Homework 3

Automated Learning and Data Analysis
Dr. Thomas Price

Spring 2020

Instructions

Due Date: March, 31 2020 at 11:45 PM
Total Points: 100 for CSC522; 85 for CSC422.
Submission checklist:

- Clearly list each team member's names and Unity IDs at the top of your submission.
- Your submission should be a single zip file containing a PDF of your answers, your code, and (if needed) a README file with running instructions. **Name your file**: G(homework group number)_HW(homework number), e.g. G1_HW3.
- If a question asks you to explain or justify your answer, **give a brief explanation** using your own ideas, not a reference to the textbook or an online source.
- In addition to your group submission, please also *individually* submit your Peer Evaluation form on Moodle, evaluating yours and your teammates' contributions to this homework.

Problems

1. BN Inference (12 points) [**Ge Gao**]. Compute the following probabilities according to the Bayesian net shown in Figure 1. **Note**: $P(A)$ means $P(A = true)$; $P(\sim A)$ means $P(A = false)$.

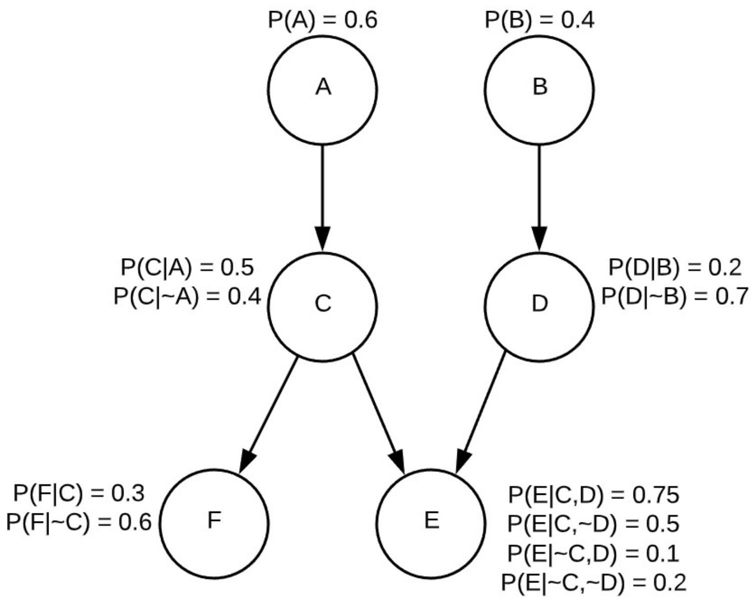

Figure 1: BN Inference

(a) Compute $P(C)$. Show your work.

(b) Compute $P(D|B, \sim C)$. Show your work.

(c) Compute $P(A, B, \sim C, D, E, F)$. Show your work.

(d) Are E and F conditionally independent given C? Justify your answer in 1 sentence.

(e) Are A and B marginally independent? Justify your answer in 1 sentence.

(f) Given evidence that $A = true$, $C = true$ $D = false$, and $F = true$, use the Bayes Net to predict whether E is more likely to be $true$ or $false$, or whether both are equally likely.

2. Linear Regression (18 points) [**Ge Gao**].

(a) Given the following four training data points of the form (x, y): $(2, 5)$, $(0, -2)$, $(1, -3)$, $(-2, -4)$, estimate the parameters for linear regression of the form $y = w_1 x^3 + w_0$.
Note that x is cubed in the formula.

i. Determine the values of w_1 and w_0 and show each step of your work.

ii. Calculate the training RMSE for the fitted linear regression.

3. ANN + Backpropagation (20 pts (522 students) / 15 pts (422 students)) [**Yang Shi**].

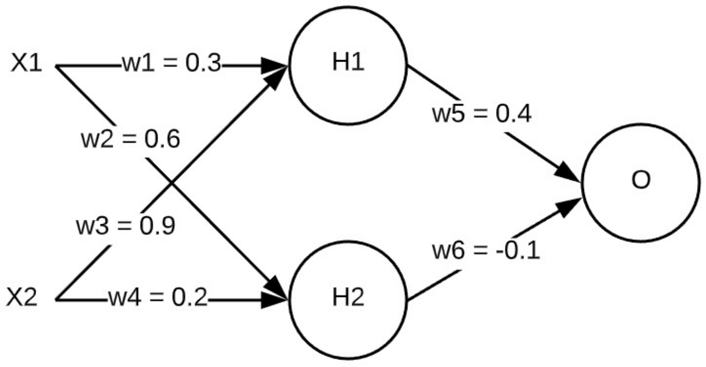

Figure 2: Neural Network Structure with initial weights

Table 1: Initial weights for given neural network in (a)

