CS 486/686 Assignment 4
Fall 2022
(125 marks)

Blake VanBerlo

Due Date: 11:59 PM ET on Tuesday, December 6, 2022

Changes

- **v1.1:** Clarified weight updates as the sum of partial derivatives over *all* examples
- **v1.2:** Improved clarity of Q1(a) description and fixed typo in Q2(c)

Instructions

- Submit the signed academic integrity statement and any written solutions in a file to the Q0 box in the A4 project on Crowdmark. **(5 marks)**.
- Submit your written answers to questions 1 and 2 as PDF files to the Q1 and Q2 boxes respectively in the A4 project on Crowdmark. I strongly encourage you to complete your write-up in LaTeX, using this source file. If you do, in your submission, please replace the author with your name and student number. Please also remove the due date, the Instructions section, and the Learning Goals section. Thank you!
- Submit any code to `Marmoset` at `https://marmoset.student.cs.uwaterloo.ca/`. Be sure to submit your code to the project named `Assignment 4 - Final`.
- No late assignment will be accepted. This assignment is to be done individually.
- Lead TAs:
  - Jess Gano (`jgano@uwaterloo.ca`)
  - Ali Saheb Pasand (`ali.sahebpasand@uwaterloo.ca`)

  The TAs' office hours will be scheduled and posted on LEARN and Piazza.

Learning goals

Decision Networks

- Construct a decision network for a given scenario
- Determine the number of policies in a decision network
- Execute the Variable Elimination Algorithm to determine the optimal policy for a decision network and its expected utility

Neural networks

- Implement a multilayer perceptron
- Implement the backpropagation algorithm
- Understand and interpret performance metrics in supervised learning

1 Decision Networks (25 marks)

Brett is deciding what gift(s) to buy her boyfriend for the holidays. She is hoping to order him a new 2022-2023 Toronto Maple Leafs jersey. If she orders it now, the price is \$250. There is a 30% chance that the store is going to have a flash sale on December 23, during which *all items will be discounted by 30%*. Brett also thinks there is a 30% chance the store will experience supply chain issues in December. If the store does not experience supply chain issues, Brett estimates that there is a 15% chance that the jersey will be sold out on December 23. However, if the store *does* experience supply chain issues, the probability the new jersey would be sold out on December 23 would be 75%. If she chooses to purchase a jersey on December 23 and the new jersey is sold out, she will receive last year's 2021-2022 edition of the jersey, which regularly sells for \$150. She therefore has two decisions to make:

- Whether to purchase one new jersey now
- Whether to purchase one jersey on December 23. When making this decision, she knows whether she bought the new jersey beforehand, whether the flash sale occurred, and whether the new jersey is sold out

Let Brett's utility function be the following:

$$U = 30\,\mathbf{1}_{[J_n \geq 1]} + 18\,\mathbf{1}_{[J_o \geq 1]} - 0.1C - 5$$

where J_n is the total number of new jerseys she purchases, J_o is the number of old jerseys she purchases, and C is the total cost of all jerseys she purchases. The indicator function $\mathbf{1}_{[\cdot]}$ evaluates to 1 when the condition in the subscript is met and 0 otherwise. For example, if Brett buys one new jersey right now and one old jersey on December 23 (because the new jersey is sold out) *and* there is a flash sale, the utility function would evaluate to $30(1) + 18(1) - 0.1(250 + (1 - 0.3)(150)) - 5 = 7.5$. As another example, if she chooses *not* to buy the new jersey now, *and* she buys the new jersey on December 23 (since it's not sold out) *and* there is *no* flash sale, then the utility is $30(1) + 18(0) - 0.1(250) - 5 = 0$.

(a) Construct a decision network for Brett's situation. Write out the factors for the utility node and each chance node. Use the following variables:

- B_1: 1 if Brett buys the new jersey right now; 0 otherwise
- F: 1 if there is a flash sale on December 23; 0 otherwise
- I: 1 if the store experiences supply chain issues; 0 otherwise
- S: 1 if the new jersey is sold out on December 23; 0 otherwise
- B_2: 1 if Brett buys a jersey on December 23; 0 otherwise
- U: Brett's utility

Marking Scheme: (10 marks)

- (6 marks) Correct factors
- (4 marks) Correct edges on decision network

(b) Recall that a policy consists of a decision function for *each* decision variable. How many possible policies are there for Brett's situation?

Marking Scheme: (2 marks)

- (2 marks) Correct total number of policies

(c) Using the Variable Elimination Algorithm for decision networks, determine the optimal policy for Brett's situation and its expected utility (rounded to 4 decimal places). Determine the policy for B_2 before you determine the policy for B_1. When there are multiple chance nodes to eliminate in a single step, eliminate them in alphabetical order.

Write out each VEA operation, along with the factor that is produced by the operation. For example, if you were to sum out Y from $f_1(X, Y)$, you would write:

Sum out Y from f_1(X, Y) to produce f_2(X).

f_2(X):

X	Value
0	0.6
1	0.4

To save space, you don't have to rewrite the factors you defined in 1(a).

