fill in.

One possible way to model this problem is with Naive Bayes. Recall that in Naive Bayes, the features $X_1, ..., X_m$ are assumed to be pairwise independent when given the label $Y$. For this problem, let $Y$ be the word we are trying to predict, and our features be $X_i$ for $i = -n, ..., -1, 1, ..., n$, where $X_i = ith$ word $i$ places from $Y$. (For example, $X_{-2}$ would be the word 2 places in front of $Y$. Again, recall that we assume each feature $X_i$ to be independent of each other, given the word $Y$. For example, in the sequence `Neural networks ____ a lot`, $X_{-2}$ = `Neural`, $X_{-1}$ = `networks`, $Y$ = the blank word (our label), $X_1$ = `a`, and $X_2$ = `lot`.

**(a)** First, let's examine the problem of language modeling with Naive Bayes.

**(i)** [1 pt] Draw the Bayes Net structure for the Naive Bayes formulation of modeling the middle word of a sequence given two preceding words and two succeeding words. You may think of the example sequence listed above:
`Neural networks ____ a lot.`




**(ii)** [1 pt] Write the joint probability $P(X_{-2}, X_{-1}, Y, X_1, X_2)$ in terms of the relevant Conditional Probability Tables (CPTs) that describe the Bayes Net.
$P(X_{-2}, X_{-1}, Y, X_1, X_2) = P(Y)P(X_{-2}|Y)P(X_{-1}|Y)P(X_1|Y)P(X_2|Y)$

**(iii)** [1 pt] What is the size of the largest CPT in the Bayes net? Assume a vocabulary size of $V$, so each variable can take on one of possible $V$ values.
Maximum CPT size is $V^2$.

**(iv)** [1 pt] Write an expression of what label $y$ that Naive Bayes would predict for $Y$ (Hint: Your answer should involve some kind of arg max and CPTs.)
$y = \arg\max_y P(X_{-2}, X_{-1}, y, X_1, X_2) = \arg\max_y P(y)P(X_{-2}|y)P(X_{-1}|y)P(X_1|y)P(X_2|y)$

**(v)** [3 pts] Describe 2 problems with the Naive Bayes Approach for the general problem of language modeling. Hint: do you see any problems with the assumptions that this approach makes?

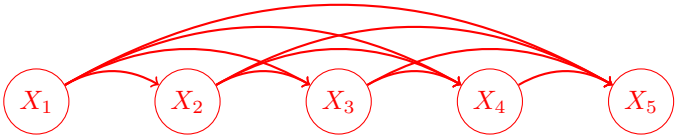
- Wrong independence assumptions: Naive Bayes assumes, by the structure of the Bayes Net, that each word within our 5 word window is independent of the others given the middle word. This is not a great assumption to make, because all the words in a neighborhood affect each other, even when given a few other words.
- Wrong to ignore the order of the words: Naive Bayes does not explicitly take into account the ordering of the words. For instance, if all the $P(X_i|Y)$ distributions are very similar, then the joint distribution $P(X_{-i}, X_{-(i+1)}, ..., Y, ..., X_{i-1}, X_i)$ is pretty agnostic to the orderings of the words. Phrases like "I hate 188 homework assignments" and "188 homework assignments hate I" would have similar probabilities, even though the later phrase intuitively should have a far less probability of occurring since it's grammatically incorrect.

Now, let's change our setting a bit. Instead of trying to fill in a blank given surrounding words, we are now only

given the preceding words. Say that we have a sequence of words: $X_1, ..., X_{m-1}, X_m$. We know $\{X_i\}_{i=0}^{m-1}$ but we don't know $X_m$.

**(b)** For this part, assume that every word is conditioned on all previous words. We will call this the **Sequence Model**.

**(i)** [1 pt] Draw the Bayes Net (of only $X_1, X_2, X_3, X_4, X_5$) for a 5-word sequence, where we want to predict the fifth word in a sequence $X_5$ given the previous 4 words $X_1, X_2, X_3, X_4$. Again, we are assuming here that each word depends on all previous words.




**(ii)** [1 pt] Write an expression for the joint distribution of a general sequence of length $m$: $P(X_1, ..., X_m)$.

Using chain rule, we get: $P(X_1)P(X_2|X_1)...P(X_m|X_1, ..., X_{m-1}) = \Pi_{i=1}^{m} P(X_i|\{X_j\}_{j=1}^{i-1})$

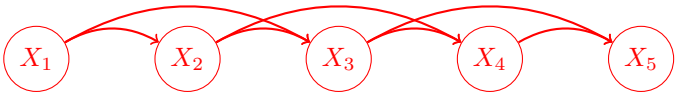
**(iii)** [1 pt] What is the size of the largest CPT in the Bayes net? Assume a vocabulary size of $V$, so each variable can take on one of possible $V$ values.

