Fundamental differences between Dropout and Weight Decay in Deep Networks

Fundamental Differences Between Dropout and Weight Decay in Deep Networks

David P. Helmbold
UC Santa Cruz
dph@soe.ucsc.edu

Philip M. Long
Sentient Technologies
phil.long@sentient.ai

March 8, 2016

Abstract

We study dropout and weight decay applied to deep networks with rectified linear units and the quadratic loss.

We show how using dropout in this context can be viewed as adding a regularization penalty term that grows exponentially with the depth of the network when the more traditional weight decay penalty grows polynomially. We then show how this difference affects the inductive bias of algorithms using one regularizer or the other: we describe a random source of data that dropout is unwilling to fit, but that is compatible with the inductive bias of weight decay. We also describe a source that is compatible with the inductive bias of dropout, but not weight decay.

We also show that, in contrast with the case of generalized linear models, when used with deep networks with rectified linear units and the quadratic loss, the regularization penalty of dropout (a) is not just a function of the independent variables, but also depends on the response variables, and (b) can be negative.

Finally, the dropout penalty can drive a learning algorithm to use negative weights even when trained with monotone training data.

1 Introduction

The 2012 ImageNet Large Scale Visual Recognition challenge was won by the University of Toronto team by a surprisingly large margin. In an invited talk at NIPS, Hinton [2012] credited the dropout training technique for much of their success. Dropout training is a variant of stochastic gradient descent (SGD) where, as each example is processed, the network is temporarily perturbed by randomly “dropping out” nodes of the network. The gradient calculation and weight updates are performed on the reduced network, and the dropped out nodes are then restored before the next SGD iteration. Since the ImageNet competition, dropout has been successfully applied to a variety of domains [Dahl, 2012, Deng et al., 2013, Dahl et al., 2013, Kalchbrenner et al., 2014, Chen and Manning, 2014], and is widely used [Schmidhuber, 2015, He et al., 2015, Szegedy et al., 2015]; for example, it is incorporated into popular packages such as Torch [Torch], Caffe [Caffe] and TensorFlow [TensorFlow]. Dropout has also sparked substantial research on related methods (e.g. Goodfellow et al. [2013], Wan et al. [2013]).

It is intriguing that crippling the network during training often leads to such dramatically improved results, and a number of possible explanations have been suggested. Hinton et al. [2012] suggest that dropout controls network complexity by restricting the ability to coadapt weights and illustrate how it appears to learn simpler functions at the second layer. Others [Baldi and Sadowski, 2013, Bachman et al., 2014] view dropout as an ensemble method combining the different network topologies resulting from the random deletion of nodes. Given this view, the participants in the ensemble have a special structure; in particular, they share many weights, and intuitively have less diversity than ensembles arising from techniques like boosting and bagging.

In this work, we examine the effect of dropout on the inductive bias of the learning algorithm. A match between dropout's inductive bias and some important applications could explain the success of dropout, and its popularity also motivates the study of its inductive bias. This paper uses formal analysis of dropout in multi-layer networks to uncover important properties of the method and identify fundamental differences between the biases of dropout and L_2-regularization (a.k.a. weight decay and Tikhonov regularization) in multi-layer networks, building on earlier work for single-layer networks [Wager et al., 2013, Helmbold and Long, 2015].

Our analysis is for multilayer neural networks with the square loss at the output node. The hidden layers use the popular rectified linear units [Nair and Hinton, 2010] outputting $\sigma(a) = \max(0, a)$ where a is the node's activation (the weighted sum of its inputs). We omit explicit bias inputs.

Given an (empirical) source distribution P and a particular network architecture, L_2 training uses stochastic gradient descent to try to find a set of weights $\mathcal{W}_{L_2}$ that minimizes the L_2 criterion

$$J_2(\mathcal{W}) = R(\mathcal{W}) + \frac{\lambda}{2}||\mathcal{W}||_2^2,$$

where $R(\mathcal{W})$ is the risk or expected loss with respect to distribution P. Dropout training is similar, but tries to find a set of weights $\mathcal{W}_D$ minimizing the criterion

$$J_D(\mathcal{W}) = \mathbf{E}_{\text{dropout patterns } \mathcal{R}}\; R(\mathcal{W}, \mathcal{R})$$

where $\mathcal{R}$ indicates the random set of nodes dropped out, and $R(\mathcal{W}, \mathcal{R})$ is the risk of the modified weights for that dropout pattern (see Section 2 for a complete description).

By studying the minimizer of the penalized risk, we abstract away sampling issues to focus on the inductive bias, as in [Breiman, 2004, Zhang, 2004, Bartlett et al., 2006, Long and Servedio, 2010, Helmbold and Long, 2015]. Alternatively, as mentioned above, P might be an empirical distribution over a training set, and our analysis addresses what kinds of training data dropout and weight decay are willing to fit.

Analogous to the L_2 penalty on the risk, $\frac{\lambda}{2}||\mathcal{W}||_2^2$, it is convenient to define the *dropout penalty* on weights $\mathcal{W}$ as $J_D(\mathcal{W}) - R(\mathcal{W})$. The dropout penalty indicates how much dropout training discriminates against a particular weight vector.

Our results support the intuition that, compared with L_2 regularization, the dropout criterion allows large weights, but discourages the computation of large values. In particular, we prove the following.

1. Unlike in the one-layer case, the dropout penalty depends on both the instances and the labels in the training set.
2. Unlike in the one-layer case, the dropout penalty is sometimes negative, and thus promotes (as opposed to penalizing) certain weight vectors. This can happen only when some of the weights (or inputs) are negative, and requires non-linearity in the hidden nodes.
3. Using the intuition that dropout penalizes the computation of large values, we exhibit a source with small inputs and a family of width n, depth d networks with small weights where the dropout penalty grows as n^{2d-1} while the L_2 regularizer grows only as $d^3 n$. (The exponential growth with d of the dropout penalty is reminiscent of some regularizers for deep networks studied by Neyshabur et al. [2015].)

Source	$R(\mathcal{W}_{\mathcal{D}})$	$R(\mathcal{W}_{L_2})$
P_1	$> 10^{51}$	< 1
P_2	$< \frac{1}{8}$	$= 1$

Table 1: Some upper and lower bounds on the risk of an optimizer $\mathcal{W}_{\mathcal{D}}$ of the dropout criterion and an optimizer $\mathcal{W}_{L_2}$ of the L_2 criterion. These are special cases of Theorems 2, 3, 10, and 15. The dropout probability q is kept constant at $1/3$ and $\lambda = 10^{-4}$ for both P_1 and P_2. The depth d of the network in P_1 is 20. In P_2, the width K of the network is 9999, the inputs ϵ are $10^{-4}/\sqrt{8}$. The ratios between the risks of dropout and weight decay on P_1 and P_2 may be made as large as desired, while keeping q and λ the same for both P_1 and P_2.

4. This exponential growth in the dropout penalty is exploited in the design of source distribution P_1 and network architecture where we prove that dropout training results in a model with extremely poor fit while standard L_2 regularization learns much more successfully.
5. On the other hand, L_2 regularization discriminates against large weights more than dropout training. Using this property we design a second source P_2 and accompanying network architecture where L_2 provably fails to learn accurately while optimizing the dropout criterion results in far more accurate hypothesis.
6. As a byproduct of this last analysis, we show how dropout training can prefer negative weights even when the function being learned is monotonic.

Special cases of some of our results are summarized in Table 1. With the L_2 regularization parameter λ and the dropout probability q (which controls the strength of the dropout regularization) fixed, we see dropout failing and weight decay succeeding for one distribution, and opposite behavior on another.

Related work

Previous formal analysis of the inductive bias of dropout has concentrated on the single-layer setting, where a single neuron combines the (potentially dropped-out) inputs. Wager et al. [2013] considered the case that the distribution of label y given feature vector $\mathbf{x}$ is a member of the exponential family and the log-loss is used to evaluate models. They pointed out that, in this situation, the criterion optimized by dropout can be decomposed into the original loss and a term that does not depend on the labels. They then gave approximations to this dropout regularizer and discussed its relationship with other regularizers.

Wager et al. [2014] considered dropout for learning topics modeled by a Poisson generative process. They exploited the conditional independence assumptions of the generative process to show that the excess risk of dropout training due to training set variation has a term that decays more rapidly than the straightforward empirical risk minimization, but also has a second additive term related to document length. They also discussed situations where the model learned by dropout has small bias.

Baldi and Sadowski [2014] analyzed dropout in linear networks, and showed how dropout can be approximated by normalized geometric means of subnetworks in the nonlinear case.