Marking Scheme: (13 marks)

- (10 marks) Correct execution of VEA (including final policy and expected utility)
- (3 marks) VEA operations and factors listed clearly

2 Neural Networks (100 marks)

In this part of the assignment, you will implement a feedforward neural network from scratch. Additionally, you will implement multiple activation functions, loss functions, and performance metrics. Lastly, you will train a neural network model to perform both a classification and a regression task.

2.1 Bank Note Forgery - A Classification Problem

The classification problem we will examine is the prediction of whether or not a bank note is forged. The labelled dataset included in the assignment was downloaded from the UCI Machine Learning Repository. The target $y \in \{0, 1\}$ is a binary variable, where 0 and 1 refer to fake and real respectively. The features are all real-valued. They are listed below:

- Variance of the transformed image of the bank note
- Skewness of the transformed image of the bank note
- Curtosis of the transformed image of the bank note
- Entropy of the image

2.2 Red Wine Quality - A Regression Problem

The task is to predict the quality of red wine from northern Portugal, given some physical characteristics of the wine. The target $y \in [0, 10]$ is a continuous variable, where 10 is the best possible wine, according to human tasters. Again, this dataset was downloaded from the UCI Machine Learning Repository. The features are all real-valued. They are listed below:

- Fixed acidity
- Volatile acidity
- Citric acid
- Residual sugar
- Chlorides
- Free sulfur dioxide
- Total sulfur dioxide
- Density
- pH
- Sulphates
- Alcohol

2.3 Training a Neural Network

In Lecture 15, you learned how to train a neural network using the backpropagation algorithm. In this assignment, you will apply the forward and backward pass to the entire dataset simultaneously (i.e. batch gradient descent, where one batch is the entire dataset). As a result, your forward and backward passes will manipulate tensors, where the first dimension is the number of examples in the training set, n. When updating an individual weight $W_{i,j}^{(l)}$, you will need to find the sum of partial derivatives $\frac{\partial E}{\partial W_{i,j}^{(l)}}$ across all examples in the training set to apply the update. Algorithm 1 gives the training algorithm in terms of functions that you will implement in this assignment. Further details can be found in the documentation for each function in the provided source code.

Algorithm 1 Gradient descent with backpropagation

```
Require: $\eta > 0$                                              ▷ Learning rate
Require: $n_{epochs} \in \mathbb{N}^+$                            ▷ Number of epochs
Require: $X \in \mathbb{R}^{n \times f}$                          ▷ Training examples with $n$ examples and $f$ features
Require: $y \in \mathbb{R}^n$                                     ▷ Targets for training examples
  Initiate weight matrices $W^{(l)}$ randomly for each layer.     ▷ Initialize net
  for $i \in \{1, 2, \ldots, n_{epochs}\}$ do                     ▷ Conduct $n_{epochs}$ epochs
      A_vals, Z_vals ← net.forward_pass($X$)                      ▷ Forward pass
      $\hat{y}$ ← Z_vals[-1]                                      ▷ Predictions
      $L \leftarrow \mathcal{L}(\hat{y}, y)$
      Compute $\frac{\partial}{\partial \hat{y}}\mathcal{L}(\hat{y}, y)$   ▷ Derivative of error with respect to predictions
      deltas ← backward_pass(A_vals, $\frac{\partial}{\partial \hat{y}}\mathcal{L}(\hat{y}, y)$ )   ▷ Backward pass
      update_gradients()                                          ▷ $W_{i,j}^{(\ell)} \leftarrow W_{i,j}^{(\ell)} - \eta \sum_n \frac{\partial \mathcal{L}}{\partial W_{i,j}^{(\ell)}}$ for each weight
  end for
  return trained weight matrices $W^{(\ell)}$
```

2.4 Activation and Loss Functions

You will implement the following activation functions and their derivatives:

Sigmoid

$$g(x) = \frac{1}{1 + e^{-kx}}$$

Hyperbolic tangent

$$g(x) = \tanh x$$

ReLU

$$g(x) = \max(0, x)$$

Leaky ReLU

$$g(x) = \max(0, x) + \min(0, kx)$$

You will implement the following loss functions and their derivatives:

Cross entropy loss: for binary classification

Compute the average over all the examples. Note that log() refers to the natural logarithm.

$$\mathcal{L}(\hat{y}, y) = \frac{1}{n} \sum_{i=1}^{n} -(y \log(\hat{y}) + (1 - y) \log(1 - \hat{y}))$$

Mean squared error loss: for regression

$$\mathcal{L}(\hat{y}, y) = \frac{1}{n} \sum_{i=1}^{n} (\hat{y} - y)^2$$

2.5 Implementation

We have provided three Python files. Please read the detailed comments in the provided files carefully. Note that some functions have already been implemented for you.

