ENPM703- Assignment-1

Part4: Two Layer Neural Network

Neural Network

Neural Network consists of layers of interconnected nodes where each node performs a mathematical operation. In a 2-layer neural network, there are two layers of weights: one from the input layer to a hidden layer, and one from the hidden layer to the output layer. The network learns by adjusting these weights to minimize the error between predicted and the ground truth data. Neural networks are powerful for tasks like image classification, speech recognition, since they are able to perform non-linear mappings from input to output.

Forward Pass

In forward pass input data passes through the network layers to produce the output prediction. In first layer of NN, input data is transformed by multiplying it with the weights and adding biases at each layer, followed by applying activation functions like ReLU. In the second layer the output is computed layer by layer, moving from the input to the output layer, and is used to calculate the loss through loss functions like softmax loss.

Back Propagation

Backpropagation is the process of computing gradients for each weight in the network and used to minimize the loss function during training. Here we compute the gradient of the loss with respect to the weights using the chain rule.

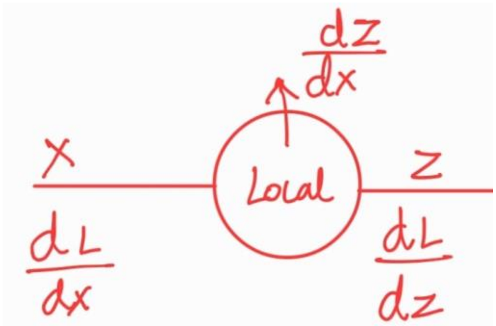
The backward pass starts from the output layer and propagates back through each layer, updating the weights proportional to the negative gradient. Thus, the weights are updates at each iteration to mimic the features of each class.

By using the chain rule, we get,

Downstream gradient(dL/dx) =

Local gradient(dz/dx) *

Upstream gradient(dL/dz)



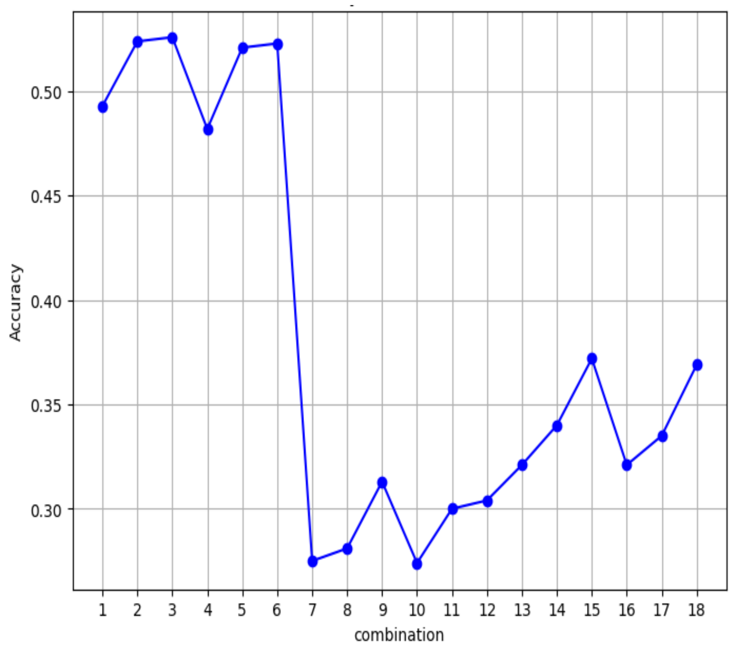
Backprop with Matrices

Since all the weights and the gradient of loss are all matrices, we are dealing with both upstream and downstream gradients as matrices. Since matrix multiplication is a computationally expensive task, we find patterns in these matrix multiplications and implement those patterns.

Thus, we boil down to,

1. **Downstream gradient(dL/dx) = Upstream gradient(dL/dz) * Transpose of weight(W^T)**
2. **Downstream gradient(dL/dw) = Transpose of weight(X^T) * Upstream gradient(dL/dz)**

Hyperparameters Tuning



1: ['lr:0.001', 'reg:1e-05', 'hid:32'],
2: ['lr:0.001', 'reg:1e-05', 'hid:64'],
3: ['lr:0.001', 'reg:1e-05', 'hid:128'],
4: ['lr:0.001', 'reg:1e-06', 'hid:32'],
5: ['lr:0.001', 'reg:1e-06', 'hid:64'],
6: ['lr:0.001', 'reg:1e-06', 'hid:128'],
7: ['lr:1e-05', 'reg:1e-05', 'hid:32'],
8: ['lr:1e-05', 'reg:1e-05', 'hid:64'],
9: ['lr:1e-05', 'reg:1e-05', 'hid:128'],
10: ['lr:1e-05', 'reg:1e-06', 'hid:32'],
11: ['lr:1e-05', 'reg:1e-06', 'hid:64'],
12: ['lr:1e-05', 'reg:1e-06', 'hid:128'],
13: ['lr:2e-05', 'reg:1e-05', 'hid:32'],
14: ['lr:2e-05', 'reg:1e-05', 'hid:64'],
15: ['lr:2e-05', 'reg:1e-05', 'hid:128'],
16: ['lr:2e-05', 'reg:1e-06', 'hid:32'],
17: ['lr:2e-05', 'reg:1e-06', 'hid:64'],
18: ['lr:2e-05', 'reg:1e-06', 'hid:128']

Learning Rate: This controls how fast the model updates its weights during training. If learning rate is too small, the model will be very slow and if its is too high it may result in overshooting.

Regularization Strength: This controls how the weights should be with respect to the image features. If this is too small, the model might overfit and if it is too high, the model might be generalized and result in underfitting

No. Hidden Layer: This is used in neural network to learn complex patterns from the data. If it is too high, the model might overfit and if it is too low the model might be inefficient,

Why NN is Better?

Neural networks outperform traditional machine learning models because they can automatically learn complex features from raw data. They are better at handling high dimensional and non-linear data, and they can generalize better when trained with large datasets.

Neural networks also enable deep learning, where multiple hidden layers allow for learning more abstract and hierarchical patterns, leading to state-of-the-art results in fields like computer vision and natural language processing.