The impact of dropout (and its relative dropconnect) on generalization (roughly, how much dropout restricts the search space of the learner) was studied in Wan et al. [2013].

In the on-line learning with experts setting, Van Erven et al. [2014] showed that applying dropout in the on-line trials leads to algorithms that automatically adapt to the input sequence without requiring doubling or other parameter-tuning techniques.

2 Preliminaries

The network topology is fixed for training, and will generally be understood from context. We use $\mathcal{W}$ to denote a particular setting of the weights in the network and $\mathcal{W}(\mathbf{x})$ to denote the network's output on input $\mathbf{x}$ using weights $\mathcal{W}$. The hidden nodes are Rectified Linear Units (ReLUs), and the output node is linear.

We focus on square loss, so the loss of $\mathcal{W}$ on example $(\mathbf{x}, y)$ is $(\mathcal{W}(\mathbf{x}) - y)^2$. The ***risk*** is the expected loss with respect to a source distribution P, we denote the risk of $\mathcal{W}$ as $\underline{R_P(\mathcal{W})} \stackrel{\text{def}}{=} \mathbf{E}_{(\mathbf{x},y)\sim P}\left((\mathcal{W}(\mathbf{x}) - y)^2\right)$. The subscript will often be omitted when P is clear from the context.

The goal of L_2 training on a given source is to find a weight vector minimizing the ***L_2 criterion*** with regularization strength λ: $\underline{J_2(\mathcal{W})} \stackrel{\text{def}}{=} R(\mathcal{W}) + \frac{\lambda}{2}||\mathcal{W}||_2^2$. We use $\underline{\mathcal{W}_{L_2}}$ to denote a minimizer of this criterion. The L_2 penalty, $\frac{\lambda}{2}||\mathcal{W}||_2^2$, is non-negative. This is useful, for example, to bound the risk of a minimizer $\mathcal{W}_{L_2}$ of J_2, since $R(\mathcal{W}) \leq J_2(\mathcal{W})$; indeed, we use this.

Dropout training independently removes nodes in the network. In our analysis each non-output node is dropped out with the same probability $\underline{q}$, so $\underline{p} = 1 - q$ is the probability that a node is kept. (The output node is always kept; dropping it out has the effect of cancelling the training iteration.) When a node is dropped out, the node's output is set to 0. To compensate for this reduction, the values of the kept nodes are multiplied by $1/p$. With this compensation, the dropout can be viewed as injecting zero-mean additive noise at each non-output node [Wager et al., 2013]. [1] When dropout probability $q = 0$, all nodes will be kept all the time, and the dropout criterion becomes the risk, and as q increases so do the perturbations to the network. Therefore it is natural to interpret q as controlling the strength of the dropout penalty.

The ***dropout process*** is the collection of random choices, for each node in the network, of whether the node is kept or dropped out. A realization of the dropout process is a ***dropout pattern***, which is a boolean vector indicating the kept nodes. For a network $\mathcal{W}$, an input $\mathbf{x}$, and dropout pattern $\mathcal{R}$, let $\underline{\mathcal{D}(\mathcal{W}, \mathbf{x}, \mathcal{R})}$ be the output of $\mathcal{W}$ when nodes are dropped out or not following $\mathcal{R}$ (including the $1/p$ rescaling of kept nodes' outputs). The goal of dropout training on source P is to find a weight vector minimizing the ***dropout criterion*** for a given dropout probability:

$$\underline{J_D(\mathcal{W})} \stackrel{\text{def}}{=} \mathbf{E}_{\mathcal{R}}\mathbf{E}_{(\mathbf{x},y)\sim P}\left((\mathcal{D}(\mathcal{W}, \mathbf{x}, \mathcal{R}) - y)^2\right).$$

This criterion is equivalent to the risk of the dropout-modified network, and we use $\mathcal{W}_D$ to denote a minimizer of it. Since the selection of the dropout pattern and example from the source are independent, the order of the two expectations can be swapped, yielding

$$J_D(\mathcal{W}) = \mathbf{E}_{(\mathbf{x},y)\sim P}\mathbf{E}_{\mathcal{R}}\left((\mathcal{D}(\mathcal{W}, \mathbf{x}, \mathcal{R}) - y)^2\right).$$

This motivates the study of the dropout criterion on individual examples as the dropout criterion for a source distribution is just the expectation of criteria for single examples.

Consider now the example in Figure 1. The weight parameter $\mathcal{W}$ is the all-1 vector. $\mathcal{W}(1, -1) = 0$ as each hidden node computes 0. Each dropout pattern indicates the subset of the four lower nodes to be kept, and when $q = p = 1/2$ each subset is equally likely to be kept. If $\mathcal{R}$ is the dropout pattern where input x_2 is dropped and the other nodes are kept, then the network computes $\mathcal{D}(\mathcal{W}, (1, -1), \mathcal{R}) = 8$ (recall that when $p = 1/2$ the values of non-dropped out nodes are doubled). Only three dropout patterns produce a non-zero output, so when source P is concentrated on the example $\mathbf{x} = (1, -1), y = 8$ the dropout criterion is:

$$J_D(\mathcal{W}) = \frac{1}{16}(8-8)^2 + \frac{2}{16}(4-8)^2 + \frac{13}{16}(0-8)^2 = 54. \tag{1}$$

[1] Some authors use a similar adjustment where the weights are scaled down at prediction time instead of inflating the kept nodes' outputs at training time.

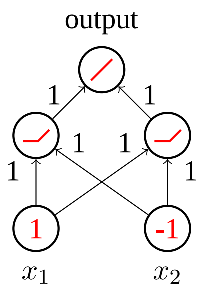

Figure 1: A network and weights where the dropout penalty is negative.

As mentioned in the introduction, the *dropout penalty* of a weight vector for a given a source and dropout probability is the amount that the dropout criterion exceeds the risk, $J_D(\mathcal{W}) - R(\mathcal{W})$. Wager et al. [2013] show that for 1-layer generalized linear models, the dropout penalty is non-negative.

Since $\mathcal{W}(1, -1) = 0$, we have $R(\mathcal{W}) = 64$, and the dropout penalty is negative in our example. In Section 6, we give a necessary condition for this.

Appendix A contains the notation used throughout the paper.

3 Growth of the dropout penalty as a function of network depth

In this section, we show that the dropout penalty can grow exponentially in the depth d even when the size of individual weights remains constant.

The network architecture is d layers of n nodes (counting the input layer, but not the output, see Figure 2) with each layer completely connected to the previous one, and the top layer connected to the output node. Fix the dropout probability at $1/2$.

Theorem 1 *There are weights $\mathcal{W}$ for a network of depth d and width n, and a source P such that (a) every weight has magnitude at most one, but (b) the dropout penalty is at least $\frac{n^{2d}}{n+1}$ which is $\Omega(n^{2d-1})$.*

Proof: The weights $\mathcal{W}$ are all set to 1, and the source distribution over examples is concentrated on the single example $(\mathbf{x}, y) = (\underbrace{(1, 1, ..., 1)}_{n \text{ ones}}, n^d)$ (see Figure 2).

It is a simple induction to show that, for these weights and inputs, the network outputs n^d, and has zero square loss (since $\mathcal{W}(\mathbf{x}) = n^d = y$).

Consider now dropout on this network with dropout probability $1/2$. This is equivalent to changing all of the weights from one to two and, independently with probability $1/2$ replacing the computed output of each node with 0. For a fixed dropout pattern, each node on a given layer has the same weights, and receives the same (kept) inputs. Thus, the value computed at every node on the same layer is the same.

Let H be the (random) value computed at the n nodes in the final hidden layer (which is a function of the dropout in the lower layers), and let us condition on the event that $H = h$ for an arbitrary $h \geq 0$. Let $r_1, ..., r_n$ be the indicator variables for whether each node in the final layer is kept, so $\mathbf{P}(r_i = 1) = 1/2$. Since $h \geq 0$, the output $\hat{y} = 2h \sum_i r_i$.

