Jared Amaral

BJ Bae

28 March 2023

CNN and Regularization

For this assignment, we’ve created a Convolutional Neural Network that has the ability to differentiate between 10 different classes of objects, namely automobiles, cats, dogs, ships, airplanes, birds, deer, frogs, horses, and trucks. To prevent overfitting, we’ve applied different methods of regularization. Regularization penalizes the predictions of our network during training so that it can perform well on future datasets. This paper will discuss these different methods and the results from doing so.

The first time running the model, we experienced a low accuracy of 63.9%. The model was predicting the same class because we did not pre-process the data. Data preprocessing is important because it allows the model to work better by removing inconsistencies in the data. To do this, we center the data by calculating the dataset and subtracting it and then normalize the data, or “Divide each dimension by its standard deviation, after the data has been centered”(Convolutional Neural Networks in Practice). Standardizing the data before using the data as input to the CNN slightly hurt test loss by ~1 and test accuracy by ~1% from the standard model’s 1.908 loss and 63.9% test accuracy. This might have occurred because of excessive noise in our data.

Batch Normalization is a technique that standardizes the input for each mini-batch. This improves model learning and reduces the number of epochs to train the model. Batch Normalization increased the test accuracy to 70.7%. The training loss fell to very close to 0 as validation loss stayed around 1 signifying some overfitting in the model. This is reflective of how the model operates as it only remedies the variance in initialization of inputs rather than focusing on regularization and model generalization which is covered in the next section. The increase in model performance could be attributed to reduced noises through normalizing in batches rather than as a whole input, but this would have to be confirmed through further study.

The first regularization method we used was L2 Regularization (Ridge Regression). L2 Regularization adds a “squared magnitude” as a penalty to the loss function(L1 and L2 Regularization Methods). The problem with summing the squared weights is that if there are much larger weights, then that weight will contribute to most of the model complexity, therefore not generalizing the model. By using a regularization

$$\frac{1}{2}\lambda \sum_i \sum_j w_{i,j}^2.$$

rate *lambda*, we can lower weight values closer to 0 to reduce the complexity of the model. For our model, L2 Regularization resulted in a training accuracy of ~72% and a testing accuracy of 69.9% compared to the initial model evaluation of ~64% accuracy, which is an improvement off the initial model but slightly lower than the batch normalized model. A hypothesis that we tested was that combining both batch normalization and L2 regularization would increase model performance, because one would reduce noise and the other would limit model complexity. This turned out to be false using the same model structure. The test accuracy came out to be 67.5% with a test loss of 1.547, which is both less than the batch normalized and L2 regularized models.There might be some conflicts between the two methods and overfitting that might be contributing to lower performance that we aren’t fully sure of why.

The second regularization method we used for the network is L1 Regularization (Lasso Regression). In L1 Regularization, the regularization rate, *lambda*, works the same as L2 Regularization. The main difference between L1 and L2 is that, “Lasso

$$\sum_{i=1}^{n}(Y_i - \sum_{j=1}^{p} X_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

shrinks the less important feature’s coefficient to zero thus, removing some features altogether” (L1 and L2 Regularization Methods). With L1 Regularization, our model received a training accuracy of ~45% and a testing accuracy of 46.7%. The model was not overfitting to the training data, but the model performed worse in overall prediction accuracy. This outcome aligns with what the blogpost said about L2 being overall a better performing solution to regularization compared to L1.

Elastic net regularization is just a combination of both L1 and L2 regularization terms. This did even worse than the L1 regularization alone which was surprising because it is easy to assume that having two regularization terms is better than having one. The test accuracy was 44.5% but the training accuracy of ~45.4% also explains why this model might be underperforming. The regularization terms could limit the model’s performance by trying to generalize too much creating an slightly underfitting model.

Dropout is used to make the model more robust in finding different pathways of neuron activations to get to the same result without having to rely on a single path. It also reduces overfitting by making all the different neurons carry some workload of identifying certain features without having dead neurons that don’t contribute. The dropout helped the model achieve a test accuracy of 72.0% with training accuracies of ~91%. This was the best performing model so far and dropout really helps with using all the neurons of the neural network to generalize the data.

A similar technique is by combining dropout with max norm. Max norm is a way of capping the activation of a neuron to a certain value so the gradient does not explode. The constant that is used is around .3 or .4 according to the blogpost. The test accuracy of this model was 72.2%. This model, like the previous model, seems to have some overfitting.

For further exploration, we changed the kernel size of the previous model to (5,5) which increased the test accuracy to an even higher 74.6%. Since this was successful we increased the kernel size further to (10,10) but this decreased model performance down to 68.8% test accuracy. I ran this model again and the test accuracy went up by ~2%, but still less than the kernel size of (5,5). For the last model we added complexity to the neural network at the end of the convolution to have a structure of 1024-512-256-128, but the accuracy dipped slightly below 70% test accuracy with high overfitting.

References

https://towardsdatascience.com/convolutional-neural-networks-in-practice-406426c6c19a

https://towardsdatascience.com/l1-and-l2-regularization-methods-ce25e7fc831c

https://developers.google.com/machine-learning/crash-course/regularization-for-simplicity/lambda

https://machinelearningmastery.com/batch-normalization-for-training-of-deep-neural-networks/