1. `neural_network.py`: Contains an implementation of a `NeuralNetwork` class. You must implement the `forward_pass()`, `backward_pass()`, and `update_weights()` methods in the `NeuralNetwork` class. **Do not change the function signatures. Do not change anything else in this file!**
2. `operations.py`: Contains multiple classes for multiple activation functions, loss functions, and functions for performance metrics. The activation functions extend a base `Activation` class and the loss functions extend a base `Loss` class. You must implement all the blank functions as indicated in this file. **Do not change the function signatures. Do not change anything else in this file!**
3. `train_experiment.py`: Provides a demonstration of how to define a `NeuralNetwork` object and train it on one of the provided datasets. Feel free to change this file as you desire.

Please complete the following tasks.

(a) Implement the empty functions in `neural_network.py` and `operations.py`. Zip and submit these two files on Marmoset.

Please do not invoke any numpy random operations in `neural_network.py` and `operations.py`. This may tamper with the automatic grading.

Marking Scheme: (84 marks)

Unit tests for `neural_network.py`:

- `NeuralNetwork.forward_pass()`
  (1 public test + 2 secret tests) * 4 marks = 12 marks
- `NeuralNetwork.backward_pass()`
  (1 public test + 2 secret tests) * 5 marks = 15 marks
- `NeuralNetwork.update_weights()`
  (1 public test + 2 secret tests) * 5 marks = 15 marks

Unit tests for `operations.py`:

- `Sigmoid.value()`
  (1 public test + 2 secret tests) * 1 mark = 3 marks
- `Sigmoid.derivative()`
  (1 public test + 2 secret tests) * 1 mark = 3 marks
- `Tanh.value()`
  (1 public test + 2 secret tests) * 1 mark = 3 marks
- `Tanh.derivative()`
  (1 public test + 2 secret tests) * 1 mark = 3 marks
- `ReLU.value()`
  (1 public test + 2 secret tests) * 1 mark = 3 marks
- `ReLU.derivative()`
  (1 public test + 2 secret tests) * 1 mark = 3 marks
- `LeakyReLU.value()`
  (1 public test + 2 secret tests) * 1 mark = 3 marks
- `LeakyReLU.derivative()`
  (1 public test + 2 secret tests) * 1 mark = 3 marks
- `CrossEntropy.value()`
  (1 public test + 2 secret tests) * 1 mark = 3 marks
- `CrossEntropy.derivative()`
  (1 public test + 2 secret tests) * 1 mark = 3 marks

- `MeanSquaredError.value()`
  (1 public test + 2 secret tests) * 1 mark = 3 marks
- `MeanSquaredError.derivative()`
  (1 public test + 2 secret tests) * 1 mark = 3 marks
- `accuracy`
  (1 public test + 2 secret tests) * 1 mark = 3 marks
- `mean_absolute_error`
  (1 public test + 2 secret tests) * 1 mark = 3 marks

Once you have implemented the functions, you can train the neural networks on the two provided datasets. The bank note forgery dataset is in `data/banknote_authentication.csv` and the wine quality dataset is in `data/wine_quality.csv`. In `train_experiment.py`, we have provided some code to instantiate a neural network and train on an entire dataset. Implement the required functions and then complete the next activities.

(b) Execute k-fold cross validation for the banknote forgery dataset with $k = 5$. Use the sigmoid activation function for your output layer. Report the number of layers, the number of neurons in each layer, and the activation functions you used for your hidden layers. Train for 1000 epochs in each trial and use $\eta = 0.01$.

To perform cross validation, randomly split the data into 5 folds. For each fold, train the model on the remaining data and determine the trained model's accuracy on the validation set *after training is complete*. You can use `NeuralNetwork.evaluate()` to determine the accuracy on the validation set (i.e. fold).

Produce a plot where the x-axis is the epoch number and the y-axis is the average training loss across all experiments for the current epoch. Report the average and standard deviation of the accuracy on the validation set over each experiment.

For example, for your first fold, 80% of the examples should be in the training set and 20% of the examples should be in the validation set (i.e. fold 1). You will require the loss obtained after executing the forward pass for each of the 1000 epochs. After model has trained, use the trained model to calculate the accuracy on the validation set. This is one experiment. You will need to run this experiment 5 times in total, plotting the average loss at epoch i for each epoch. You will report the average and standard deviation of the accuracy achieved on the validation set during each experiment.

Marking Scheme: (8 marks)

- (6 marks) Reasonably correct plot.
- (2 marks) Reasonable accuracy (average and standard deviation)

(c) Execute k-fold cross validation for the wine quality dataset with $k = 5$. Use the Identity activation function for your output layer.Report the number of layers, the number of neurons in each layer, and the activation functions you used for your hidden layers. Train for 1000 epochs in each trial and use $\eta = 0.001$.

To perform cross validation, randomly split the data into 5 folds. For each fold, train the model on the remaining data and determine the trained model's mean absolute error on the fold. You can use `NeuralNetwork.evaluate()` to determine the mean absolute error on the validation set (i.e. fold).

Produce a plot where the x-axis is the epoch number and the y-axis is the average training loss across all experiments for the current epoch. Report the average and standard deviation of the mean absolute error on the validation set over each experiment.

Marking Scheme: (8 marks)

- (6 marks) Reasonably correct plot.
- (2 marks) Reasonable mean absolute error (average and standard deviation)