Using a bias-variance decomposition,

$$\mathbf{E}((\hat{y} - n^d)^2) = (\mathbf{E}[\hat{y}] - n^d)^2 + \mathbf{Var}(\hat{y}) = (nh - n^d)^2 + \mathbf{Var}(\hat{y}).$$

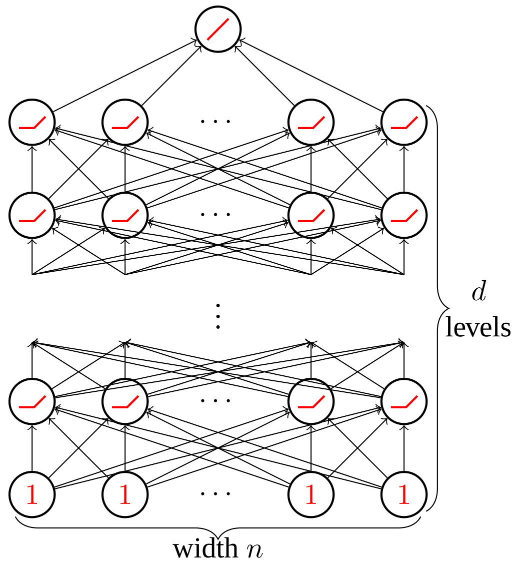

Figure 2: Network used for Section 3, each connection has weight 1.

Since $\sum_i r_i$ is binomially distributed, we have

$$\mathbf{E}((\hat{y} - n^d)^2) = (nh - n^d)^2 + 4h^2 \frac{n}{4} = (nh - n^d)^2 + h^2 n.$$

Using calculus, we can see that this is minimized when $h = \frac{n^d}{n+1}$, where it takes the value $\frac{n^{2d}}{n+1} = \Omega(n^{2d-1})$.

Since, for every value of h, after conditioning on $H = h$, $\mathbf{E}((\hat{y} - n^d)^2) \geq \frac{n^{2d}}{n+1}$, this inequality also holds when we average over the values of H. ■

4 When does the exponential penalty matter?

Adding the same n^d penalty to the risk of all weight settings for a width n, depth d network would not affect the behavior of the learning algorithm. So observing that the dropout penalty can grow exponentially in d does not necessarily imply that there is a substantial effect on the learned weights. This section contains an example where the dropout criterion's exponential dependence on the depth does have a substantial effect. In particular, we show that the dropout criterion forces the learner to use a highly inaccurate model compared to the model optimizing the L_2 criterion (weight decay).

Throughout this section we consider learning with the width-2 network architecture of Figure 2, which has d layers of 2 nodes each plus an output node. The source distribution used, which we denote $\underline{P_1}$, puts probability 1 on on the single example $\mathbf{x} = (1, 1)$ and $y = 2d^d$.

4.1 L_2 succeeds

In this subsection we show that that any minimizer $\mathcal{W}_{L_2}$ of the L_2 criterion (for source P_1) has small expected loss.

Theorem 2 *For any minimizer $\mathcal{W}_{L_2}$ of J_2, $R(\mathcal{W}_{L_2}) \leq \lambda d^3$.*

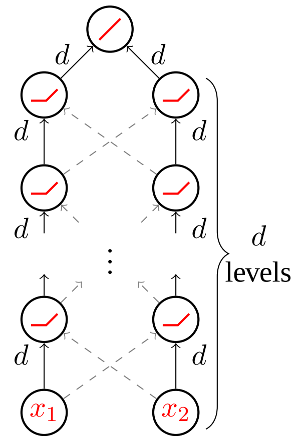

Figure 3: The network $\mathcal{W}$ in the proof of Theorem 2. The dashed-gray connections have weight 0.

Proof: Consider the network weights $\mathcal{W}$, pictured in Figure 3.

The connections between corresponding nodes on adjacent layers have weight d, as do the connections to the output node. The other "cross-connections" between layers have weight 0.

By construction, $\mathcal{W}(1,1) = 2d^d = y$, so $J_2(\mathcal{W}) = \frac{\lambda}{2}||\mathcal{W}||_2^2 = \lambda d^3$. This implies that any minimizer $\mathcal{W}_{L_2}$ of J_2 has $J_2(\mathcal{W}_{L_2}) \leq \lambda d^3$. Since $R(\mathcal{W}') \leq J_2(\mathcal{W}')$ for every $\mathcal{W}'$, the bound on $J_2(\mathcal{W}_{L_2})$ implies $R(\mathcal{W}_{L_2}) \leq \lambda d^3$. ■

4.2 Dropout fails

Here we prove that an algorithm minimizing $J_{\mathcal{D}}$ refuses to fit data drawn from P_1: any minimizer $\mathcal{W}_{\mathcal{D}}$ of the dropout criterion for P_1 has large risk.

Theorem 3 *If $((1+q)(1-q))^d \leq 1/8$ then any minimizer $\mathcal{W}_D$ of the dropout criterion has $R_{P_1}(\mathcal{W}_D) \geq \frac{d^{2d}}{2}$.*

The idea behind the proof is that any weights $\mathcal{W}$ such that $\mathcal{W}(1,1)$ is large enough to have small risk must also have a value of $J_D(\mathcal{W})$ too large to be a minimizer.

If dropout pattern $\mathcal{R}$ partitions the network so that the inputs are no longer connected to the outputs then every set of weights $\mathcal{W}$ computes 0, i.e. $\mathcal{D}(\mathcal{W}, \mathbf{x}, \mathcal{R}) = 0$. This is formalized in the following definition.

Definition 4 *A dropout pattern $\mathcal{R}$ cuts the network if every path from an input to the output contains a node dropped out in $\mathcal{R}$. Let CUT be the event that $\mathcal{R}$ cuts the network, and NOCUT be the event that $\mathcal{R}$ does not cut the network.*

Lemma 5 *$\mathcal{R}$ cuts the network if and only if it drops out both nodes on some layer.*

The probability that $\mathcal{R}$ does not cut the network is $\mathbf{P}(\text{NoCut}) = \left(1-(1-p)^2\right)^d = (p(2-p))^d$.

Proof: If both nodes on a layer are dropped out, then the network is cut. If no layer has this property then, since adjacent layers started completely connected, there will always be a path from some input to the output node. ■

When a dropout pattern cuts the network, all choices of the weights produce output 0 and loss $(2d^d)^2$ (for source P_1). This motivates the following simplification of $J_{\mathcal{D}}$ that allows us to focus on the NoCut case.

Definition 6 *In this section, let*
$\underline{\tilde{J}_{\mathcal{D}}(\mathcal{W})} = J_{\mathcal{D}}(\mathcal{W}) - \mathbf{P}(\text{Cut})(2d^d)^2 = \mathbf{E}_{\mathcal{R}}(\mathbb{1}_{\text{NoCut}}(\mathcal{R})\ (\mathcal{D}(\mathcal{W},(1,1),\mathcal{R}) - 2d^d)^2).$

Thus $\tilde{J}_{\mathcal{D}}(\mathcal{W})$ is the contribution to $J_{\mathcal{D}}(\mathcal{W})$ from those dropout patterns that do not cut the network.

Lemma 7 *$\tilde{J}_{\mathcal{D}}$ has the same minimizer(s) as $J_{\mathcal{D}}$.*

Proof: All weights yield the same loss when $\mathcal{R}$ cuts the network. ■

Next, we show that if $\mathcal{W}(1,1)$ is large enough to be even a poor approximation to $2d^d$, then $\tilde{J}_{\mathcal{D}}(\mathcal{W})$ is very large.

Lemma 8 *If $\mathcal{W}(1,1) \geq d^d$ and $p^{-d} \geq 8$ then $\tilde{J}_{\mathcal{D}}(\mathcal{W}) > \frac{d^{2d}}{2}$.*

Proof: If no node anywhere in the network is dropped out, then the value computed by each (non-output) node will be multiplied by $1/p$ by the dropout. These factors accumulate, so for an arbitrary node z, if j is the number of (non-output) nodes on paths from the inputs to z, the value computed by z on $(1,1)$ when dropout is applied is a factor $(1/p)^j$ greater than the value computed without dropout. In particular, the value of the output is $(1/p)^d\mathcal{W}(1,1)$. Since the probability that no node is dropped out is p^{2d},

$$\tilde{J}_{\mathcal{D}}(\mathcal{W}) \geq p^{2d}((1/p)^d\mathcal{W}(1,1) - 2d^d)^2.$$

If $\mathcal{W}(1,1) \geq d^d$ and $p^{-d} \geq 8$ then $p^{-d}\mathcal{W}(1,1) > 2d^d$, implying

$$\tilde{J}_{\mathcal{D}}(\mathcal{W}) \geq p^{2d}((p^{-d}d^d - 2d^d)^2 = p^{2d}\left(p^{-2d}d^{2d} - 4p^{-d}d^{2d} + 4d^{2d}\right).$$

This last expression is greater than $\frac{1}{2}d^{2d}$ whenever $p^{-d} \geq 8$, completing the proof. ■

