ML Assignment 2: Artificial Neural Networks

Anirudh Agrawal	Aviral Agarwal	Vikramjeet Das
2018A7PS0099H	2018A7PS0192H	2018A7PS0280H

1 Introduction and Implementation

An Artificial Neural Network consists of multiple layers which process the data and predict the output. Each layer consists of multiple nodes, which takes inputs from the previous layers and transforms it according to a function. The goal of the Artificial Neural Network is to learn a suitable set of weights to minimise the loss function. This is done by updating the weights using gradient descent. We briefly mention the forward and backprop equations of the different kinds of layers/units we implement.

1.1 Dense Layer

1.1.1 Parameters

- Weight matrix W of shape $d_i \times d_o$
- Bias vector b of shape $1 \times d_o$

1.1.2 Forward Propagation

$$f(x) = x \cdot W + b$$

1.1.3 Backward Propagation

$$\nabla_W f(x) = x$$
$$\nabla_b f(x) = 1$$

1.2 Tanh

1.2.1 Backward Propagation

$$\nabla_x tanh(x) = 1 - tanh(x)^2$$

1.3 Sigmoid

1.3.1 Forward Propagation

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

1.3.2 Backward Propagation

$$\nabla_x \sigma(x) = \sigma(x)(1 - \sigma(x))$$

1.4 Rectified Linear Unit (ReLU)

1.4.1 Forward Propagation

$$relu(x) = max(x, 0)$$

1.4.2 Backward Propagation

$$\nabla_x relu(x) = \begin{cases} 0, & \text{if } x \leq 0 \\ 1, & \text{otherwise} \end{cases}$$

1.5 Mean Squared Error

1.5.1 Forward Propagation

$$mse(y, \hat{y}) = \frac{(y - \hat{y})^2}{n}, where\ there\ are\ n\ samples$$

1.5.2 Backward Propagation

$$\nabla_{\hat{y}} mse(y, \hat{y}) = \frac{2(\hat{y} - y)}{n}$$

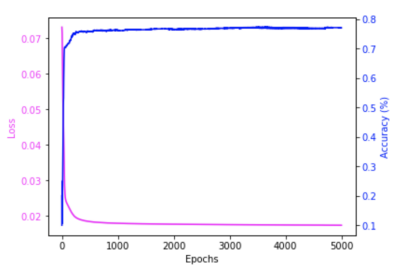
1.6 Binary Crossentropy

1.6.1 Forward Propagation

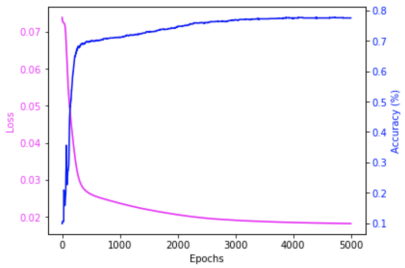
$$BCE(y, \hat{y}) = ylog(\hat{y}) + (1 - y)log((1 \hat{-} y))$$

1.6.2 Backward Propagation

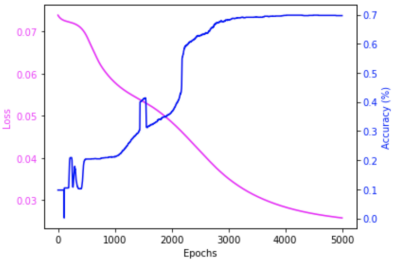
$$\nabla_{\hat{y}} BCE(y, \hat{y}) = \frac{1 - y}{1 - \hat{y}} - \frac{y}{\hat{y}}$$



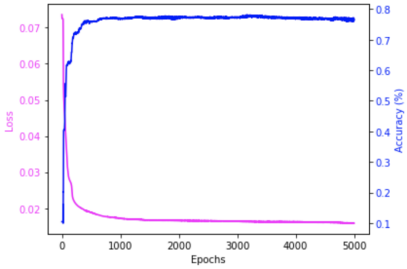
(a) 1 layer NN with lr=0.1



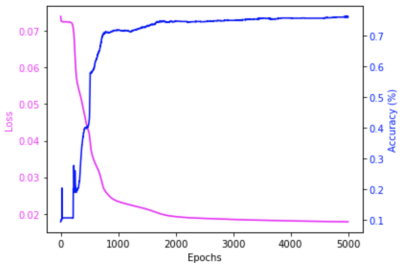
(b) 1 layer NN with lr=0.01



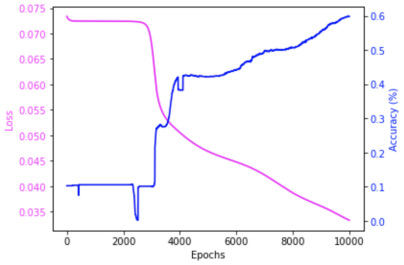
(c) 1 layer NN with lr=0.001



(d) 2 layer NN with lr=0.1



(e) 2 layer NN with lr=0.01



(f) 2 layer NN with lr=0.001

1.7 Categorical Crossentropy with Logits

1.7.1 Forward Propagation

$$CCE(y, \hat{y}) = \frac{1}{n} \sum_{i \ \epsilon \ class} y_i ln(softmax(\hat{y}_i))$$

$$softmax(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{k}} e^{x_j}$$

1.7.2 Backward Propagation

$$\nabla_{\hat{y}} CCE(y, \hat{y}) = \hat{y} - y$$

1.8 Feature Scaling

We pre-process our training data by scaling each feature to an interval of [0, 1]. This is essentially Min-Max Scaling. We transform each feature in the following way:

$$x_j^{(i)} := \frac{x_j^{(i)} - min(x_j)}{max(x_j) - min(x_j)}$$

where $i = 1...m$ represents each training example, and $j = 1...n$ represents each feature. Next, we apply the same transformation to the test dataset, using the minimum and maximum of each feature as calculated from the train dataset. The transformation for the test dataset is given by:

$$x_{j_{test}}^{(i)} := \frac{x_{j_{test}}^{(i)} - min(x_{j_{train}})}{max(x_{j_{train}}) - min(x_{j_{train}})}$$

2 Hyperparameters

We show our results with two configurations of model structure. The first with 1 intermediate layer with 20 units, and the other with 2 intermediate layers with 20 units. We use the ReLU activation in all intermediate layers and softmax activation in final layer.

3 Results

With one intermediate layer and training for 5000 epochs, we get a final test loss of 0.0206, and a test accuracy of 75.39%. With 2 intermediate layers and training for 5000 epochs, we get a final test loss of 0.0227, and a test accuracy of 75.55%.