CS 188 Summer 2021 Introduction to Artificial Intelligence — Written HW 4

Due: Wednesday 08/04/2021 at 11:59pm (submit via Gradescope).

Policy: Can be solved in groups (acknowledge collaborators) but must be written up individually

Submission: Your submission need not follow this template exactly, but you must tag where each question begins in your writeup when submitting this HW on Gradescope.

First name	
Last name	
SID	
Collaborators	

For staff use only:

Q1.	Probabilistic Language Modeling	/40
Q2.	Machine Learning	/40
	Total	/80

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.

(iii) [1 pt] What is the size of the largest CPT involved in calculating the joint probability? Assume a vocabulary size of V, so each variable can take on one of possible V values.

(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.)

(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?

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: $P(X_1, ..., X_m)$.

(iii) [1 pt] What is the size of the largest CPT involved in calculating the joint probability? Assume a vocabulary size of V, so each variable can take on one of possible V values.

(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 $N = 3$.

(ii) [2 pts] Write an expression for the N-gram representation of the joint distribution of a general sequence of length m: $P(X_1, ..., X_m)$. Please use set notation (for example: For tokens $X_i, ..., X_j$, please write something of the form $\{X_k\}_{k=i}^{j}$). Your answer should express the joint distribution $P(\{X_i\}_{i=1}^{m})$, 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.

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

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

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

(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 $N = 1$?

(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 $N = 2$?

(iii) [3 pts] What is the importance of smoothing in the context of language modeling?
Hint: see your answer for the previous subquestion.

(iv) [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	
am	2	
sam	2	
.	3	
do	1	
not	1	
like	1	
green	1	
eggs	1	
and	1	
ham	1	
<unk>	0	
TOTAL	17	

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

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

(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.

(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 $N = 2$.

(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.

Q2. [40 pts] Machine Learning

In this question, we will attempt to develop more intuition about how Neural Networks work. In parts A and B, we will discuss gradient descent, and in part C we look at backprop.

(a) Gradient descent is a procedure which allows you to minimize any loss function. As as example, let's consider a simple function $Loss(w) = w^2$ and let's assume that we want to minimize this function. Perform gradient descent on this loss function by using the update rule $w \leftarrow w - \alpha \frac{dLoss}{dw}$, where α is the learning rate.

(i) [1 pt] What is $\frac{dLoss}{dw}$? Write your answer in terms of w.

(ii) [1 pt] What is the optimal w that minimizes this loss function? We denote this value of w as w^*.

(iii) [2 pts] Carry out one iteration of gradient descent (i.e., weight update). What is the resulting weight (and corresponding post-update loss) for the scenarios below? Plot the loss function (w^2) by hand and, for each of the two scenarios below, draw the direction in which w is updated (an arrow on the w axis from w_{old} to w_{new}).

1. $\alpha = 0.1$, $w = 3$

2. $\alpha = 1$, $w = -2$

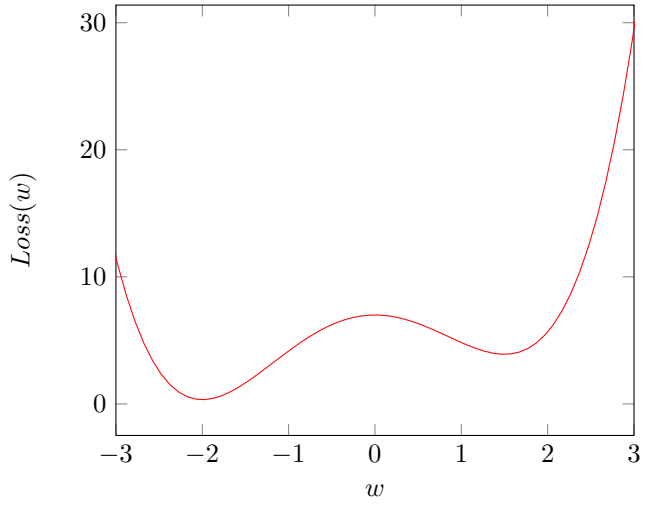
(iv) [4 pts] Assume w is initialized to some nonzero value. Assume we are still working with $Loss(w) = w^2$

1. Which value of α allows gradient descent to make w converge to w^* in the least amount of steps?

2. For what range of α does w never converge?
   Hint: You may consider for which γ where $w_t = \gamma w_{t-1}$ will never converge.

