Dropout – To prevent Neural Networks from Overfitting

Owais Ahmad
owais96_scs@jnu.ac.in
School of Computer and System Sciences,
Jawaharlal Nehru University, New Delhi, India

Abstract

Deep NNs with large number of parameters are powerful machine learning systems. But overfitting is a serious problem in these networks. These NNs are slow to use, making difficulties to deal with overfitting by combining the predictions of different large NNs at testing time. Dropout could be a technique to manage these problems. The idea behind is to randomly drop nodes (along with their incoming and outgoing links) from the NNs during training time. This prevents nodes from co-adapting excessively. During training, dropout samples from an oversized number of various thinned networks. Now during test time, it's easy to approximate the effect of averaging the predictions of these thinned networks by a simple use of single un-thinned network that has comparatively smaller weights. This method reduces over-fitting and provides major improvements over other regularization methods.

Keywords

Dropout, Deep, Dense, Neural Network, Backpropagation, Over-fitting, Regularization

Introduction

In Deep NNs there are multiple hidden layers and this makes them very complex models that may have very complicated relationships between their inputs and outputs. We have limited training data, however, many of them are complicated relationships and can be the results of sampling noise. If it's taken from the identical distribution, they're going to exist within the training set but not in real test data.

This ends up in overfitting with large NNs, however, the idea of averaging the outputs of many separately trained nodes is prohibitively expensive. Combining several models is most helpful when the individual models are different from one another and in order to make NNs models different, they should have either different architectures or be trained on different data. Training many types of architectures is hard because finding optimal hyperparameters for every architecture could be a daunting task and training each large network requires plenty of computation. Moreover, large networks normally require large amount of training data and there might not be enough data available to train various networks on different subsets of the data. Even if someone is able to train many different large networks, using all of them at test time is infeasible in applications where it's important to respond quickly. The term "Dropout" refers to dropping out units (visible and hidden) in a NNs. By dropping a unit out, it means temporarily removing it from the network, together with all its incoming and outgoing connections, as shown in Figure 1.

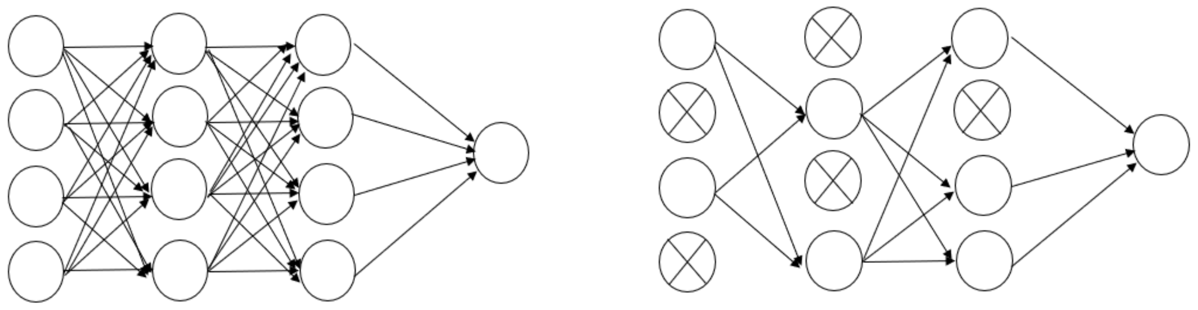

Figure 1

By dropping the nodes in neural network at training time stops the model to overfit on the training data and when applied on test data it gives faster and better results.

Here dropping the nodes means dropping the features randomly so as to make it efficient during Testing period.

Literature Survey

A study done by Wiesler and his team members found that overfitting can also be avoided by restricting the model size. This has the additional benefit of accelerating both training and recognition. The DNN size can be reduced by decreasing the number of layers or hidden nodes per layer. But typically, the best results are achieved with larger models in combination with early stopping **[1].** So, basically here I am trying to perform same thing.

Srivastava and Geoffrey Hilton at Google Brains, working on similar topic to prevent overfitting by dropping the nodes found out that dropout neural networks can be trained using stochastic gradient descent in a manner similar to standard neural nets. The only difference is that for each training case in a mini-batch **[2].** They try to sample a thinned network by dropping out some nodes in each layer with p probability and results were very good.

