CSE 6363: Machine Learning
University of Texas at Arlington
Spring 2022
Alex Dillhoff

Assignment 2

This assignment covers neural networks and backpropagation.

1 Neural Network Library

In this first part of this assignment, you will create a neural network library. The library will be made up of documented classes and functions that allow users to easily construct a neural network with an arbitrary number of layers and nodes. Through implementing this library, you will understand more clearly the atomic components that make up a basic neural network.

1.1 The `Layer` Class

For the layers you will create in this assignment, it is worth it to create a parent class named `Layer` which defined the forward and backward functions that are used by all layers.
In this way, we can take advantage of polymorphism to easily compute the forward and backward passes of the entire network.

1.2 Linear Layer

Create a class that implements a linear layer. The class should inherit the `Layer` class and implement both `forward` and `backward`.
For a given input, the forward pass is computed as

$$f(\mathbf{x}; \mathbf{w}) = \mathbf{w}^T \mathbf{x}.$$

The backward pass should compute the gradient with respect to the weights.

$$\frac{d}{d\mathbf{w}} f(x; \mathbf{w}) = \mathbf{x}$$

This is then multiplied with the gradients computed by the layer ahead of this one.

1.3 Sigmoid Function

Create a class that implements the logistic sigmoid function. The class should inherit the `Layer` class and implement both `forward` and `backward`.

1.4 Hyperbolic Tangent Function

Create a class that implements the hyperbolic tangent function. The class should inherit the `Layer` class and implement both `forward` and `backward`.

1.5 Softmax

Create a class that implements the softmax function. The class should inherit the `Layer` class and implement both `forward` and `backward`.
The softmax function is defined as

$$p_k = \frac{e^{f_k}}{\sum_j e^{f_j}}.$$

The backward pass should compute the gradient with respect to the weights.

$$\frac{d}{d\mathbf{w}} f(x; \mathbf{w}) = \mathbf{x}$$

1.6 Negative Log Likelihood

Create a class that implements negative log likelihood loss. The class should inherit the `Layer` class and implement both `forward` and `backward`.

1.7 The `Sequential` Class

In order to create a clean interface that includes multiple layers, you will need to create a class that contains a list of layers which make up the network. The `Sequential` class will contain a list of layers. New layers can be added to it by appending them to the current list. This class will also inherit from the `Layer` class so that it can call forward and backward as required.

1.8 Saving and Loading

Implement a weight saving and loading feature for a constructed network such that all model weights can be saved to and loaded form a file. This will enable trained models to be stored and shared.

2 Testing your library

Construct a neural network with 1 hidden layer of 2 nodes in order to solve the XOR problem. Construct the input using `numpy`. You can reference the code we used for multi-layer perceptrons in class to help. Train and verify that your model can solve the XOR problem.

Save the weights as `XOR_solved.w`.

3 Handwritten Digit Recognition

In the second part of the assignment, you will use your neural network library to construct several networks for handwritten digit recognition. There are 60000 training images and 10000 testing images. You should randomly select 10% of the training images to use as a validation set.

In this part, you can experiment with the number of layers and nodes per layer as you wish. Use the loss of the validation set to guide your selection of hyperparameters. Experiment with at least 3 configurations of hyperparameters, plotting the training and validation loss as you train each configuration. Stop training when the loss does not improve after 5 steps. In your report, include the training/validation plots with each choice of hyperparameters.

Once you have trained at least 3 different models, evaluate each one on the test set. Include the test accuracy with your report.

4 Playing with Hyperparameters

The model behaves differently depending on many factors. Hyperparameters play a huge role in this. Experiment with the following scenarios and include a written section describing your observations during training under each scenario.

- Initialize the model with all zeros.
- Initialize the model with random values between -10 and 10.
- Train on MNIST using a learning rate of 1 and then again with a learning rate of 0.001. Plot the training loss curve in both instances.

Submission

Create a zip file that includes all of your code as well as your report. The TA should be able to easily run the code to reproduce all plots and results. Include any additional instructions, if necessary.