Lemma 9 *Under the assumptions of Theorem 3, the all-zero weight vector $\mathcal{Z}$ has $\tilde{J}_{\mathcal{D}}(\mathcal{Z}) \leq d^{2d}/2$.*

Proof: We have

$$\begin{aligned}\tilde{J}_{\mathcal{D}}(\mathcal{Z}) &= \mathbf{Pr}(\text{NoCut})\ (0 - 2d^d)^2 \\ &= (p(2-p))^d\,4d^{2d}.\end{aligned}$$

The theorem's condition on q implies that $(p(2-p))^d \leq 1/8$, completing the proof. ■

Proof of Theorem 3 : When $((1+q)(1-q))^d \leq 1/8$ then $p^d = (1-q)^d$ is also at most $1/8$. Combining Lemmas 8 and 9 implies $\mathcal{W}_{\mathcal{D}}(1,1) < d^d$ (since otherwise $\tilde{J}_{\mathcal{D}}(\mathcal{W}_{\mathcal{D}})$ for the minimizing $\mathcal{W}_{\mathcal{D}}$ would be larger than $\tilde{J}_{\mathcal{D}}(\mathcal{Z})$). Thus $R(\mathcal{W}_{\mathcal{D}}) > (d^d - 2d^d)^2 = d^{2d}$, completing the proof. ■

5 A source more compatible with dropout than L_2

Here we give a source and network topology that is more compatible with the inductive bias of dropout than L_2.

The network topology has an output node and a single hidden layer of K rectified linear units. The output node is connected only to the hidden units. For $1 \leq k \leq K$ and $1 \leq i \leq 2$, let v_{ki} be the weight from input i to hidden node k, and w_k be the weight on the connection from hidden node k to the output. (See Figure 4 for one setting of weights for this topology.)

The source P_2 for this network is concentrated on the single example $\mathbf{x} = (\epsilon, \epsilon)$ and $y = 1$.

5.1 L_2 fails

The following is the main result of this subsection.

Theorem 10 *Let weights $\mathcal{W}_{L_2}$ be any minimizer of J_2 for source P_2. If $\lambda \geq \sqrt{8}\epsilon$, then $R_{P_2}(\mathcal{W}_{L_2}) = 1$, and if $\lambda < \sqrt{8}\epsilon$ then $R_{P_2}(\mathcal{W}_{L_2}) = \frac{\lambda^2}{8\epsilon^2}$.*

With respect to a minimizer $\mathcal{W}_{L_2}$, let v_{ki}^* denote the weights at the hidden nodes and w_k^* denote the weights at the output node. We begin by proving some lemmas.

Our first lemma collects together some properties of optimal $\mathcal{W}_{L_2}$.

Lemma 11 *Any optimal $\mathcal{W}_{L_2} = (V^*, \mathbf{w}^*)$ satisfies the following conditions:*
(a) for each hidden node $k \in \{1, \ldots, K\}$, $v_{k1}^* = v_{k2}^*$;
(b) $\mathcal{W}_{L_2}(\epsilon, \epsilon) \leq 1$;
(c) $\mathcal{W}_{L_2}$ maximizes $\mathcal{W}(\epsilon, \epsilon)$ among those weight vectors with $||\mathcal{W}||_2 = ||\mathcal{W}_{L_2}||_2$;
(d) all weights of $\mathcal{W}_{L_2}$ are non-negative.

Proof: Most parts are proven by contradicting the optimality of $\mathcal{W}_{L_2}$.

(a) If $v_{k1}^* \neq v_{k2}^*$, then replacing each of them with their average results in a set of weights computing the same values at all nodes, but with a smaller L_2 penalty.

(b) If $\mathcal{W}_{L_2}(\epsilon, \epsilon) > 1$ then scaling down the output weights would reduce both parts of the L_2 criterion.

(c) Let $y_{\max}$ be the maximum of $\mathcal{W}(\epsilon, \epsilon)$, from among networks with $||\mathcal{W}||_2 = ||\mathcal{W}_{L_2}||_2$. If $y_{\max} > 1$, then we could improve on $J_2(\mathcal{W}_{L_2})$ by scaling down the weights in the output layer of $\mathcal{W}$ until the resulting weights $\widetilde{\mathcal{W}}$ had $\widetilde{\mathcal{W}}(\epsilon, \epsilon) = 1$. This $\widetilde{\mathcal{W}}$ would have zero error, and a smaller norm that $\mathcal{W}_{L_2}$.

If $y_{\max} \leq 1$, then any $\mathcal{W}$ with $||\mathcal{W}||_2 = ||\mathcal{W}_{L_2}||_2$ and $\mathcal{W}_{L_2}(\epsilon, \epsilon) < \mathcal{W}(\epsilon, \epsilon) \leq 1$, would have smaller error but the same penalty, and therefore $J_2(\mathcal{W}) < J_2(\mathcal{W}_{L_2})$.

(d) If any hidden node k has a negative v_{k1} or v_{k2} then both are negative by (a). Replacing both weights by 0 reduces the L_2 penalty without changing the network's output on P_2.

If any hidden node k has a negative w_k then setting w_k to 0 reduces the L_2 penalty without decreasing the network's output. Scaling up the output weights so that the modified weights $\mathcal{W}$ have $||\mathcal{W}||_2 = ||\mathcal{W}_{L_2}||_2$ will then increase the network's output, contradicting (c). ■

This allows us to restrict our attention to network weights $\mathcal{W}$ where for each k we have $v_{k1} = v_{k2}$. We use v_{k1} to denote the shared value of these weights.

Informally, parts (b) and (c) of Lemma 11 engender a view of the learner straining against the yolk of the L_2 penalty to produce a large enough output on (ϵ, ϵ). This motivates us to ask how large $\mathcal{W}(\epsilon, \epsilon)$ can be, for a given value of $||\mathcal{W}||_2^2$. The following definition captures how much each hidden node contributes to each of these quantities.

Definition 12 *The contribution of hidden node k to the activation at the output node is $2\epsilon w_k v_{k1}$, and the contribution to the L_2 penalty from these weights is $\frac{\lambda}{2}\left(2v_{k1}^2 + w_k^2\right)$.*

We now bound the contribution to the activation in terms of the contribution to the L_2 penalty. Note that as the L_2 "budget" increases, so does the the maximum possible contribution to the output node's activation.

Lemma 13 *If $w_k^2 + v_{k1}^2 + v_{k2}^2 = B$ then hidden node k's contribution to the output node's activation is maximized when $w_k = \sqrt{B/2}$ and $v_{k1} = v_{k2} = \sqrt{B}/2$, where it achieves the value $\frac{\epsilon B}{\sqrt{2}}$.*

Proof: Since $2v_{k1}^2 + w_k^2 = B$, we have $w_k = \sqrt{B - 2v_{k1}^2}$, so the contribution to the activation can be re-written as $2\epsilon v_{k1}\sqrt{B - 2v_{k1}^2}$. Taking the derivative with respect to v_{k1}, and solving, we get $v_{k1} = \pm\sqrt{B}/2$ and the maximum is at the positive solution (when v_{k1} is positive). When $v_{k1} = \sqrt{B}/2$ we have $w_k = \sqrt{B/2}$ and thus the maximum contribution of the hidden node is

$$2\epsilon\sqrt{B/2}\left(\sqrt{B}/2\right) = \frac{\epsilon B}{\sqrt{2}}.$$

■

Lemma 13 immediately implies the following.

Lemma 14 *The maximum of $\mathcal{W}(\epsilon, \epsilon)$, subject to $||\mathcal{W}||_2^2 \le A$, is $\epsilon A/\sqrt{2}$.*

Proof: When maximized, the contribution of each hidden node to the activation at the output is $\epsilon/\sqrt{2}$ times the hidden node's contribution to the sum of squared-weights. Since each weight in $\mathcal{W}$ is used in exactly one hidden node's contribution to the output node's activation, this completes the proof. ■

Note that this bound is independent of K, the number of hidden units.

Proof (of Theorem 10): Combining Lemma 11 and Lemma 14, $\mathcal{W}_{L_2}$ minimizes

$$\left(\frac{\epsilon||\mathcal{W}||_2^2}{\sqrt{2}} - 1\right)^2 + \frac{\lambda}{2}||\mathcal{W}||_2^2.$$

Its derivative with respect to $||\mathcal{W}||_2^2$ is

$$\sqrt{2}\,\epsilon\left(\frac{\epsilon||\mathcal{W}||_2^2}{\sqrt{2}} - 1\right) + \frac{\lambda}{2}.$$

If $\sqrt{2}\,\epsilon\left(\frac{\epsilon||\mathcal{W}||_2^2}{\sqrt{2}} - 1\right) + \lambda/2 > 0$ whenever $||\mathcal{W}||_2^2 \ge 0$ (i.e. $\lambda > \sqrt{8}\,\epsilon$), then the L_2 criterion is minimized when $||\mathcal{W}||_2^2 = 0$, where it takes the value 1.

