On the Implicit Bias of Dropout

Poorya Mianjy [1] **Raman Arora** [1] **Rene Vidal** [2]

Abstract

Algorithmic approaches endow deep learning systems with implicit bias that helps them generalize even in over-parametrized settings. In this paper, we focus on understanding such a bias induced in learning through dropout, a popular technique to avoid overfitting in deep learning. For single hidden-layer linear neural networks, we show that dropout tends to make the norm of incoming/outgoing weight vectors of all the hidden nodes equal. In addition, we provide a complete characterization of the optimization landscape induced by dropout.

1. Introduction

Modern machine learning systems based on deep neural networks are usually over-parameterized, i.e. the number of parameters in the model is much larger than the size of the training data, which makes these systems prone to overfitting. Several explicit regularization strategies have been used in practice to help these systems generalize, including ℓ_1 and ℓ_2 regularization of the parameters (Nowlan and Hinton, 1992). Recently, (Neyshabur et al., 2015) showed that a variety of such norm-based regularizers can provide size-independent capacity control, suggesting that the network size is not a good measure of complexity in such settings. Such a view had been previously motivated in the context of matrix factorization (Srebro et al., 2005), where it is preferable to have many factors of limited overall influence rather than a few important ones.

Besides explicit regularization techniques, practitioners have used a spectrum of algorithmic approaches to improve the generalization ability of over-parametrized models. This includes early stopping of back propagation (Caruana et al., 2001), batch normalization (Ioffe and Szegedy, 2015) and dropout (Srivastava et al., 2014). In particular, dropout, which is the focus of this paper, randomly drops hidden nodes along with their connections at training time. Dropout was introduced by Srivastava et al. (2014) as a way of breaking up co-adaptation among neurons, drawing insights from the success of the sexual reproduction model in the evolution of advanced organisms. While dropout has enjoyed tremendous success in training deep neural networks, the theoretical understanding of how dropout (and other algorithmic heuristics) provide regularization in deep learning remains somewhat limited.

We argue that a prerequisite for understanding implicit regularization due to various algorithmic heuristics in deep learning, including dropout, is to analyze their behavior in simpler models. Therefore, in this paper, we consider the following learning problem. Let $\mathrm{x} \in \mathbb{R}^{d_2}$ represent an input feature vector with some unknown distribution $\mathcal{D}$ such that $\mathbb{E}_{\mathrm{x}\sim\mathcal{D}}[\mathrm{x}\mathrm{x}^\top] = \mathrm{I}$. The output label vector $\mathrm{y} \in \mathbb{R}^{d_1}$ is given as $\mathrm{y} = \mathrm{M}\mathrm{x}$ for some $\mathrm{M} \in \mathbb{R}^{d_1 \times d_2}$. We consider the hypothesis class represented by a single hidden-layer linear network parametrized as $h_{\mathrm{U},\mathrm{V}}(\mathrm{x}) = \mathrm{U}\mathrm{V}^\top\mathrm{x}$, where $\mathrm{V} \in \mathbb{R}^{d_2\times r}$ and $\mathrm{U} \in \mathbb{R}^{d_1\times r}$ are the weight matrices in the first and the second layers, respectively. The goal of learning is to find weight matrices U, V that minimize the expected loss $\ell(\mathrm{U},\mathrm{V}) := \mathbb{E}_{\mathrm{x}\sim\mathcal{D}}[\|\mathrm{y}-h_{\mathrm{U},\mathrm{V}}(\mathrm{x})\|^2] = \mathbb{E}_{\mathrm{x}\sim\mathcal{D}}[\|\mathrm{y}-\mathrm{U}\mathrm{V}^\top\mathrm{x}\|^2]$.

A natural learning algorithm to consider is back-propagation with dropout, which can be seen as an instance of stochastic gradient descent on the following objective:

$$f(\mathrm{U},\mathrm{V}) := \mathbb{E}_{b_i\sim\mathrm{Ber}(\theta),\mathrm{x}\sim\mathcal{D}}\left[\left\|\mathrm{y} - \frac{1}{\theta}\mathrm{U}\,\mathrm{diag}(\mathrm{b})\mathrm{V}^\top\mathrm{x}\right\|^2\right], \quad (1)$$

where the expectation is w.r.t. the underlying distribution on data as well as randomization due to dropout (each hidden unit is dropped independently with probability $1-\theta$). This procedure, which we simply refer to as dropout in this paper, is given in Algorithm 1.