Maximum CPT table size is $V^m$

**(c)** You should have gotten a very large number for the previous part, which shows how infeasible the sequence model is. Instead of the model above, let's now examing another modeling option: N-grams. In N-gram language modeling, we add back some conditional assumptions to bound the size of the CPTs that we consider. We limit the tokens of consideration from "all previous words" to instead using only "the previous $N-1$ words." This creates the conditional assumption that, given the previous $N-1$ words, the current word is independent of any word before the previous $N-1$ words. For example, for $N = 3$, if we are trying to predict the 100th word, then given the previous $N - 1 = 2$ words (98th and 99th words), then the 100th word is independent of words $1, \ldots, 97$ of the sequence.

**(i)** [1 pt] Making these additional conditional independence assumption changes our Bayes Net. Redraw the Bayes Net from part (ci) to represent this new N-gram modeling of our 5-word sequence: $X_1, X_2, X_3, X_4, X_5$. Use $N = 3$.

$N = 3$ means that each token only depends on the previous $N - 1 = 2$ tokens.




**(ii)** [2 pts] Write an expression for the N-gram representation of the joint distribution of a general sequence of length $m$: $P(X_1, ..., X_m) = P(X_{1:m})$. Your answer should express the joint distribution $P(X_{1:m})$ in terms of $m$ and $N$.

Hint: If you find it helpful, try it for the 5 word graph above first before going to a general $m$ length sequence.

First, we start with the 5 word graph above. Joining the CPTs given by the Bayes net structure, we have

$$P(X_{1:5}) = P(X_1)P(X_2|X_1)P(X_3|X_1, X_2)P(X_4|X_2, X_3)P(X_5|X_3, X_4).$$

Generalizing this pattern yields

$$P(X_{1:m}) = P(X_1)\Pi_{i=2}^{m} P(X_i|X_{\max(i-(N-1),1):i-1}).$$

**(iii)** [1 pt] What is the size of the largest CPT in the Bayes net above? Again, assume a vocabulary size of $V$, and $m > N$.

$V^N$ is the largest table size.

**(iv)** [2 pts] Describe one disadvantage of using N-gram over Naive Bayes.

- The sum of all CPT sizes needed to calculate a joint distribution for N-gram is much larger than in Naive Bayes.

**(v)** [4 pts] Describe an advantage and disadvantage of using N-gram over the Sequence Model above.

Advantage: The CPT sizes for the N-gram joint distribution are much smaller than the Sequence Model.

Disadvantage: It's likely that we are making too many independence assumptions by ignoring all words before the preceding $N-1$ words.

**(d)** In this question, we see a real-world application of smoothing in the context of language modeling.

Say we have the following training corpus from Ted Geisel:

`i am sam . sam i am . i do not like green eggs and ham .`

Consider the counts given in the tables below, as calculated from the sentence above.

| 1-gram | |
|---|---|
| Token | Count |
| i | 3 |
| am | 2 |
| sam | 2 |
| . | 3 |
| do | 1 |
| not | 1 |
| like | 1 |
| green | 1 |
| eggs | 1 |
| and | 1 |
| ham | 1 |
| TOTAL | 17 |

| 2-gram | phrases | starting with i |
|---|---|---|
| Token1 | Token2 | Count |
| i | am | 2 |
| i | do | 1 |
| TOTAL | | 3 |

| 2-gram | phrases | starting with am |
|---|---|---|
| Token1 | Token2 | Count |
| am | sam | 1 |
| am | . | 1 |
| TOTAL | | 2 |

**(i)** [1 pt] Based on the above dataset and counts, what is the $N$-gram estimate for $N = 1$, for the sequence of 3 tokens `i am ham`? In other words, what is $P(i, am, ham)$ for $N = 1$?

$$\begin{aligned} P(i, am, ham) &= P(i)P(am)P(ham) \\ &= \frac{3}{17} \cdot \frac{2}{17} \cdot \frac{1}{17} \end{aligned}$$

**(ii)** [1 pt] Based on the above dataset and counts, what is the $N$-gram estimate for $N = 2$, for the sequence of 3 tokens `i am ham`? In other words, what is $P(i, am, ham)$ for $N = 2$?

$$\begin{aligned} P(i, am, ham) &= P(i)P(am|i)P(ham|am) \\ &= \frac{3}{17} \cdot \frac{2}{3} \cdot \frac{0}{2} \\ &= 0 \end{aligned}$$

**(iii)** [5 pts] Perform Laplace $k$-smoothing on the above problem and re-compute $P(i, am, ham)$ with the smoothed distribution, for $N = 2$. In order to calculate this, complete the pseudocount column for each entry in the probability tables. Note we add a new `<unk>` entry, which represents any token not in the table.

