Project Extension 1: Neural Network Dropout

Tony Maricic
Alison Reed

December 2023

1 High Level Overview of Our Extension

We chose to add dropout to our existing basic neural network implementation for this extension. The intuition behind this extension is that as a neural network grows in size and depth, we start to see the model over-fit the training data. What this means is that the model will get to a point of low bias and high variance, where it will not perform well on test data. To prevent this, we can "drop" random nodes in the input and hidden layers of the model with each iteration. We first choose a dropout rate; indicated as the variable p. This is a value between 0-1 that represents the percentage of nodes per layer that will be dropped. We then propagate forward, "ignoring" those nodes(and any weights associated with them) and calculate our output. Lastly, we do back-propagation to update our weights and biases while still ignoring these dropped nodes. This prevents the model from trying to fit the training data exactly by only giving it random nodes at a time, therefore mitigating the problem of over-fitting.

2 Breaking Down The Code

Our code applies dropout to nodes by using a "dropout mask". This is a vector of 1s and 0s by which we multiply our activation values to zero out the activation values of dropped nodes. To avoid over-weighting the remaining nodes in our network, we divide our dropout mask by (1-p).
First, we apply this dropout mask to our input layer to zero out "p" percent of the nodes. We then forward propagate through the network. We multiply each hidden activation layer by a dropout mask. Note, we do not apply dropout to the output layer as it produces our predictions. Next, we calculate the deltas for the gradient descent step in backwards propagation. The calculation of the output layer's delta is the same as that in the original implementation. To account for the dropped nodes in the hidden and input layers in backwards propagation when calculating deltas, we multiply the dropout mask at that layer by the product of $W^T \delta^{i+1}$ and then proceed as in the original implementation, multiplying the result by f'(z). The gradient descent step does not change.

3 Results

Dataset	With/Without Dropout	Num Iterations	Alpha	Activation	Input Layer p value	Hidden Layer p value	Accuracy
Original Dataset	Without	3,000	.25	Sigmoid	N/A	N/A	~85%
Original Dataset	With	10,000	.5	Sigmoid	.2	.2	~95%
New Dataset	Without	3,000	.25	Sigmoid	N/A	N/A	~16%
New Dataset	With	3,000	.5	Sigmoid	.2	.2	15%

Figure 1: The prediction accuracy on the MNIST digits data-set increased from 85% to 95% when we applied dropout to the neural network. The neural network with and without dropout did not perform well on the KMNIST Japanese characters data-set.

MNIST Digits Data-set Hyper-parameters:
The neural network from the homework implementation is on the smaller side with 64 input nodes and only one hidden layer. When we applied dropout to this network using the same hyper-parameters as the homework implementation (3000 iterations, alpha = 0.25), performance decreased because we were underfitting the data - removing nodes from an already small network. To counteract this, we chose a low dropout rate .2 (20 percent of nodes per layer) and increased the number of iterations to 10,000. Dropout is a form of regularization - it prevents the model from making complex adaptations to perfectly fit the training set, but in doing so, it also reduces the capacity of the neural network to learn. The neural net learns more slowly because only a subset of nodes are being trained in any given iteration. Thus, we increased our alpha/learning rate from 0.25 to 0.5 when we applied dropout.

KMNIST Japanese Characters Data-set Hyper-parameters: The images in the KMNIST Japanese character data-set are much larger than the images in the digits data-set. The Japanese character images are 28x28 so the input layer of the neural network has 784 nodes. Thus, training the neural network on this data-set took much longer and prevented extensive hyper-parameter experimentation. We believe that the combination of having so many features, only one hidden layer and 3000 iterations led to an under-fit model which resulted in poor prediction accuracy. Ideally for this data-set, we would have more hidden layers, more iterations, and a higher dropout rate. In general, dropout performs better on larger neural networks with more hidden layers and a higher dropout rate (typically between 0.2 and 0.5).