Noise learning is one important cause of overfitting. Xue Ying argued in his paper that logically noise reduction becomes one researching direction for overfitting inhibition **[3].** So, by dropping the nodes in input layer and subsequent hidden layer we can have more generalized model which is much better.

Choosing number of nodes for each layer will depend on problem NN is trying to solve, types of data network is dealing with, quality of data and some other parameters. Number of input and output nodes depends on training set in hand **[4].** The concept of dropout of nodes can be found from these as well as we will drop the nodes in input layer. This can be also considered as feature selection but on only training data

Mechanism

The choice of which units to drop is random. Within the simplest case, each unit is retained with a hard and fast probability p independent of other units, where p may be chosen with a validation set or can simply be set at 0.5, which seems to be near optimal for a large range of networks and tasks. For the input units, however, the optimal probability of retention is closer to 1 than to 0.5 as shown in Figure 2.

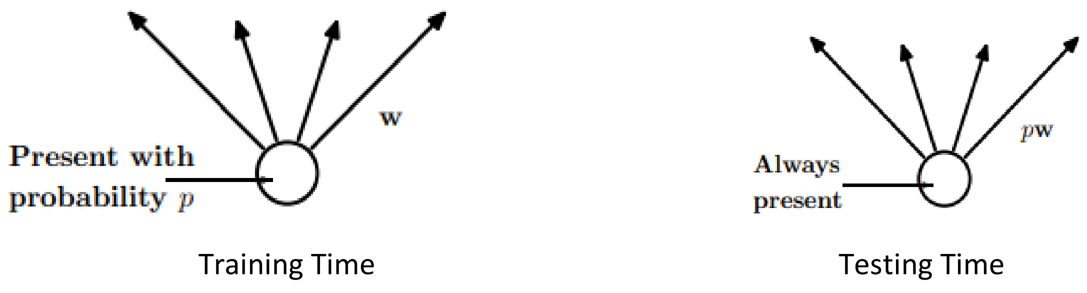

Figure 2

Here by dropping the nodes it will reduce the capacity of our NNs. If there are n nodes present in a layer and probability with which nodes are deleted is p then instead of n nodes only np nodes will be present after dropout and hence during training time nodes are preventing from co-adaption too much.

Description

Whenever we try to train highly dense neural network there is high chance that our model will overfit .That is for training data our accuracy was very high but for testing data or on unseen data our model was giving very bad accuracy because our model was overfitted during training .In other words we can say that we had low bias and high variance . So, to overcome this problem we can use the concept of Dropout. Here basically we drop some of nodes during training with p probability as explained above. Note we do this procedure at training time and not at testing time.

Now the model which we get from training can be used to test on our testing data. Here we will get more generalized model and our accuracy of prediction will be definitely high.

Analysis

There is a Drawback of dropout that it increases training time. A dropout network hardly takes 2-3 times longer to train than a standard NNs of the same architecture. A major cause behind this increase is because the parameter updates are very noisy. In each training case, it effectively tries to train a different random architecture. The gradients that are being computed are not gradients of the final architecture that will be used during test time. Therefore, it’s not surprising that training data takes a long time. However, it is likely that this method prevents overfitting. This creates a trade-off between training time and overfitting. So, with more training time, one can use high dropout and will cause less overfitting. However, a way to get advantages of some of the benefits of dropout without stochasticity is to marginalize the noise to get a regularized that does the same thing as the dropout procedure, in expectation

References

[1] Wiesler, Simon, Richard, Alexander, Schl¨uter, Ralf, and Ney, Hermann. Mean-normalized stochastic gradient for large-scale deep learning. In IEEE International Conference on Acoustics, Speech, and Signal Processing, pp. 180–184, Florence, Italy, May 2014.

[2] Srivastava, Nitish, Hinton, Geoffrey, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan. Dropout: A simple way to prevent NNss from overfitting. J. Mach. Learn. Res., 15(1):1929–1958, January 2014.

[3] Xue Ying. An Overview of Overfitting and its Solutions. IOP Conf. Series: Journal of Physics: Conf. Series 1168 (2019) 022022 IOP Publishing

[4] Mirza Cilimkovic ,Institute of Technology Blanchardstown NNss and Back Propagation Algorithm