CS 486/686 Assignment 4
Winter 2023
(90 marks)

Wenhu Chen

Due Date: 11:59 PM ET on April 10th

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. 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`.
- Late assignment will be accepted with 5% off per day. This assignment is to be done individually.
- Lead TAs:
  - Thakur, Nandan (n3thakur@uwaterloo.ca)

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

1 Decision Tree (25 marks)

Jeeves is a valet to Bertie Wooster. On some days, Bertie likes to play tennis and asks Jeeves to lay out his tennis things and book the court. Jeeves would like to predict whether Bertie will play tennis (and so be a better valet). Each morning over the last two weeks, Jeeves has recorded whether Bertie played tennis on that day and various attributes of the weather (training set).

Jeeves would like to evaluate the classifier he has come up with for predicting whether Bertie will play tennis. Each morning over the next two weeks, Jeeves records more data (test set).

Training Set:

Day	Outlook	Temp	Humidity	Wind	Tennis?
1	Sunny	Hot	High	Weak	Yes
2	Sunny	Hot	High	Strong	Yes
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	No
5	Rain	Cool	Normal	Weak	No
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	No
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	No
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Test set:

Day	Outlook	Temp	Humidity	Wind	Tennis?
1	Sunny	Mild	High	Strong	Yes
2	Rain	Hot	Normal	Strong	No
3	Rain	Cool	High	Strong	No
4	Overcast	Hot	High	Strong	No
5	Overcast	Cool	Normal	Weak	Yes
6	Rain	Hot	High	Weak	No
7	Overcast	Mild	Normal	Weak	No
8	Overcast	Cool	High	Weak	No
9	Rain	Cool	High	Weak	No
10	Rain	Mild	Normal	Strong	No
11	Overcast	Mild	High	Weak	Yes
12	Sunny	Mild	Normal	Weak	Yes
13	Sunny	Cool	High	Strong	No
14	Sunny	Cool	High	Weak	No

(a) Construct a decision tree using ID3 algorithm. Basically, using greedy algorithm to find the testing feature which can bring most information gain at each level. And then you need to expand the tree until: (1) the node is dominated by a class, (2) the node contains no more features, (3) the node contains no examples. You need to show the whole construction process step by step, please provide all the entropy, information gain, etc to justify your solution.

Marking Scheme: (21 marks)

- (10 marks) Correct construction process.
- (6 marks) Correct computation at each step.
- (5 marks) Correct final tree structure.

(b) What is the test set accuracy based on your constructed decision tree.

Marking Scheme: (4 marks)

- (4 marks) Correctly calculate the accuracy.

2 Neural Networks (65 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` $\leftarrow$ `net.forward_pass(`X`)` ▷ Forward pass
    $\hat{y} \leftarrow$ `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` $\leftarrow$ `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

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

ReLU

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

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_net.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_net.py` and `operations.py`. Zip and submit these two files on Marmoset.

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

Marking Scheme: (52 marks)

Unit tests for `neural_network.py`:

- `NeuralNetwork.forward_pass()`
  (1 public test + 1 secret test) * 6 marks = 12 marks
- `NeuralNetwork.backward_pass()`
  (1 public test + 1 secret test) * 6 marks = 12 marks
- `NeuralNetwork.update_weights()`
  (1 public test + 1 secret test) * 6 marks = 12 marks

Unit tests for `operations.py`:

- `Sigmoid.value()`
  (1 public test + 1 secret test) * 1 mark = 2 marks
- `Sigmoid.derivative()`
  (1 public test + 1 secret test) * 1 mark = 2 marks
- `ReLU.value()`
  (1 public test + 1 secret test) * 1 mark = 2 marks
- `ReLU.derivative()`
  (1 public test + 1 secret test) * 1 mark = 2 marks
- `CrossEntropy.value()`
  (1 public test + 1 secret test) * 1 mark = 2 marks
- `CrossEntropy.derivative()`
  (1 public test + 1 secret test) * 1 mark = 2 marks
- `MeanSquaredError.value()`
  (1 public test + 1 secret test) * 1 mark = 2 marks
- `MeanSquaredError.derivative()`
  (1 public test + 1 secret test) * 1 mark = 2 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: (6 marks)

- (4 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: (7 marks)

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