Otherwise, setting this derivative to 0 and solving for $||\mathcal{W}||_2^2$ gives us that the minimum of the criterion occurs when

$$||\mathcal{W}||_2^2 = \frac{\sqrt{2}}{\epsilon} - \frac{\lambda}{2\epsilon^2}.$$

Evaluating the risk using Lemma 14:

$$R(\mathcal{W}_{L_2}) = \left(\frac{\epsilon}{\sqrt{2}}\left(\frac{\sqrt{2}}{\epsilon} - \frac{\lambda}{2\epsilon^2}\right) - 1\right)^2 = \left(\frac{\lambda}{2\sqrt{2}\,\epsilon}\right)^2 = \frac{\lambda^2}{8\epsilon^2}.$$

for the weights minimizing the L_2 criterion in this case.

Thus, overall the risk of the minimizer of the L_2 criterion is $\min\left\{1, \frac{\lambda^2}{8\epsilon^2}\right\}$. ■

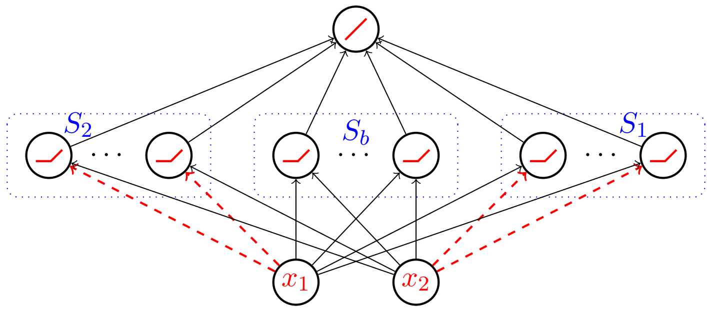

Figure 4: The network weights $\mathcal{U}$ from the proof of Theorem 15. All connections to the output node have weight 1, while the black connections to the hidden nodes have weight $\frac{3p}{2\epsilon K}$ and the red dashed connections have a larger-magnitude negative weight, $\frac{-3p}{\epsilon K}$. Each of the three sets of hidden nodes that share weights has $K/3$ members.

5.2 Dropout succeeds

Next, we show that dropout succeeds at learning P_2 with the same architecture.

Theorem 15 *The risk of any optimizer $\mathcal{W}_\mathcal{D}$ of the dropout criterion satisfies $E_{(\mathbf{x},y)\sim P_2}(\mathcal{W}_\mathcal{D}(x) - y)^2) \leq \left(q + \sqrt{\frac{3q}{\widetilde{K}}}\right)^2$ where $\widetilde{K} = 3\lfloor K/3 \rfloor$. Therefore, as K gets large, $E_{(\mathbf{x},y)\sim P_2}((\mathcal{W}_\mathcal{D}(x) - y)^2) \leq q^2(1 + o(1))$.*

Our proof of Theorem 15 will use some lemmas.

Let $\underline{\mathbf{s}} \in \{0,1\}^2$ be the indicator variables for keeping the input nodes, and let $\underline{\mathbf{r}} \in \{0,1\}^K$ be the indicator variables for keeping the hidden nodes.

The first step is to define a slightly simpler criterion that is equivalent to $J_\mathcal{D}(\mathcal{W})$, but focussing on the case where $\mathbf{s} \neq (0,0)$.

Definition 16 *In this section we re-define $\underline{\tilde{J}_\mathcal{D}(\mathcal{W})}$ to be*

$$\mathbf{E}_{\mathbf{s},\mathbf{r}}[(\mathcal{D}(\mathcal{W}, (\epsilon,\epsilon), (\mathbf{s},\mathbf{r})) - 1)^2] - \mathbf{Pr}[\mathbf{s} = (0,0)] = \mathbf{E}_{\mathbf{s},\mathbf{r}}[\mathbb{1}_{\mathbf{s}\neq(0,0)}(\mathcal{D}(\mathcal{W}, (\epsilon,\epsilon), (\mathbf{s},\mathbf{r})) - 1)^2].$$

Lemma 17 *$\tilde{J}_\mathcal{D}$ has the same minimizer as $J_\mathcal{D}$.*

Proof: All networks produce the same output on $(0,0)$, so they also produce the same output (and same loss) when both inputs are dropped out. ■

To simplify the presentation, assume that K is a multiple of three.

Our analysis will use weights $\mathcal{U}$ that we will define momentarily. We will not show that $\mathcal{U}$ is optimal, but we will use its analysis to bound the (non-dropout) risk of any dropout-optimal $\mathcal{W}_\mathcal{D}$.

Definition 18 *The network $\mathcal{U}$ is shown in Figure 4. It has three sets of $K/3$ hidden nodes: S_1, S_b, S_2. (It may be helpful to think of "b" here as standing for "both".)*

All weights on connections from the hidden nodes to the output are 1.

For $M = \frac{3p}{2\epsilon K}$, each hidden $j \in S_1$ has weight M on the connection from input 1, and weight $-2M$ on the connection from input 2.

S_2 has these weights reversed: Each hidden $j \in S_2$ has weight M on the connection from input 2, and weight $-2M$ on the connection from input 1.

Members of S_b have weights M from both inputs.

Whenever at least one input value is kept, then this construction guarantees that $\mathcal{U}$'s output will be unbiased when viewed as a function of the (random) hidden-node dropouts.

Lemma 19 *If* $\mathbf{s} \neq (0,0)$, *then* $\mathbf{E_r}(\mathcal{D}(\mathcal{U}, (\epsilon, \epsilon), (\mathbf{s}, \mathbf{r}))) = 1$.

Proof: If $\mathbf{s} = (1,1)$, then, since the negative weight to each input of S_1 is larger, each node in S_1 evaluates to 0. Similarly, each node in S_2 evaluates to 0. The $K/3$ nodes in S_b each evaluate to $\frac{3}{K}$, since each input contributes $\epsilon \times (1/p) \times \frac{3p}{2\epsilon K} = \frac{3}{2K}$. Recall that $\mathbf{r}/p$ combines the dropout-or-keep choices at the hidden nodes and the subsequent rescaling. Since each component of $\mathbf{r}/p$ has an expectation of 1, this completes the proof in the case $\mathbf{s} = (1,1)$.

Since the remaining two cases are symmetric, we consider just the case $\mathbf{s} = (1,0)$. In this case, each node in S_1 gets a contribution $\epsilon \times (1/p) \times \frac{3p}{2\epsilon K} = \frac{3}{2K}$ from input 1, and 0 from input two. Since the second input was dropped out, this also holds for the hidden nodes in S_b. Because the weight to nodes in S_2 from input 1 is negative, these nodes all evaluate to 0. Thus, before multiplying componentwise by variables $r_1/p, ..., r_K/p$, a total of $\frac{2K}{3}$ hidden nodes evaluate to $\frac{3}{2K}$, and the rest evaluate to 0. Since each r_k/p has expectation 1, this completes the proof in this case, and thereby overall. ■

Proof: (of Theorem 15): Using a bias-variance decomposition, for any weights $\mathcal{W}$,

$$\begin{aligned}
\tilde{J}_{\mathcal{D}}(\mathcal{W}) &= \sum_{\mathbf{s} \in \{(0,1),(1,0),(1,1)\}} \mathbf{Pr}[\mathbf{s}] \mathbf{E_r}[(\mathcal{D}(\mathcal{W}, (\epsilon, \epsilon), (\mathbf{s}, \mathbf{r})) - 1)^2] \\
&= \sum_{\mathbf{s} \in \{(0,1),(1,0),(1,1)\}} \mathbf{Pr}[\mathbf{s}] \left((\mathbf{E_r}[\mathcal{D}(\mathcal{W}, (\epsilon, \epsilon), (\mathbf{s}, \mathbf{r}))] - 1)^2 + \mathbf{Var_r}[\mathcal{D}(\mathcal{W}, (\epsilon, \epsilon), (\mathbf{s}, \mathbf{r}))] \right) \quad (2) \\
&\geq p^2 (\mathbf{E_r}[\mathcal{D}(\mathcal{W}, (\epsilon, \epsilon), ((1,1), \mathbf{r}))] - 1)^2. \quad (3)
\end{aligned}$$