(v) [2 pts] Why must α always be positive when performing gradient descent?

(b) It is unlikely that we have a loss function as nice as w^2. Say we instead want to minimize some more complex loss $Loss(w) = \frac{w^4}{2} + \frac{w^3}{3} - 3w^2 + 7$, a polynomial with local minima at $w = -2, 1.5$, a global minimum at $w = -2$, a local maximum at $w = 0$, and limits that go to infinity for both $w \to \infty$ and $w \to -\infty$.



(i) [3 pts] Why do Neural Networks use gradient descent for optimization instead of just taking the derivative and setting it equal to 0? You may use the example error function from above to explain your reasoning.

(ii) [1 pt] What is the optimal w^*, given the loss above?

(iii) [3 pts] Let α and w take on the values below. For each case, perform some update steps and report whether or not gradient descent will converge to the optimum w^* after an infinite number of steps. If not, report whether it converges to some other value, or does not converge to any value.

1. $\alpha = 1$, $w = 0$

2. $\alpha = 1$, $w = -2$

3. $\alpha = 1$, $w = 1$

4. $\alpha = 0.1, w = 3$

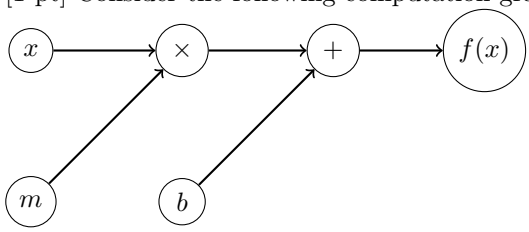
5. $\alpha = 0.1, w = 2$

6. $\alpha = 0.1, w = -10$

(iv) [3 pts] From the subquestion above, explain the effect of learning rate being (a) too high and (b) too low.

(c) Let's now look at some basic neural networks drawn as computation graphs.

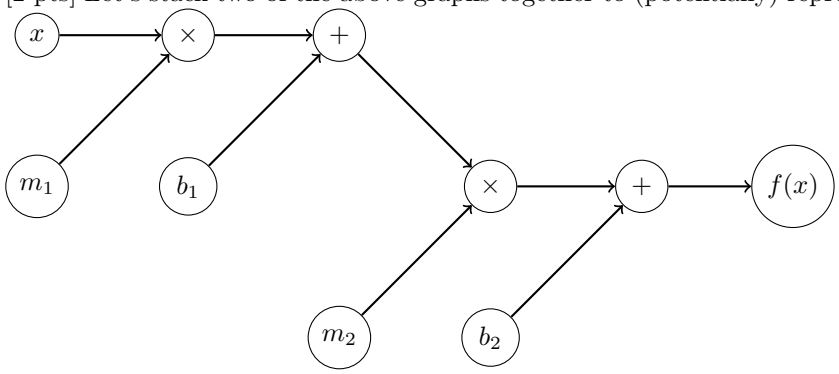
(i) [1 pt] Consider the following computation graph, which represents a 1 layer Neural Network.



1. Write the equation for the network's output ($y = f(x)$) in terms of m, x, b.

2. Describe the types of functions that can be encoded by such a function (given that the variables that it can control are m and b).

(ii) [2 pts] Let's stack two of the above graphs together to (potentially) represent a 2 layer "Neural Network."

