CS 486/686 Assignment 4
Winter 2023

1 Neural Networks

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.

1.1 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

1.2 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

1.3 Training a Neural Network

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.

1.4 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

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

1.5 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.

Implement the empty functions in `neural_net.py` and `operations.py`.

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`.