Thus, for any dropout optimizer $\mathcal{W}_{\mathcal{D}}$, we have

$$p^2 (\mathbf{E_r}[\mathcal{D}(\mathcal{W}_{\mathcal{D}}, (\epsilon, \epsilon), ((1,1), \mathbf{r}))] - 1)^2 \leq \tilde{J}_{\mathcal{D}}(\mathcal{W}_{\mathcal{D}}) \leq \tilde{J}_{\mathcal{D}}(\mathcal{U}). \quad (4)$$

Lemma 19 and (2) together imply that

$$\tilde{J}_{\mathcal{D}}(\mathcal{U}) = \sum_{\mathbf{s} \in \{(0,1),(1,0),(1,1)\}} \mathbf{Pr}[\mathbf{s}] \mathbf{Var_r}[\mathcal{D}(\mathcal{U}, (\epsilon, \epsilon), (\mathbf{s}, \mathbf{r}))].$$

The value of $\mathcal{D}(\mathcal{U}, (\epsilon, \epsilon), ((1,1), \mathbf{r}))$ is obtained by scaling a $\mathrm{Bin}(K/3, p)$-distributed random variable by a factor $\frac{3}{Kp}$. Thus

$$\mathbf{Var_r}[\mathcal{D}(\mathcal{U}, (\epsilon, \epsilon), ((1,1), \mathbf{r}))] = \frac{9}{K^2 p^2} \times \frac{Kp(1-p)}{3} = \frac{3(1-p)}{Kp}.$$

Similarly, $\mathcal{D}(\mathcal{U}, (\epsilon, \epsilon), ((1,0), \mathbf{r}))$ and $\mathcal{D}(\mathcal{U}, (\epsilon, \epsilon), ((0,1), \mathbf{r}))$ are scaled $\mathrm{Bin}(2k/3, p)$ random variables, so

$$\begin{aligned}
\mathbf{Var_r}[\mathcal{D}(\mathcal{U}, (\epsilon, \epsilon), ((1,0), \mathbf{r}))] &= \mathbf{Var_r}[\mathcal{D}(\mathcal{U}, (\epsilon, \epsilon), ((0,1), \mathbf{r}))] \\
&= \frac{9}{4K^2 p^2} \times \frac{2Kp(1-p)}{3} \\
&= \frac{3(1-p)}{2Kp},
\end{aligned}$$

so that

$$\tilde{J}_{\mathcal{D}}(\mathcal{U}) = p^2 \times \frac{3(1-p)}{Kp} + 2p(1-p) \times \frac{3(1-p)}{2Kp} = \frac{3(1-p)}{K}. \tag{5}$$

We can now get a bound for $\mathcal{W}_D$ by applying (4): $p^2(\mathbf{E}_{\mathbf{r}}[\mathcal{D}(\mathcal{W}_D, (\epsilon, \epsilon), ((1,1), \mathbf{r}))] - 1)^2 \leq \frac{3(1-p)}{K}$. This implies

$$(\mathbf{E}_{\mathbf{r}}[\mathcal{D}(\mathcal{W}_D, (\epsilon, \epsilon), ((1,1), \mathbf{r}))] - 1)^2 \leq \frac{3(1-p)}{p^2 K}. \tag{6}$$

For all $\mathcal{W}$ and all $\mathbf{x}$, the only effect of the dropout process when all the nodes are kept is to scale up the values at each layer by a factor of $1/p$, so

$$\mathcal{D}(\mathcal{W}, \mathbf{x}, ((1,1), (1, ..., 1))) = \frac{1}{p^2}\mathcal{W}(\mathbf{x}).$$

Since the components of $\mathbf{r}$ have mean p and the output node is linear, we have

$$\mathcal{D}(\mathcal{W}, \mathbf{x}, ((1,1), (1, ..., 1))) = \frac{1}{p}\mathbf{E}_{\mathbf{r}}[\mathcal{D}(\mathcal{W}, \mathbf{x}, ((1,1), \mathbf{r}))],$$

giving

$$\mathbf{E}_{\mathbf{r}}[\mathcal{D}(\mathcal{W}, \mathbf{x}, ((1,1), \mathbf{r}))] = \frac{1}{p}\mathcal{W}(\mathbf{x}).$$

Substituting into (6) gives

$$\left(\frac{1}{p}\mathcal{W}_{\mathcal{D}}(\epsilon, \epsilon) - 1\right)^2 \leq \frac{3(1-p)}{Kp^2}.$$

This can be rewritten

$$\mathcal{W}_{\mathcal{D}}(\epsilon, \epsilon) \in \left[p - \sqrt{\frac{3(1-p)}{K}}, p + \sqrt{\frac{3(1-p)}{K}}\right]$$

or

$$1 - \mathcal{W}_{\mathcal{D}}(\epsilon, \epsilon) \in \left[q - \sqrt{\frac{3q}{K}}, q + \sqrt{\frac{3q}{K}}\right].$$

Thus,

$$(\mathcal{W}_{\mathcal{D}}(\epsilon, \epsilon) - 1)^2 \leq \left(q + \sqrt{\frac{3q}{K}}\right)^2 \leq (1 + o(1))q^2,$$

completing the proof. ∎

5.3 Dropout uses negative weights

We can now prove the surprising result that the optimizing $\mathcal{W}_D$ for source P_2 must use negative weights, even though the function to be learned is a mapping from positive $\mathbf{x} = (\epsilon, \epsilon)$ to $y = 1$, and, for all $\mathcal{W}$, $\mathcal{W}(0) = 0$.

Theorem 20 *If $K > 3(1+p)/p^2$, any $\mathcal{W}_D$ minimizing $\tilde{J}_{\mathcal{D}}(\mathcal{W})$ for source P_2 must include negative weights.*

Proof: Assume to the contrary that all weights in a $\mathcal{W}_D$ are non-negative. When the inputs are positive, every hidden node will compute a non-negative value, and output the weighted sum of its inputs. Therefore the network with weights $\mathcal{W}_D$ behaves linearly, and this linear behavior is preserved under dropout patterns. In particular, if $\mathbf{x}$ and $\mathbf{x}'$ have no negative components then for all dropout patterns $\mathcal{R}$:

$$\mathcal{D}(\mathcal{W}_D, \mathbf{x}, \mathcal{R}) + \mathcal{D}(\mathcal{W}_D, \mathbf{x}', \mathcal{R}) = \mathcal{D}(\mathcal{W}_D, \mathbf{x} + \mathbf{x}', \mathcal{R}) \tag{7}$$

(see Baldi and Sadowski [2014] for additional properties of dropout in linear networks).

We now use a bound on $\tilde{J}_{\mathcal{D}}(\mathcal{W}_D)$ obtained by neglecting the variance terms in (2):

$$\begin{aligned}
\tilde{J}_{\mathcal{D}}(\mathcal{W}_D) &\geq \sum_{\mathbf{s} \in \{(0,1),(1,0),(1,1)\}} \mathbf{Pr}[\mathbf{s}](\mathbf{E}_{\mathbf{r}}[\mathcal{D}(\mathcal{W}_D, (\epsilon, \epsilon), (\mathbf{s}, \mathbf{r}))] - 1)^2 \\
&= p^2(\mathbf{E}_{\mathbf{r}}[\mathcal{D}(\mathcal{W}_D, (\epsilon, \epsilon), ((1,1), \mathbf{r}))] - 1)^2 \\
&\quad + p(1-p)(\mathbf{E}_{\mathbf{r}}[\mathcal{D}(\mathcal{W}_D, (\epsilon, \epsilon), ((1,0), \mathbf{r}))] - 1)^2 \\
&\quad + p(1-p)(\mathbf{E}_{\mathbf{r}}[\mathcal{D}(\mathcal{W}_D, (\epsilon, \epsilon), ((0,1), \mathbf{r}))] - 1)^2.
\end{aligned}$$

Equation (7) implies that for every dropout pattern $\mathbf{r}$ on the hidden nodes,

$$\begin{aligned}
\mathcal{D}(\mathcal{W}_D, (\epsilon, \epsilon), ((1,1), \mathbf{r})) &= \mathcal{D}(\mathcal{W}_D, (\epsilon, 0), ((1,1), \mathbf{r})) + \mathcal{D}(\mathcal{W}_D, (0, \epsilon), ((1,1), \mathbf{r})) \\
&= \mathcal{D}(\mathcal{W}_D, (\epsilon, \epsilon), ((1,0), \mathbf{r})) + \mathcal{D}(\mathcal{W}_D, (\epsilon, \epsilon), ((0,1), \mathbf{r})).
\end{aligned}$$

