Assignment I

Alok Kumar Trivedi

MLP Model :

procedure of building the MLP Model

- Initialized the W1 , W2 matrices of required dimension with random number and b1 and b2 with zeros
- Divided the X and y data set in batches of 500
- Decided to keep one hidden layer of size 64
- determined z1 and applied relu activation function to get A1, similarly determined z2 and applied softmax activation function to derive A2
- used cross entropy loss function to get the error value
- determine dz2 and dz1 using appropriate formulas to adjust the weights with the help of a learning rate value
- perfomed mini batch gradient descent with batch size of 500 for 400 epochs.

Architecture of the MLP Model

- input layer - (512, None), Hidden layer - 64, output layer - 10
- weight W_1 - (64, 512),bias b_1- (64,1), weight w_2 - (10,64) , bias b_2- (10,1)

Learning rate for augmented data - .01, Learning rate for unaugmented data - .008,

Epochs - 800

Batch size - 500

Evaluation Metrics - As test data has 10000 dataset and all ten classes are equally represented .hence we can use Accuracy as Evaluation Metrics

Instruction :

- The Data set used for training the model were first pre-processed by dividing them by 255 , then they were passed to the feature extraction function and then fed to the model
- Model requires Numpy and Matplotlib library

Backpropagation :

The above model includes input layer , one hidden layer and output layer of size 512,64 and 10 respectively. In order to provide non-linearity to our model we have used Relu activation function for the hidden layer and softmax activation function for the output layer as it has more than two classes .

Backpropagation is a process of adjusting weights of model according to the loss incurred. Here we determine impact of total loss on all weight by taking the derivative of the loss with respect to the weights.

$$L = \frac{1}{m} * \sum y * \log A2$$

Here L is total loss incurred , m is size of the batch , y is actual vector where A is final output vector which stores different probability value for each possible outcome.

As we have one hidden layer in our model, we have two weight vector of size (64,512) and (10,64) respectively along with two bias vector b1 and b2 of size (64,1) and (10,1) respectively. we will derive derivative of loss function with respect to each weight and bias vector.

$$\frac{\partial L}{\partial W_2} = \frac{\partial L}{\partial A2} * \frac{\partial A2}{\partial Z2} * \frac{\partial Z2}{\partial W2}$$

we will derive each term separately .

First we will derive term $\frac{\partial A2}{\partial Z2}$.

Since we have used softmax activation function at the output layer

$$A = \frac{e^z}{\sum e^z}$$

$$a_i = \frac{e_i^z}{e_i^z + \sum_{k!=i} e_k^z}$$

$$\frac{\partial a_i}{\partial z_j} = \frac{\partial \frac{e_i^z}{e_i^z + \sum_{k!=i} e_k^z}}{\partial z_j}$$

When i != j,

$$\frac{\partial a_i}{\partial z_j} = \frac{(e_j^z + \sum_{k!=j} e_k^z) * 0 - (e_i^z * e_j^z)}{(e_j^z + \sum_{k!=j} e_k^z)^2}$$

$$= -\frac{e^{z_i}}{\sum e^{z_i}} * \frac{e^{z_j}}{\sum e^{z_i}}$$

$$= -a_i * a_j$$

when i = j,

$$\frac{\partial a_i}{\partial z_j} = \frac{(e_i^z + \sum_{k!=i} e_k^z) * e^{z_i} - e^{z_i} * e^{z_i}}{(e_i^z + \sum_{k!=i} e_k^z)^2}$$

$$= \frac{e^{z_i}}{\sum e^{z_k}} * \frac{\sum_{k!=i} e^{z_k}}{\sum e^{z_k}}$$

$$= a_i * (1 - a_i)$$

Now,

$$\frac{\partial L}{\partial a_i} = -\frac{\partial y_i * \log_a i + \sum_{k!=i} y_k * \log_a k}{a_i}$$

$$\frac{\partial L}{\partial z_i} = \frac{\partial L}{\partial a_i} * \frac{\partial a_i}{\partial z_i}$$

$$= -\frac{y_i}{a_i} * \frac{\partial a_i}{\partial z_i} - \sum_{k!=i} \frac{y_k}{a_k} * \frac{\partial a_k}{\partial z_i}$$

$$= -\frac{y_i}{a_i} * a_i * (1 - a_i) - \sum_{k!=i} \frac{y_k}{a_k} * a_k * a_i$$

$$= -y_i + y_i * a_i - a_i * \sum_{k!=i} y_k$$

$$= -y_i + a_i * \sum y_k$$

$$a_i - y_i$$

$$\frac{\partial L}{\partial z_i} = a_i - y_i$$

Let,

$$dz2 = \frac{\partial L}{\partial z_i}$$

Now ,

$$\frac{\partial Z2}{\partial W2} = A1$$

$$\frac{\partial L}{\partial W2} = (A2 - Y) * A1$$

similarly ,

$$\frac{\partial L}{\partial b2} = \frac{\partial L}{\partial A2} * \frac{\partial A2}{\partial Z2} * \frac{\partial Z2}{\partial b2}$$

$$\frac{\partial Z2}{\partial b2} = 1$$

,

$$\frac{\partial L}{\partial b2} = \sum (A2 - Y)$$

Now ,

$$\frac{\partial L}{\partial W1} = \frac{\partial L}{\partial A2} * \frac{\partial A2}{\partial Z2} * \frac{\partial Z2}{\partial A1} * \frac{\partial A1}{\partial Z1} * \frac{\partial Z1}{\partial W1}$$

we know that,

$$\frac{\partial L}{\partial A2} * \frac{\partial A2}{\partial Z2} = (A2 - Y)$$

Also,

$$\frac{\partial Z2}{\partial A1} = W2$$

Since Hidden layer activation function is Relu, and derivative of relu with respect to z1 is

$$\frac{\partial A1}{\partial Z1} = 1, when z1 >= 0$$

and

$$\frac{\partial A1}{\partial Z1} = 0, when z1 < 0$$

Let d be a matrix of shape and size same as z1 which stores when z1 greater than or equal to 0 and 0 when z1 is less than 0
then,

$$\frac{\partial A1}{\partial Z1} = d$$

and,

$$\frac{\partial Z1}{\partial W1} = X$$

$$\frac{\partial L}{\partial W1} = (A2 - Y) * W2 * d * X$$

Similarly ,

$$\frac{\partial L}{\partial b1} = \frac{\partial L}{\partial A2} * \frac{\partial A2}{\partial Z2} * \frac{\partial Z2}{\partial A1} * \frac{\partial A1}{\partial Z1} * \frac{\partial Z1}{\partial b1}$$
$$= \sum (A2 - Y) * d * W2$$

Comparison of the performence of two models :

MLP Model trained by augmented data performed better than the model trained by unaugmented data by some margin. We achieved a accuracy 79.6 and 77.5 by models trained on augmented and unaugmented data respectively.

- augmented data set is created by combining original data and augmented data , hence it has more feature to offer to feature extraction algorithm as compared to original dataset
- As augmented dataset is combination of original data and augmented data, it was less prone to overfit as compared to original data.
- Model trained by augmented data took more epoch and computational time to train, hence had more time to learn the features.