Dropout: A Simple Way to Prevent Neural Networks from Overfitting

Summary written by Nicholas Hinke
March 15, 2022

Summary: This paper provides a comprehensive overview of an extremely useful technique to prevent model overfitting, especially for deep networks with many parameters. As is often the case for deep, complex networks, overfitting continues to plague models of all architectures across a wide variety of application domains–particularly in data-constrained environments. Denoted as *dropout*, the technique presented in this paper aims to provide a general approach to combating overfitting by preventing neighboring neurons within the model from excessively co-adapting. Inspired by the then-recent work on sexual reproduction [5], addition of noise to hidden units [9, 10], stochastic and deterministic regularization techniques [3, 6, 8], adversarial dropout [1, 2], and Bayesian Neural Networks (BNNs) [11], the authors sought to provide an efficient and widely-applicable regularization technique that can be applied to many different network architectures. Moreover, this paper thoroughly examines the performance improvements gained on many different datasets when adding dropout to the existing state-of-the-art models. In fact, it was experimentally determined that adding dropout increased model performance on **all** of the tested datasets [7].

Approach: As noted above, the primary goal of dropout is to combat network overfitting by preventing excessive co-adaptation among neighboring neurons. In other words, the authors state that one of the driving forces behind model overfitting is due to nearby neurons making up for the mistakes of one another. Consequently, this behavior creates complex co-adaptations related to the training data that prohibit the model from performing optimally on new unseen test data. In order to prevent these co-adaptations from forming, the authors propose the idea of dropout, which entails randomly dropping some neurons (and their connections) at each iteration during training to create and train a "thinned network" [7].

Following this idea, a neural network with n neurons can be viewed as a set of 2^n potential thinned networks with shared weights. Thus, training a new thinned network at each training iteration can essentially be viewed as equivalent to finding a set of weights that will be globally applicable to all 2^n possible thinned networks. After training on the thinned networks to find the weights $\boldsymbol{w_{train}}$, a single network without any dropout is used at runtime for testing. In this single network, the weight parameters are computed as $\boldsymbol{w_{test}} = p\boldsymbol{w_{train}}$ where p is the probability that a given neuron is retained within each thinned network. This downscaling is done so that the output of the network during testing matches the expected output of the thinned networks during training [7].

The principal benefit of the approach used within dropout can be seen by considering another very successful regularization technique: model combination. Model combination is very effective at preventing overfitting because it incorporates the outputs of several different models, typically of varying architectures. However, model combination is not only extremely time-expensive when constructing, training, and validating each of the constituent networks, but it is also computationally-expensive at runtime due to the necessity of propagating the inputs through several different models. By contrast, the dropout technique retains the advantage of incorporating many different networks, but eliminates the need for designing additional models with potentially millions of more parameters [7].

Strengths: As stated within the paper, the key strength of dropout is its ability to make every neuron within a network more robust and less reliant on its neighbors. Moreover, the authors experimentally demonstrated the generality of this approach by applying it to wide a variety of datasets with several different network architectures (including CNNs and RBMs). In fact, the dropout technique even played a vital role in the success of the "AlexNet" network when it won the 2012 ILSVRC competition [4]. Additionally, while the two approaches aim to address very similar problems, the authors showed that using dropout was much faster and more scalable than using BNNs [7].

Weaknesses: Although in some ways dropout can be viewed as simplifying model training, its biggest drawback by far is the increased training time. In fact, model training times can be increased by as much as 2-3x when using dropout due to the noisy nature of the parameter updates. Additionally, while still effective in data-constrained environments, it was shown that the performance of a standard network with dropout is notably inferior to that of a BNN when trained on very small datasets [7, 11].

Reflections: As briefly discussed by the authors, it could be valuable to further investigate the performance of Gaussian dropout, or even dropout using other probability distributions. Additionally, it would quite beneficial to study methods to improve the performance of dropout when applied to text domains, as it was stated in the paper that performance increases were much less significant on text datasets than audio or visual datasets [7]. Finally, it would be very interesting to examine both the training and runtime efficiency of the dropout technique when implemented on today's modern computer hardware and GPUs.

References

[1] O. Dekel, O. Shamir, and L. Xiao. Learning to classify with missing and corrupted features. *Machine Learning*, 81:149–178, 2008.

[2] A. Globerson and S. T. Roweis. Nightmare at test time: robust learning by feature deletion. *Proceedings of the 23rd international conference on Machine learning*, 2006.

[3] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. C. Courville, and Y. Bengio. Maxout networks. In *ICML*, 2013.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60:84–90, 2012.

[5] A. Livnat, C. H. Papadimitriou, N. Pippenger, and M. W. Feldman. Sex, mixability, and modularity. *Proceedings of the National Academy of Sciences*, 107:1452–1457, 2010.

[6] N. Srebro and A. Shraibman. Rank, trace-norm and max-norm. In *COLT*, 2005.

[7] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15:1929–1958, 2014.

[8] L. van der Maaten, M. Chen, S. Tyree, and K. Q. Weinberger. Learning with marginalized corrupted features. In *ICML*, 2013.

[9] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML '08*, 2008.

[10] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.*, 11:3371–3408, 2010.

[11] H. Y. Xiong, Y. Barash, and B. J. Frey. Bayesian prediction of tissue-regulated splicing using rna sequence and cellular context. *Bioinformatics*, 27 18:2554–62, 2011.