Plugging this in and abbreviating the expectations $\mathbf{E}_{\mathbf{r}}[\mathcal{D}(\mathcal{W}_D, (\epsilon, \epsilon), (\mathbf{s}, \mathbf{r}))]$ with $\mu_{\mathbf{s}}$ we get

$$\tilde{J}_{\mathcal{D}}(\mathcal{W}_D) \geq p^2(\mu_{1,0} + \mu_{0,1} - 1)^2 + p(1-p)(\mu_{1,0} - 1)^2 + p(1-p)(\mu_{0,1} - 1)^2.$$

Consider the right-hand-side as a function of $\mu_{1,0}$ and $\mu_{0,1}$. It is convex and symmetric, so is minimized when $\mu_{1,0} = \mu_{0,1} = \mu$, implying

$$\tilde{J}_{\mathcal{D}}(\mathcal{W}_D) \geq \min_{\mu} \; p^2(2\mu - 1)^2 + 2p(1-p)(\mu - 1)^2.$$

Using calculus, the minimizing $\mu = 1/(1+p)$. Using this value and algebra yields

$$\tilde{J}_{\mathcal{D}}(\mathcal{W}_D) \geq \frac{(1-p)p^2}{1+p}.$$

Equation (5) shows that $\tilde{J}_{\mathcal{D}}(\mathcal{U}) = 3(1-p)/K$, which leads to the desired contradiction when the number of hidden nodes $K > 3(1+p)/p^2$. ■

Note that, when p is close to 1, then Theorem 20 applies when $K > 6$.

The negative weights in Construction $\mathcal{U}$ were created specifically to control the variance under dropout. It is intriguing that, in this situation, the weight vector optimizing the dropout criteria *must* also use negative weights, presumably to control the variance of its outputs. Since the variance is closely related to the probability of computing large values, an alternative interpretation is that the negative weights learned by dropout serve the purpose of reducing the probability under dropout of computing large values.

Typically, regularizers soften or remove structure from learned models, but here we see dropout adding structure (negative weights) without evidence for it in the data. Previously dropout has ben interpreted as discouraging the co-adaption of weights (e.g. Hinton et al. [2012], Srivastava et al. [2014]). In this example, intuitively, a minimizer of the dropout criterion uses coadapted negative weights to reduce the variance of the output of the network by sometimes countering the effect of associated positive weights.

6 A necessary condition for negative dropout penalty

Section 2 contains an example where the dropout penalty is negative. The following theorem includes a necessary condition.

Theorem 21 *The dropout penalty can be negative. A necessary condition for this in rectified linear networks is that some weights (or inputs) are negative.*

Proof: Baldi and Sadowski [2014] show that for networks of linear units (as opposed to the non-linear rectified linear units we focus on) the network's output without dropout equals the expected output over dropout patterns, so in our notation: $\mathcal{W}(\mathbf{x})$ equals $\mathbf{E}_{\mathcal{R}}(\mathcal{D}(\mathcal{W}, \mathbf{x}, \mathcal{R}))$. Assume for the moment that the network consists of linear units and the source is concentrated on the single input $\mathbf{x}$. Using the bias-variance decomposition for square loss and this property of linear networks,

$$J_D(\mathcal{W}) = \mathbf{E}_{\mathcal{R}}\left((\mathcal{D}(\mathcal{W}, \mathbf{x}, \mathcal{R}) - y)^2\right) = (\mathbf{E}_{\mathcal{R}}(\mathcal{D}(\mathcal{W}, \mathbf{x}, \mathcal{R}) - y)^2 + \mathbf{Var}_{\mathcal{R}}(\mathcal{D}(\mathcal{W}, \mathbf{x}, \mathcal{R})) \geq (\mathcal{W}(\mathbf{x}) - y)^2$$

and the dropout penalty is again non-negative. Since the same calculations go through when averaging over a source distribution, we see that the dropout penalty is always non-negative for networks of linear nodes. When all the weights and inputs in a network of rectified linear units are positive, then the rectified linear units behave as linear units, so the dropout penalty will again be non-negative. ■

7 Characterizing when the multi-layer dropout penalty depends on the labels

Recall that the dropout penalty is the amount that the expected dropout loss exceeds the square loss of the network. In contrast with its behavior on a variety of linear models including logistic regression, the dropout penalty can depend on the value of the response variable in deep networks with ReLUs and the quadratic loss.

Before formally stating our result, we introduce the following notation. Call a node in the network *penultimate* if it connects directly to the output node, and let $\underline{K}$ be the number of penultimate nodes. Let $\mathcal{W}$ be an arbitrary setting of all the weights of the network, and $\underline{\mathbf{w}} \in \mathbf{R}^K$ be the output node's vector of weights (for the K penultimate nodes).

For each input $\mathbf{x}$ let $\underline{\mathbf{h}(\mathcal{W}, \mathbf{x})}$ be the vector of K values produced by the penultimate nodes (without dropout). Similarly, for dropout pattern $\mathcal{R}$, let $\underline{\mathcal{H}(\mathcal{W}, \mathbf{x}, \mathcal{R})}$ be the vector of K values presented to the output node by the dropout process (including zeroing if the corresponding node is dropped, and scaling up if it is kept).

When using dropout in deep networks with non-linear units, many weight settings have the property that $\mathbf{h}(\mathcal{W}, \mathbf{x}) \neq \mathbf{E}_{\mathcal{R}}(\mathcal{H}(\mathcal{W}, \mathbf{x}, \mathcal{R})$. Consider the case where $\mathbf{x} = (1, -2)$ and a hidden node has similar positive weights on the two inputs. The value at the hidden node without dropout will be 0, but with dropout the hidden node's value is never negative but will be positive when the negative input is dropped and the positive one kept. Therefore the expected hidden node value under dropout is positive, and different from its value without dropout. Similar situations arise when the two inputs have the same sign, but their weights at the hidden node have different signs. In fact, the expected value (under dropout) of ReLU hidden nodes is the same as the value computed without dropout only when all of the weight-input products have the same sign.

We are now ready to state the theorem showing when the dropout penalty depends on the labels y.

Theorem 22 *The dropout penalty for weights $\mathcal{W}$ and the source distribution concentrated on a single example $(\mathbf{x}, y)$ depends on the label y if and only if*

$$\mathbf{w} \cdot (\mathbf{E}_{\mathcal{R}}(\mathcal{H}(\mathcal{W}, \mathbf{x}, \mathcal{R})) - \mathbf{h}(\mathcal{W}, \mathbf{x})) \neq 0. \tag{8}$$

For a distribution P that is not concentrated on a single example, if any instance $\mathbf{x}$ with positive probability under P satisfies (8), then the dropout penalty will depend on the label (distribution) y that P associates with the instance.

Proof (of Theorem 22): Suppose that P is concentrated on a single $(\mathbf{x}, y)$ pair, and the weights are $\mathcal{W}$.

Since the output node is a linear node, the output of the network without dropout is $\mathbf{w} \cdot \mathbf{h}(\mathcal{W}, \mathbf{x})$, while the output of the network under dropout pattern $\mathcal{R}$ is $\mathbf{w} \cdot \mathcal{H}(\mathcal{W}, \mathbf{x}, \mathcal{R})$. We now examine the dropout penalty, which is the expected dropout loss minus the non-dropout loss. We will use $\boldsymbol{\delta}$ as a shorthand for $\mathcal{H}(\mathcal{W}, \mathbf{x}, \mathcal{R}) - \mathbf{h}(\mathcal{W}, \mathbf{x})$.

