Homework 3

CSE 3521 – Summer 2018

Instructor: Andrew R. Plummer

Department of Computer Science and Engineering
The Ohio State University

June 15th, 2018

Task: Work through each set of exercises. If you get stuck on any of the exercises you can ask Yi or myself for help by email or during office hours.

What to submit: Submit your answers for all of the exercises in this document to the appropriate dropbox on the Carmen site. Answers for the concept check and proof sections can be hand-written (e.g., submitted as a scanned image), but please make sure that your writing is readable. Answers to the coding section must be written in python and must be runnable by the grader.

Due date: Submit your answers to the Carmen dropbox by 11:59pm, Jun. 27th.

Concept check

1. (2pt) Using the alarm network on slide 3 of the Bayesian Inference slides, compute $P(B \,|\, +j, +m)$.

2. (3pt) Refer to the Naive Bayes Classifier shown below. Suppose C has domain $\{c_1, c_2, c_3\}$ and each X_i is a Boolean variable with values *true* and $false$. Using the Bayesian net G, compute the following distribution, showing the manner in which you derived your answer.

$$P(C \,|\, X_1 = false, X_2 = true, X_3 = false).$$

3. (3pt) The sigmoid function

$$s(z) = \frac{1}{1 + e^{-z}}$$

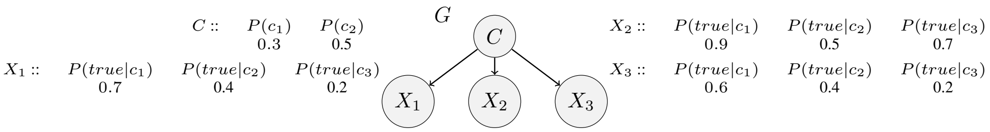

G
C :: $P(c_1)$ $P(c_2)$
0.3 0.5
X_1 :: $P(true|c_1)$ $P(true|c_2)$ $P(true|c_3)$
0.7 0.4 0.2
X_2 :: $P(true|c_1)$ $P(true|c_2)$ $P(true|c_3)$
0.9 0.5 0.7
X_3 :: $P(true|c_1)$ $P(true|c_2)$ $P(true|c_3)$
0.6 0.4 0.2
C
X_1
X_2
X_3

Figure 1: A Naive Bayes Classifier.

has derivative $s'(z) = s(z)(1 - s(z))$. Moreover, recall that during backpropogation the derivative $s'(z)$ is a factor in the gradient computation used to update the weights of a multilayer perceptron (see slides 28-30 in the neural-nets.pdf slide set). Activation functions like sigmoid have a "saturation" problem: when z is very large or very small, $s(z)$ is close to 1 or 0, respectively, and so $s'(z)$ is close to 0. As a result, corresponding gradients will be nearly 0, which slows down training. Affine activation functions with positive slope always have a positive derivative and thus will (more or less) not exhibit saturation, but they have other drawbacks (think back to lab 6). Do a little research and find a non-affine activation function that avoids the saturation problem (hint: ReLU). In your own words, describe how this activation is non-affine and also avoids the saturation problem. Briefly discuss any drawbacks your chosen activation function may have, as well as similar alternatives that avoid these drawbacks.

Coding

1. (8pt) Implement in python a convolutional layer (without identity activation) that computes the application of the 3x3 vertical and horizontal sobel masks below to an input image of size 5x5x3 with zero-padding of size 1. That is, the weights of your convolutional layer will not be learned, but rather hard-coded to match the values of the filters. To make things concrete, use the input volume and masks below:

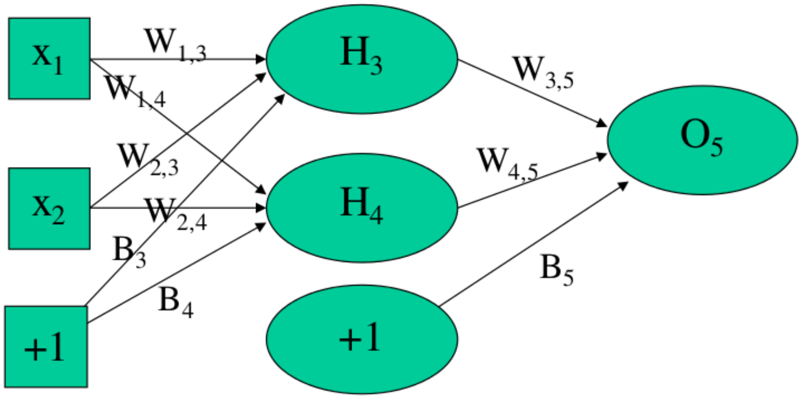
$$\text{input volume:} \quad \begin{matrix} 3 & 1 & 3 & 8 & 2 \\ 4 & 1 & 5 & 7 & 9 \\ 2 & 1 & 4 & 5 & 0 \\ 4 & 1 & 5 & 8 & 3 \\ 3 & 1 & 4 & 7 & 2 \end{matrix} \quad \begin{matrix} 5 & 4 & 1 & 3 & 8 \\ 4 & 9 & 1 & 4 & 7 \\ 7 & 3 & 1 & 4 & 6 \\ 8 & 4 & 1 & 5 & 2 \\ 2 & 3 & 1 & 8 & 2 \end{matrix} \quad \begin{matrix} 2 & 2 & 3 & 7 & 3 \\ 6 & 9 & 4 & 4 & 5 \\ 1 & 1 & 1 & 1 & 1 \\ 8 & 3 & 4 & 5 & 5 \\ 7 & 2 & 3 & 1 & 4 \end{matrix}$$

$$\text{vertical mask:}\quad \begin{matrix}-1 & 0 & 1\\ -2 & 0 & 2\\ -1 & 0 & 1\end{matrix} \qquad\qquad \text{horizontal mask:}\quad \begin{matrix}1 & 2 & 1\\ 0 & 0 & 0\\ -1 & -2 & -1\end{matrix}$$

2. (9 pt) Implement a perceptron that can learn the Boolean function AND using the threshold activation function.

Fun with proofs

1. (5pt) Prove that a multilayer perceptron with one hidden layer with two neurons and output layer with one neuron is an affine function of the input if the activation function for each neuron is an affine function. To make things simple and concrete, you need only demonstrate the result for the mlp show below. Briefly explain the implications of this result for using multilayer perceptrons with affine activation functions to learn the XOR data.



$$\begin{aligned} O_5 = g(\quad & W_{3,5} \cdot g(W_{1,3} \cdot x_1 + W_{2,3} \cdot x_2 + B_3) \\ & + W_{4,5} \cdot g(W_{1,4} \cdot x_1 + W_{2,4} \cdot x_2 + B_4) \\ & + B_5 \quad) \end{aligned}$$

Figure 2: Multilayer Perceptron