It is easy to check (see Lemma A.1 in the supplementary) that the objective in equation (1) can be written as

$$f(\mathrm{U},\mathrm{V}) = \ell(\mathrm{U},\mathrm{V}) + \lambda\sum_{i=1}^{r}\|\mathrm{u}_i\|^2\|\mathrm{v}_i\|^2, \quad (2)$$

where $\lambda = \frac{1-\theta}{\theta}$ is the regularization parameter, and u_i and v_i represent the i^{th} columns of U and V, respectively. Note

[1]Department of Computer Science, Johns Hopkins University, Baltimore, USA [2]Department of Biomedical Engineering, Johns Hopkins University, Baltimore, USA. Correspondence to: Raman Arora <arora@cs.jhu.edu>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018.

that while the goal was to minimize the expected squared loss, using dropout with gradient descent amounts to finding a minimum of the objective in equation (2); we argue that the additional term in the objective serves as a regularizer, $R(\mathrm{U},\mathrm{V}) := \lambda \sum_{i=1}^{r} \|\mathrm{u}_i\|^2 \|\mathrm{v}_i\|^2$, and is an explicit instantiation of the implicit bias of dropout. Furthermore, we note that this regularizer is closely related to *path regularization* which is given as the square-root of the sum over all paths, from input to output, of the product of the squared weights along the path (Neyshabur et al., 2015). Formally, for a single layer network, path regularization is given as

$$\psi_2(\mathrm{U},\mathrm{V}) = \left(\sum_{i=1}^{r} \sum_{j=1}^{d_1} \sum_{k=1}^{d_2} u_{ji}^2 v_{ki}^2 \right)^{\frac{1}{2}}. \tag{3}$$

Interestingly, the dropout regularizer is equal to the square of the path regularizer, i.e. $R(\mathrm{U},\mathrm{V}) = \lambda \psi_2^2(\mathrm{U},\mathrm{V})$. While this observation is rather immediate, it has profound implications owing to the fact that path regularization provides size-independent capacity control in deep learning, thereby supporting empirical evidence that dropout finds good solutions in over-parametrized settings.

In this paper, we focus on studying the optimization landscape of the objective in equation (2) for a single hidden-layer linear network with dropout and the special case of an autoencoder with tied weights. Furthermore, we are interested in characterizing the solutions to which dropout (i.e. Algorithm 1) converges. We make the following progress toward addressing these questions.

1. We formally characterize the implicit bias of dropout. We show that, when minimizing the expected loss $\ell(\mathrm{U},\mathrm{V})$ with dropout, any global minimum $(\tilde{\mathrm{U}},\tilde{\mathrm{V}})$ satisfies $\psi_2(\tilde{\mathrm{U}},\tilde{\mathrm{V}}) = \min\{\psi_2(\mathrm{U},\mathrm{V}) \text{ s.t. } \mathrm{U}\mathrm{V}^\top = \tilde{\mathrm{U}}\tilde{\mathrm{V}}^\top\}$. More importantly, for auto-encoders with tied weights, we show that all *local* minima inherit this property.
2. Despite the non-convex nature of the problem, we completely characterize the global optima by giving necessary and sufficient conditions for optimality.
3. We describe the optimization landscape of the dropout problem. In particular, we show that for a sufficiently small dropout rate, all local minima of the objective in equation (2) are global and all saddle points are non-degenerate. This allows Algorithm 1 to efficiently escape saddle points and converge to a global optimum.

The rest of the paper is organized as follows. In Section 2, we study dropout for single hidden-layer linear auto-encoder networks with weights tied between the first and the second layers. This gives us the tools to study the dropout problem in a more general setting of single hidden-layer linear networks in Section 3. In Section 4, we characterize the optimization landscape of the objective in (2), show that it satisfies the strict saddle property, and that there are no spurious local minima. We specialize our results to matrix factorization in Section 5, and in Section 6, we discuss preliminary experiments to support our theoretical results.

Algorithm 1 Dropout with Stochastic Gradient Descent

input Data $\{(\mathrm{x}_t, \mathrm{y}_t)\}_{t=0}^{T-1}$, dropout rate $1-\theta$, learning rate η

1: Initialize $\mathrm{U}_0, \mathrm{V}_0$
2: **for** $t = 0, 1, \ldots, T-1$ **do**
3: sample b_t element-wise from Bernoulli(θ)
4: Update the weights

$$\mathrm{U}_{t+1} \leftarrow \mathrm{U}_t - \eta \left(\frac{1}{\theta} \mathrm{U}_t \operatorname{diag}(\mathrm{b}_t) \mathrm{V}_t^\top \mathrm{x}_t - \mathrm{y}_t \right) \mathrm{x}_t^\top \mathrm{V}_t \operatorname{diag}(\mathrm{b}_t)$$

$$\mathrm{V}_{t+1} \leftarrow \mathrm{V}_t - \eta \mathrm{x}_t \left(\frac{1}{\theta} \mathrm{x}_t^\top \mathrm{V}_t \operatorname{diag}(\mathrm{b}_t) \mathrm{U}_t^\top - \mathrm{y}_t^\top \right) \mathrm{U}_t \operatorname{diag}(\mathrm{b}_t)$$