$$\begin{aligned}
\text{dropout penalty} &= \mathbf{E}_{\mathcal{R}}\left((\mathbf{w} \cdot \mathcal{H}(\mathcal{W}, \mathbf{x}, \mathcal{R}) - y)^2\right) - (\mathbf{w} \cdot \mathbf{h}(\mathcal{W}, \mathbf{x}) - y)^2 \\
&= \mathbf{E}_{\mathcal{R}}\left((\mathbf{w} \cdot \mathcal{H}(\mathcal{W}, \mathbf{x}, \mathcal{R}) - \mathbf{w} \cdot \mathbf{h}(\mathcal{W}, \mathbf{x}) + \mathbf{w} \cdot \mathbf{h}(\mathcal{W}, \mathbf{x}) - y)^2\right) - (\mathbf{w} \cdot \mathbf{h}(\mathcal{W}, \mathbf{x}) - y)^2 \\
&= \mathbf{E}_{\mathcal{R}}\left((\mathbf{w} \cdot \boldsymbol{\delta} + \mathbf{w} \cdot \mathbf{h}(\mathcal{W}, \mathbf{x}) - y)^2\right) - (\mathbf{w} \cdot \mathbf{h}(\mathcal{W}, \mathbf{x}) - y)^2 \\
&= \mathbf{E}_{\mathcal{R}}\left((\mathbf{w} \cdot \boldsymbol{\delta})^2 + 2(\mathbf{w} \cdot \boldsymbol{\delta})(\mathbf{w} \cdot \mathbf{h}(\mathcal{W}, \mathbf{x}) - y)\right) \\
&= \mathbf{E}_{\mathcal{R}}\left((\mathbf{w} \cdot \boldsymbol{\delta})^2\right) + 2(\mathbf{w} \cdot \mathbf{h}(\mathcal{W}, \mathbf{x}) - y)\mathbf{E}_{\mathcal{R}}(\mathbf{w} \cdot \boldsymbol{\delta})
\end{aligned}$$

which depends on the label y if and only if $\mathbf{E}_{\mathcal{R}}(\mathbf{w} \cdot \boldsymbol{\delta}) \neq 0$. ■

Theorem 22 does not rely on the particular kind of non-linear hidden nodes, and other kinds of nodes are unlikely have their expected values under dropout equal their non-dropout values. Consider a two-input sigmoided node with its (weighted) inputs having the same sign. If the activation is a large negative number then non-dropout output is very close to zero (logistic sigmoid) or -1 (arctan). On the other hand, both inputs will be dropped out with probability 1/4, so the expected dropout output is at least 1/8 (logistic) or -3/4 (arctan).

8 Discussion

We have theoretically examined the inductive bias of dropout, describing some of its characteristics, and contrasting it with weight decay.

When demonstrating the effect of dropout on learning algorithms, we have focused on simple sources concentrated on single examples. It is important to realize that any distribution with positive probability on examples like those we analyze will have some amount of the relevant properties, and unless there is coincidental cancellation, the distribution as a whole will exhibit (a reduced version of) the key behavior.

Deep networks of ReLUs enable the efficient computation of large values. Section 3 shows that dropout is unable to fit such large values even when weight decay can. The proof of this suggests that dropout regularization discriminates against weights leading to large values.

Our results in Section 5.3 show that minimizing the dropout criterion requires the use of negative weights, and suggests that these are used in a co-adaptive way to control the magnitude (or variance) of values computed by the dropout process. This indicates that restricting co-adaptation of weights cannot the whole story behind dropout's behavior.

Building on this analysis to more fully characterize the conditions when dropout or weight decay are more compatible with a source is the natural next step, with the ultimate goal of characterizing when dropout is most beneficial.

In addition to a possible beneficial effect on inductive bias, another possible explanation for the utility of dropout could be that dropout training is less susceptible to being trapped in local minima or stalling out on flat regions of the error surface. Exploring this could be another direction for future work.

Acknowledgments

We thank Peter Bartlett for valuable conversations.

References

P. Bachman, O. Alsharif, and D. Precup. Learning with pseudo-ensembles. *NIPS*, 2014.

P. Baldi and P. Sadowski. The dropout learning algorithm. *Artificial intelligence*, 210:78–122, 2014.

P. Baldi and P. J. Sadowski. Understanding dropout. In *Advances in Neural Information Processing Systems*, pages 2814–2822, 2013.

P. L. Bartlett, M. I. Jordan, and J. D. McAuliffe. Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101(473):138–156, 2006.

L. Breiman. Some infinity theory for predictor ensembles. *Annals of Statistics*, 32(1):1–11, 2004.

Caffe. Caffe, 2016. http://caffe.berkeleyvision.edu.

D. Chen and C. D. Manning. A fast and accurate dependency parser using neural networks. In *EMNLP*, pages 740–750, 2014.

G. E. Dahl. Deep learning how I did it: Merck 1st place interview, 2012. http://blog.kaggle.com.

G. E. Dahl, T. N. Sainath, and G. E. Hinton. Improving deep neural networks for LVCSR using rectified linear units and dropout. *ICASSP*, 2013.

L. Deng, J. Li, J. Huang, K. Yao, D. Yu, F. Seide, M. L. Seltzer, G. Zweig, X. He, J. Williams, Y. Gong, and A. Acero. Recent advances in deep learning for speech research at microsoft. *ICASSP*, 2013.

I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. C. Courville, and Y. Bengio. Maxout networks. In *ICML*, pages 1319–1327, 2013.

K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, pages 1026–1034, 2015.

D. P. Helmbold and P. M. Long. On the inductive bias of dropout. *JMLR*, 16:3403–3454, 2015.

G. E. Hinton. Dropout: a simple and effective way to improve neural networks, 2012. videolectures.net.

G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors, 2012. Arxiv, arXiv:1207.0580v1.

N. Kalchbrenner, E. Grefenstette, and P. Blunsom. A convolutional neural network for modelling sentences. In *ACL*, pages 655–665, 2014.

P. M. Long and R. A. Servedio. Random classification noise defeats all convex potential boosters. *Machine Learning*, 78(3):287–304, 2010.

V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, pages 807–814, 2010.

B. Neyshabur, R. Tomioka, and N. Srebro. Norm-based capacity control in neural networks. In *COLT*, pages 1376–1401, 2015.

J. Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 15:1929–1958, 2014.

C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *CVPR*, 2015.

TensorFlow. Tensorflow, 2016. https://www.tensorflow.org.

Torch. Torch, 2016. http://torch.ch.

T. Van Erven, W. Kotowski, and M. K. Warmuth. Follow the leader with dropout perturbations. *COLT*, pages 949–974, 2014.

S. Wager, S. Wang, and P. Liang. Dropout training as adaptive regularization. *NIPS*, 2013.

S. Wager, W. Fithian, S. Wang, and P. S. Liang. Altitude training: Strong bounds for single-layer dropout. *NIPS*, 2014.

L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus. Regularization of neural networks using dropconnect. In *ICML*, pages 1058–1066, 2013.

T. Zhang. Statistical behavior and consistency of classification methods based on convex risk minimization. *Annals of Statistics*, 32(1):56–85, 2004.

A Table of Notation

Notation	Meaning
$\mathbb{1}_{\text{set}}$	indicator function for "set"
$(\mathbf{x}, y)$	an example with feature vector $\mathbf{x}$ and label y
$\sigma(\cdot)$	the rectified linear unit computing $\max(0, \cdot)$
$\mathcal{W}$	an arbitrary weight setting for the network
w, v	specific weights, often subscripted
$\mathcal{W}(\mathbf{x})$	the output value produced by weight setting $\mathcal{W}$ on input $\mathbf{x}$
P	an arbitrary source distribution over $(\mathbf{x}, y)$ pairs
$R_P(\mathcal{W})$	the risk (expected square loss) of $\mathcal{W}$ under source P
q, p	probabilities that a node is dropped out (q) or kept (p) by the dropout process
$\mathcal{R}$	a dropout pattern, indicates the kept nodes
$\mathbf{r}, \mathbf{s}$	dropout patterns on subsets of the nodes
$\mathcal{D}(\mathcal{W}, \mathbf{x}, \mathcal{R})$	Output of dropout with network weights $\mathcal{W}$, input $\mathbf{x}$, and dropout pattern $\mathcal{R}$
$J_D(\mathcal{W})$	the dropout criterion
$J_2(\mathcal{W})$	the L_2 criterion
λ	the L_2 regularization strength parameter
$\mathcal{W}_D$	an optimizer of the dropout criterion
$\mathcal{W}_{L_2}$	an optimizer of the L_2 criterion
P_1 and P_2	source distributions used in Sections 5 and 4 respectively
α	the probability of non-zero example under P_2
ϵ	feature value under P_2
n, d	the network width and depth
K	the number of nodes in a hidden layer
k	an arbitrary node (index) in the hidden layer
i	an arbitrary input node (index)
$\mathcal{Z}$	the all-zero weight vector
$\mathcal{U}$	a constructed weight vector used in section 5
$\tilde{J}_{\mathcal{D}}(\mathcal{W})$	$J_D()$ modified by neglecting terms where network cut (Section 4)
$\tilde{J}_{\mathcal{D}}(\mathcal{W})$	$J_D()$ modified by neglecting terms where all inputs dropped (Section 5)
$\mathcal{H}(\mathcal{W}, \mathbf{x}, \mathcal{R})$	Values output by top (last) hidden layer after dropout at all levels