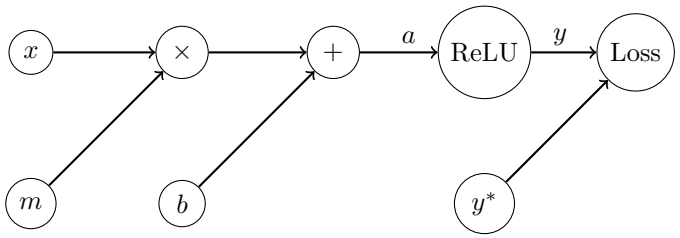


1. Write the equation for $y = f(x)$ in terms of m_1, m_2, b_1, b_2, x.

2. Describe the types of functions that can be encoded by such a function (given that the variables that it can control are the 4 learnable parameters m_1, m_2, b_1, b_2). Compare this with the previous neural network's expressive power.

3. Is this actually a 2-layer network? If it is, explain. If not, rewrite it (algebraically) as a 1-layer network with only 2 learnable weights. Why do neural networks need nonlinearities?

(iii) [4 pts] Now, let's go back to the first NN and add a nonlinearity node. Recall $ReLU(x) = \max(0, x)$. Also consider a loss function $Loss(y, y^*)$ which represents the error between our network's output (y) and the true label (y^*) from the dataset. We will perform an abbreviated version of backpropagation on this network.



1. Compute $\frac{\partial Loss}{\partial a}$ using Chain Rule. Use Mean Squared Error as the loss function, which is defined as $MSE(y, y^*) = (y - y^*)^2$ where y^* = true label and y is the predicted output from the neural network.

2. Find $\frac{\partial Loss}{\partial m}$. Note that since we are doing backprop, we can reuse calculations from part 1.

3. Find $\frac{\partial Loss}{\partial b}$. Note that since we are doing backprop, we can reuse calculations from part 1.

4. What is the gradient descent update rule for updating m? What is the update rule for b?

For the next few parts, we analyze the Perceptron algorithm. In the perceptron algorithm, we predict $+1$ if $\vec{w}^T \vec{f}(x) \geq 0$, and predict -1 else, where $\vec{f}(x)$ is a feature vector.

(d) [3 pts] When implementing the perceptron algorithm with a neural network, the following function might be of use: $sign(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases}$. If we added this $sign(x)$ node to our neural network drawings, what would happen during backpropagation through this node?
Hint: what does the gradient look like for various x values?

(e) [2 pts] Draw the binary perceptron prediction function as a "neural-network"-styled computation graph. Assume 3 dimensional weight and feature vectors: that is, $[w_0, w_1, w_2]$ is the weight vector and $[f_0(x), f_1(x), f_2(x)]$ is the feature vector. Recall that in the perceptron algorithm, we take the dot product of the weight vector and the feature vector. In addition to the addition and multiplication nodes, add a loss node at the end, to represent the prediction error which we would like to minimize. Label the edge which represents the perceptron model's output as y.
Hint: $y = sign(w_0 * f_0(x) + w_1 * f_1(x) + w_2 * f_2(x))$

(f) [2 pts] Using Mean Squared Error $(y - y^*)^2$ as the loss function, compute $\frac{\partial Loss}{\partial w_i}$. Because of the problem you noticed in the previous part with including the *sign* node, as we are doing chain rule below, use the custom gradient $\frac{\partial sign(x)}{\partial x} = \left[\frac{\partial sign(x)}{\partial x}\right]_{custom} = 1$.

(g) In this part, we will derive the gradient update rule for the perceptron using our graph above.

(i) [1 pt] The loss gradient is defined as $\nabla_w Loss = \begin{bmatrix} \frac{\partial Loss}{\partial w_0} \\ \frac{\partial Loss}{\partial w_1} \\ \frac{\partial Loss}{\partial w_2} \end{bmatrix}$. Using your answer from the previous question, write out the loss gradient.

(ii) [4 pts] What is the gradient update rule ($\vec{w} \leftarrow \vec{w} - \alpha \nabla_w Loss$) for the cases below?
Hint: your answers will be in terms of $\vec{f}(x)$ and α.

1. $y = -1, y^* = 1$

2. $y = 1, y^* = -1$

3. $y = y^*$

(iii) [1 pt] For $\alpha = \frac{1}{4}$, compare the update rules you derived for the 3 cases above with the Perceptron update formula in the notes and lecture. Briefly describe your observations.