Machine Learning

Yaffa Atkins

Research Reports

The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks:

The lottery ticket hypothesis staters that there exists a small subnetwork within a bigger network that can obtain accuracy similar to that of the original network within the same number of epochs. These trainable subnetworks which are uncovered through pruning are called winning tickets since they were initialized properly with weights and connections that capable of learning. Pruning, or eliminating unnecessary weights from neural networks can reduce storage space by ninety percent and decrease energy consumption without negatively impacting the model's final accuracy. The goal of the lottery ticket hypothesis is to initially train the optimal network rather than spend time reducing the big network to smaller networks.

Dense randomly initialized networks are easier to train than sparse networks because there are more possible subnetworks, meaning there are more possible winning tickets. Since winning tickets can be trained in isolation, training schemas can be designed to search for winning tickets and prune them early on. Additionally, architectures that are ideal for learning can be recognized based on properties seen in the winning tickets.

One shot pruning is when the network's connections are only pruned once. First the weights are randomly initialized, then the network is trained to find the optimal weights, next a percentage of the connections are pruned, and the remaining weights are reset to their initial value. What is left is a winning ticket. Iterative pruning has multiple iterations where only a portion of the total percent of pruned connections are pruned at each iteration. While one-shot pruning is less computationally expensive, iterative pruning finds smaller subnetworks that match the original networks accuracy, and the goal here is to identity the smallest possible winning tickets.

At the start of iterative pruning, the more the network is pruned, the faster it learns and the higher accuracy it reaches. But eventually the network is overly pruned, and learning begins to slow down until it reaches the speed of the original network. Winning tickets perform far worse and train slower when randomly initialized, so structure alone is not enough, the initialization is very important.

How Does Batch Normalization Help Optimization?

Normalization is when values that are on different scales are adjusted to all be on a common scale. The batch size refers to the number of samples that pass through the network at one time, so larger batch sizes mean faster training, but sometimes smaller batch sizes are preferred. Batch is the default normalization method used in most deep learning models but there is still a poor understanding of what exactly causes it to be so effective.

The most widely accepted explanation relates to internal covariate shift. ICS addresses the change of distribution of layer inputs caused by updates to previous layers, which can negatively impact training, and batch normalization reduces ICS. But there is little evidence to support this explanation. Internal covariant shift is the difference in the gradients over time and batch normalization has no strong contribution to reducing ICS, in fact, networks with batch normalization often show an increase in ICS. Reducing ICS does not contribute toward better network performance.

The cause of batch normalizations effectiveness is that since it normalizes the data, the gradients more reliable and predictable. This allows the algorithm to take larger steps without running into large changes in the loss landscape which enables the use of a broader range of learning rates and makes training faster. Batch normalization also reduces the demand for regularizing the data because the mean and variance calculated to perform the normalization are based on the values of the current batch. When a higher learning rate is used, performance of a learning model is much better with batch normalization compared to without batch normalization. Other normalization methods also have improved performance and smooth loss landscape, giving them similar advantages to batch normalization. Batch normalization requires large batch sizes for good results, but another method known as group normalization can solve this problem, since its performance is consistent. Group normalization is when one feature is normalized over several channels or layers, and it is often a preferred method over batch normalization.

Understanding and Utilizing Deep Neural Networks Trained with Noisy Labels:

Deep neural networks are remarkably accurate when trained with enough correct data labels. But often there are noisy labels and DNNs end up memorizing them during training and overfit the errors. Labeling extensive data correctly is expensive but training with too many errors severely decreases model performance. DNNs can fit noisy labels in training but the generalization error will still be large, even if the network is tested on the same noise, generalization occurs in the sense of distribution meaning predicted labels and real labels will have the same distribution even with noise. It is important to understand how noisy labels affect training to design practical models to deal with the noise.

A method was developed on top of the co-training strategy to train DNNs against noisy data. In the co-teaching strategy, two networks are trained simultaneously. Two mini batches draw from the noisy data set and each of the two networks select small loss samples and feed them to the other network. The number of small loss examples selected is set according to the noise ratio of the training data set, but this information is not available. Iterative Noisy Cross Validation (INCV) is used to select a subset of examples with smaller noise ratio resulting in more stable training and the noise ratio of the selected set can be estimated. An increase in the noise ratio leads to a severe decrease in co-teaching performance.

INCV helps increase the number of selected samples by randomly dividing the training data in half, training each half separately, identifying clean samples, removing samples with large categorical cross entropy loss, and performing cross validation where the union of the clean samples from both halves is selected as the final samples. If the model needs more samples, iterate through this algorithm. Then co-teaching takes full advantage of the identified samples to train DNNs against noisy data. To improve stability and test accuracy, let the two networks focus on the selected set for the first epochs then incorporate the candidate set.