Weight	From	To	Initial Value
w1	X1	H1	0.3
w2	X1	H2	0.6
w3	X2	H1	0.9
w4	X2	H2	0.2
w5	H1	O	0.4
w6	H2	O	-0.1

You are given the above (Figure 2) neural network with continuous input attributes $X1$ and $X2$ and continuous output variable Y. For clarity, the relationship between weights and activations is also shown in Table 1. All three activations $H1$, $H2$ and O use the linear activation function $f(z) = Mz$, with constant $M = 1$. Initial weights are as given in Figure 2 and repeated in Table 1. There is **no bias** (w_0) added to any of the units. Answer the following:

(a) **Forward Pass**: If you are given one training data point: $X1_i = 1$, $X2_i = -1$, and $Y_i = 1$. Compute the activations of the neurons $H1$, $H2$ and O.

(b) **Backward Pass**: At the end of forward pass, using the current training instance i: $X1_i = 1$, $X2_i = -1$, and $Y_i = 1$, calculate the updated value of each of the following weights after one iteration of backpropagation:

- For CSC 522: $w1$, $w5$ and $w6$
- For CSC 422: $w5$ and $w6$ ($w1$ is optional extra credit)

Use 0.1 as your learning rate and MSE (mean squared error) as your cost function. Show your work on the following steps for each weight, w ($w1$, $w5$, $w6$):

i. Let a_N be the activation at neuron N, $X1_i$ be the value of the attribute $X1$ for instance i, and Y_i be the actual class of the instance i. Write equations to define the following:

A. The cost function C in terms of Y_i and a_O

B. The activation of the final layer a_O in terms of second layer weights w_5, w_6 and the activation of the first layer a_{H1} and a_{H2}

C. The activation of the node a_{H1} in terms of inputs $X1_i$, $X2_i$ and weights w_1 and w_3

ii. For layer-2 weights, calculate $\frac{\delta C}{\delta a_O}$ and $\frac{\delta a_O}{\delta w}$. Here C is the cost function, a_O is the activation at node O, and w is the weight.

iii. For layer-1 weights, calculate $\frac{\delta C}{\delta a_O}$, $\frac{\delta a_O}{\delta a_{H1}}$, and $\frac{\delta a_{H1}}{\delta w}$.

iv. Calculate $\frac{\delta C}{\delta w}$ using the above values.

v. Calculate the updated weight w' using the $\frac{\delta C}{\delta w}$ and the learning rate.

4. Programming (50 pts (522 students) / 40 pts (422 students)) [**Yang Shi**] In this question, you will be performing a variety of machine learning operations: Lasso regression and a Neural Network.

   (a) **PART-1: Lasso Regression** (15 points)

      i. **Dataset description:** You are provided a dataset with 20 variables. Variables $x1 - x19$ refer to the independent variables, while variable y is your dependent variable. Training data is stored in the file `data/regression-train.csv`, and test data is stored in the file `data/regression-test.csv`.

      ii. **Note:** The TA will use a different version of `data/regression-test.csv`. The format (independent variables $x1 - x19$, dependent variable y) will be similar, but TA's file may contain different number of data points than the file supplied to you. Please ensure you take this into account, and do not hard code any dimensions.

      iii. In this exercise, you will apply linear regression and Lasso regression methods to the dataset supplied to you, and then compare their results to determine whether Lasso regression is needed for this dataset:

      iv. **Learning (10 pts):** You will write code in the function `alda_regression()` to train simple linear regression and lasso regression models. Detailed instructions for implementation and allowed packages have been provided in `hw3.R`. Note that for the lasso regression model, you will be using crossvalidation to tune the λ hyperparameter.

      v. **Comparison (5 pts):** From the results, use the function `regression_compare_rmse()` to compare the two regression models. You should also use the `coef` function to inspect your Lasso model's coefficients. Use the output of these functions to answer the following questions *in your PDF report*:

         A. The dataset contains 19 attributes. Are all 19 attributes useful for predicting the dependent variable? Why or why not?

         B. If not all attributes are predictive, use your Lasso model to perform feature selection. Which attributes should be kept?