Hint: the count for the new `<unk>` row in each table would be 0.

| 1-gram | | |
|---|---|---|
| Token | Count | Pseudocount |
| i | 3 | 3 + k |
| am | 2 | 2 + k |
| sam | 2 | 2 + k |
| . | 3 | 3 + k |
| do | 1 | 1 + k |
| not | 1 | 1 + k |
| like | 1 | 1 + k |
| green | 1 | 1 + k |
| eggs | 1 | 1 + k |
| and | 1 | 1 + k |
| ham | 1 | 1 + k |
| <unk> | 0 | k |
| TOTAL | 17 | 17 + 12k |

| 2-gram phrases starting with "i" | | | |
|---|---|---|---|
| Token1 | Token2 | Count | Pseudocount |
| i | am | 2 | 2 + k |
| i | do | 1 | 1 + k |
| i | <unk> | 0 | k |
| TOTAL | | 3 | 3 + 3k |

| 2-gram phrases starting with "am" | | | |
|---|---|---|---|
| Token1 | Token2 | Count | Pseudocount |
| am | sam | 1 | 1 + k |
| am | . | 1 | 1 + k |
| am | <unk> | 0 | k |
| TOTAL | | 2 | 2 + 3k |

$$\begin{aligned} P(i, am, ham) &= P(i)P(am|i)P(ham|i, am) \\ &= P(i)P(am|i)P(ham|am) \\ &= P(i)P(am|i)P(unk|am) \\ &= \frac{3+k}{17+12k} \cdot \frac{2+k}{3+3k} \cdot \frac{k}{2+3k} \end{aligned}$$

**(iv)** [3 pts] What is the importance of smoothing in the context of language modeling?

Hint: see your answer for the previous subquestion.

In language modeling, as the above example shows, it is often possible for one of the probabilities to equal 0, making the entire sequence's probability equal to 0. However, the phrase "I am ham" is plausible because all three of the words appear in the corpus. So we use smoothing to make sure that plausible phrases (in fact, all phrases) don't have a 0 probability.

We don't want any sequence to be assigned a 0 probability because then we have trouble doing an argmax over probabilities that are all equal to 0 to choose the most likely next token. In other words, we are fine with very low probabilities, but not 0 probability, for unlikely events.

**(v)** [4 pts] What is a potential problem with Laplace smoothing? Propose a solution. (Assume that you have chosen the best $k$, so finding the best $k$ is not a problem.)

Hint: Consider the effect of smoothing on a small CPT.

For a small CPT with few entries, the addition of an *unk* entry could drastically change the empirical count-based probabilities such that they no longer become realistic.

A potential solution is to use a different type of smoothing that does not blindly assign $k$ additional counts to each category.

We could also free ourselves from the restriction that $k$ must be an integer and assign a decimal $k$ that is proportional to the total counts of the CPT.

**(vi)** [2 pts] Let the likelihood $\mathcal{L}(k) = P(i, am, sam)$. Give an expression for the log likelihood, $\ln \mathcal{L}(k)$, of this sequence after $k$-smoothing. Continue to assume $N = 2$.

$$\begin{aligned}\mathcal{L}(k) &= \Pi_i P(data_i) \\ &= P(i)P(am|i)P(sam|i, am) \\ &= P(i)P(am|i)P(sam|am) \\ &= \frac{3+k}{17+12k} \cdot \frac{2+k}{3+3k} \cdot \frac{1+k}{2+3k}\end{aligned}$$

So the log likelihood is $\ln \mathcal{L}(k) = \ln \frac{3+k}{17+12k} + \ln \frac{2+k}{3+3k} + \ln \frac{1+k}{2+3k}$

**(vii)** [4 pts] Describe a procedure we could do to find a reasonable value of $k$. No mathematical computations needed.

Hint: you might want to maximize the log likelihood $\ln \mathcal{L}(k)$ on something.

We could use a validation set to choose $k$. For example, we could first train on 90% of the training data. Then, on the remaining 10% of data, we could maximize the log likelihood $\ln \mathcal{L}(k)$ with respect to $k$. We could do this by solving $\frac{\partial \mathcal{L}(k)}{\partial k} = 0$.