Dropout: A Simple Way to Prevent Neural Networks from Overfitting

The paper focuses on the topic Dropout. The main focus was on Dropout and the idea of dropout is not limited to feed-forward neural nets. It can be more generally applied to graphical models such as Boltzmann Machines. In this paper the aim of the writer is to compare Restricted Boltzmann Machine and standard Restricted Boltzmann Machine and show that dropout RBMs are better than standard RBMs in certain respects. In Dropout for each training case in a mini-batch, we sample a thinned network by dropping out units. The noise provided by dropout then allows the optimization process to explore different regions of the weight space that would have otherwise been difficult to reach. As the learning rate decays, the optimization takes shorter steps, thereby doing less exploration and eventually settles into a minimum.

In Unsupervised learning Pretraining is an effective way of making use of unlabeled data. Pretraining followed by finetuning with backpropagation has been shown to give significant performance boosts over finetuning from random initializations in certain cases. Dropout can be applied to finetune nets that have been pretrained using these techniques.

After testing with different datasets with different parameters and droputs we get the outpur as

Neural Net Comparison: Dropout can be seen as a way of doing an equally-weighted averaging of exponentially many models with shared weights. On the other hand, Bayesian neural networks (Neal, 1996) are the proper way of doing model averaging over the space of neural network structures and parameters. In dropout, each model is weighted equally, whereas in a Bayesian neural network each model is weighted taking into account the prior and how well the model fits the data, which is the more correct approach. The task is to predict the occurrence of alternative splicing based on RNA features. Alternative splicing is a significant cause of cellular diversity in mammalian tissues. Predicting the occurrence of alternate splicing in certain tissues under different conditions is important for understanding many human diseases. Given the RNA features, the task is to predict the probability of three splicing related events that biologists care about. The evaluation metric is Code Quality which is a measure of the negative KL divergence between the target and the predicted probability distributions (higher is better).

Comparison with Standard Regularizers: Several regularization methods have been proposed for preventing overfitting in neural networks. These include L2 weight decay (more generally Tikhonov regularization (Tikhonov, 1943)), lasso (Tibshirani, 1996), KL-sparsity and max-norm regularization. Dropout can be seen as another way of regularizing neural networks. In this section we compare dropout with some of these regularization methods using the MNIST data set. The same network architecture (784-1024-1024-2048-10) with ReLUs was trained using stochastic gradient descent with different regularizations.

Salient features: In a standard neural network, the derivative received by each parameter tells it how it should change so the final loss function is reduced, given what all other units are doing. Therefore, units may change in a way that they fix up the mistakes of the other units. This may lead to complex co-adaptations. This in turn leads to overfitting because these co-adaptations do not generalize to unseen data. We hypothesize that for each hidden unit, dropout prevents co-adaptation by making the presence of other hidden units unreliable. A hidden unit cannot rely on other specific units to correct its mistakes. It must perform well in a wide variety of different contexts provided by the other hidden units.

Effect on Sparsity: As a side-effect of doing dropout, the activations of the hidden units become sparse, even when no sparsity inducing regularizers are present. Thus, dropout automatically leads to sparse representations. To observe this effect, we take the autoencoders trained in the previous section and look at the sparsity of hidden unit activations on a random mini-batch taken from the test set. In a good sparse model, there should only be a few highly activated units for any data case. Moreover, the average activation of any unit across data cases should be low.

Effect of Dropout Rate: Dropout has a tunable hyperparameter p (the probability of retaining a unit in the network). The comparison is done in two situations. 1. The number of hidden units is held constant. 2. The number of hidden units is changed so that the expected number of hidden units that will be retained after dropout is held constant. In the first case If the architecture is held constant, having a small p means very few units will turn on during training. It can be seen that this has led to underfitting since the training error is also high. We see that as p increases, the error goes down. It becomes flat when $0.4 \leq p \leq 0.8$ and then increases as p becomes close to 1. Another interesting setting is the second case in which the quantity pn is held constant

where n is the number of hidden units in any particular layer. This means that networks that have small p will have a large number of hidden units. Therefore, after applying dropout, the expected number of units that are present will be the same across different architectures.

Marginalizing Dropout: Dropout can be seen as a way of adding noise to the states of hidden units in a neural network. In this section, we explore the class of models that arise as a result of marginalizing this noise. These models can be seen as deterministic versions of dropout. In contrast to standard ("Monte-Carlo") dropout, these models do not need random bits and it is possible to get gradients for the marginalized loss functions. In this section, we briefly explore these models. Deterministic algorithms have been proposed that try to learn models that are robust to feature deletion at test time (Globerson and Roweis, 2006). Marginalization in the context of denoising autoencoders has been explored previously (Chen et al., 2012).

Conclusion: Dropout is a technique for improving neural networks by reducing overfitting. Standard backpropagation learning builds up brittle co-adaptations that work for the training data but do not generalize to unseen data. Random dropout breaks up these co-adaptations by making the presence of any particular hidden unit unreliable. This technique was found to improve the performance of neural nets in a wide variety of application domains including object classification, digit recognition, speech recognition, document classification and analysis of computational biology data. This suggests that dropout is a general technique and is not specific to any domain.