   (b) **PART-2: Neural Network Implementation** (35 points for 522 students, 25 points for 422 students)

      i. **Dataset description:** You are provided the same dataset from HW2 (`data/pima-indians-diabetes.csv`). You will use it to test a neural network that you write from scratch.

      ii. **Note:** The TA may test your code with a different version of the dataset, with a different number of attributes, so use the appropriate function parameters to create your ANN.

      iii. **Instructions**: You will implement a neural network from scratch. The framework code, detailed instructions and the allowed packages are in provided in `hw3_neural_net.R`, and the corresponding checker code is in `hw3_neural_net_checker.R`. As in other coding questions, you should only submit `hw3_neural_net.R` as the answer for this question. **For CSC 522 students**, you will implement a **2-layer** neural network. **For CSC 422 students**, you will need to implement a **1-layer** neural network (perceptron). You can implement a 2-layer neural network for extra credit.

      iv. **Note 1**: As explained in the code comments, you will be implementing this ANN using *matrix multiplication.* Rather than doing backpropagation one instance at a time (as in class, and in problem 3), you will calculating the weight updates for *all* training instances at once, and summing these to calculate the net change. In R, the `%*%` operator performs matrix multiplication.

      v. **Note 2**: For a given layer in the ANN, we use a matrix W to hold the values of all weights, where $W_{i,j}$ is the weight of the edge going from input node i to output node j.

      vi. **Sigmoid implementation (3 pts):** You will write code in the function `sigmoid()` to calculate the sigmoid function ($f(\mathbf{x}) = \frac{1}{1+e^{-\mathbf{x}}}$) given an input vector $\mathbf{x}$. Correspondingly, you will need to also write code to calculate the derivatives of sigmoid function in `sigmoid_derivative()`, which takes an input vector $\mathbf{x}$, and calculate $\frac{\delta f}{\delta \mathbf{x}}$.

      vii. **Loss Calculation (3 pts):** Assume that our loss function is $C = \sum_i (y_i - O_i)^2$ over all the instances i, where y_i is the class label for the instance i, and O_i is the activation value of the last layer for that instance. You will implement the code to calculate this function in `calculate_loss()`.

viii. **Activation Calculation (0 pts):** In the function `calculate_activations()`, we have implemented a function to calculate the activation vector for a given node, given the input matrix of activations from the prior layer, and the weight matrix. The input is a matrix, where each row corresponds to one training instance and each column corresponds to the activation of a node in the previous layer. Your output is a vector, where each item in the vector represents the outputs for one training instance. **Goal**: Read over this function and test it until you understand how the matrix multiplication can calculate a full layer of the ANN for *all* training instances in one matrix calculation.

ix. **Gradient Calculation (12 pts for 522, 7 pts for 422):** There are two kinds of gradients you will need to calculate to perform back propagation, besides the gradient for the activation function that you have already calculated in `sigmoid_derivative()`. In `calculate_dCdw()`, you will need to calculate the derivative of the cost with respect to a weight matrix. **522 only**: In `calculate_dCdf()`, you will need to calculate the derivative of the cost with respect to activation values, calculated by `calculate_activations()`. **Hint:** Use `https://en.wikipedia.org/wiki/Matrix_calculus` for references about derivative calculation for matrices.

x. **Network Training (17 pts for 522, 12 pts for 422):** After writing these functions above, you have all the components you need to train a neural network. You will implement your neural network in the function `neuralnet`. The first step is to read the training dataset in, and use `rnorm` to randomly initialize the weight matrices. You need one matrix for each layer in the ANN (2 for 522; 1 for 422). The number of nodes in each layer is given as a function parameter.

After initialization, you will need to use a `for` loop to train this neural network for the given number of iterations, or epochs, also given as a parameters. In every iteration of the loop, you will first do a forward pass to calculate the activations at your first hidden layer (522 only) and your output layer, using your `calculate_activations()` function. Then use the function `calculate_loss()` to get the loss (error) C for this iteration.

You will then use the loss to calculate the gradient of the cost with respect to the weights, using `calculate_dCdw()` (for the final layer) and `calculate_dCdf()` (for the first layer – 522 only). Note that you may need to use these functions multiple times if you are implementing a multiple layer neural network.

The final step is to use the gradient you calculated for every layer's weight to update the weights. You can print the loss for every epochs to track the change of the loss.

NOTE: Your entire solution `hw3_regression.R` and `hw3_neural_net.R` should not take more than 5 minutes to run. Any solution taking longer will be awarded a zero.