5: **end for**

output $\mathrm{U}_T, \mathrm{V}_T$

1.1. Notation

We denote matrices, vectors, scalar variables and sets by Roman capital letters, Roman small letters, small letters and script letters respectively (e.g. X, x, x, and $\mathcal{X}$). For any integer d, we represent the set $\{1,\ldots,d\}$ by $[d]$. For any integer i, e_i denotes the i-th standard basis. For any integer d, $1_d \in \mathbb{R}^d$ is the vector of all ones, $\|\mathrm{x}\|$ represents the ℓ_2-norm of vector x, and $\|\mathrm{X}\|$, $\|\mathrm{X}\|_F$, $\|\mathrm{X}\|_*$ and $\lambda_i(\mathrm{X})$ represent the spectral norm, the Frobenius norm, the nuclear norm and the i-th largest singular value of matrix X, respectively. $\langle \cdot, \cdot \rangle$ represents the standard inner product, for vectors or matrices, where $\langle \mathrm{X}, \mathrm{X}' \rangle = \mathrm{Tr}(\mathrm{X}^\top \mathrm{X}')$. For a matrix $\mathrm{X} \in \mathbb{R}^{d_1 \times d_2}$, $\operatorname{diag}(\mathrm{X}) \in \mathbb{R}^{\min\{d_1,d_2\}}$ returns its diagonal elements. Similarly, for a vector $\mathrm{x} \in \mathbb{R}^d$, $\operatorname{diag}(\mathrm{x}) \in \mathbb{R}^{d \times d}$ is a diagonal matrix with x on its diagonal. For any scalar x, we define $(x)_+ = \max\{x, 0\}$, and for a matrix X, $(\mathrm{X})_+$ is the elementwise application of $(\cdot)_+$ to X. For a matrix X with a compact singular value decomposition $\mathrm{X} = \mathrm{U}\Sigma\mathrm{V}^\top$, and for any scalar $\alpha \geq 0$, we define the singular-value shrinkage-thresholding operator as $\mathcal{S}_\alpha(\mathrm{X}) := \mathrm{U}(\Sigma - \alpha \mathrm{I})_+ \mathrm{V}^\top$.

2. Linear autoencoders with tied weights

We begin with a simpler hypothesis family of single hidden-layer linear auto-encoders with weights tied such that $\mathrm{U} = \mathrm{V}$. Studying the problem in this setting helps our intuition about the implicit bias that dropout induces on weight matrices U. This analysis will be extended to the more general setting of single hidden-layer linear networks in the next section.

Recall that the goal here is to find an autoencoder network represented by a weight matrix $\mathrm{U} \in \mathbb{R}^{d_2 \times r}$ that solves:

$$\min_{\mathrm{U} \in \mathbb{R}^{d_2 \times r}} \ell(\mathrm{U}, \mathrm{U}) + \lambda \sum_{i=1}^{r} \|\mathrm{u}_i\|^4, \tag{4}$$

where u_i is the i^{th} column of U. Note that the loss function $\ell(\mathrm{U}, \mathrm{U})$ is invariant under rotations, i.e., for any orthogonal transformation $\mathrm{Q} \in \mathbb{R}^{d \times d}, \mathrm{Q}^\top \mathrm{Q} = \mathrm{Q}\mathrm{Q}^\top = \mathrm{I}_d$, it holds that

$$\ell(\mathrm{U},\mathrm{U}) = \mathbb{E}_{\mathrm{x} \sim \mathcal{D}}[\|\mathrm{y} - \mathrm{U}\mathrm{Q}\mathrm{Q}^\top \mathrm{U}^\top \mathrm{x}\|^2] = \ell(\mathrm{U}\mathrm{Q},\mathrm{U}\mathrm{Q}),$$

so that applying a rotation matrix to a candidate solution U does not change the value of the loss function. However, the regularizer is not rotation-invariant and clearly depends on the choice of Q. Therefore, in order to solve Problem (4), we need to find a rotation matrix that minimizes the value of the regularizer for a given weight matrix.

To that end, let us denote the squared column norms of the weight matrix U by $\mathrm{n_u} = (\|\mathrm{u}_1\|^2, \ldots, \|\mathrm{u}_r\|^2)$ and let $1_r \in \mathbb{R}^r$ be the vector of all ones. Then, for any U,

$$\begin{aligned} R(\mathrm{U}, \mathrm{U}) &= \lambda \sum_{i=1}^{r} \|\mathrm{u}_i\|^4 = \frac{\lambda}{r} \|1_r\|^2 \|\mathrm{n_u}\|^2 \\ &\geq \frac{\lambda}{r} \langle 1_r, \mathrm{n_u} \rangle^2 = \frac{\lambda}{r} \left(\sum_{i=1}^{r} \|\mathrm{u}_i\|^2 \right)^2 = \frac{\lambda}{r} \|\mathrm{U}\|_F^4, \end{aligned}$$

where the inequality follows from Cauchy-Schwartz inequality. Hence, the regularizer is lower bounded by $\frac{\lambda}{r}\|\mathrm{U}\|_F^4$, with equality if and only if $\mathrm{n_u}$ is parallel to 1_r, i.e. when all the columns of U have equal norms. Since the loss function is rotation invariant, one can always decrease the value of the overall objective by rotating U such that UQ has a smaller regularizer. A natural question to ask, therefore, is *if there always exists a rotation matrix Q such that the matrix UQ has equal column norms.* In order to formally address this question, we introduce the following definition.

Definition 2.1 (Equalized weight matrix, equalized autoencoder, equalizer)**.** A weight matrix U is said to be *equalized* if all its columns have equal norms. An autoencoder with tied weights is said to be *equalized* if the norm of the incoming weight vector is equal across all hidden nodes in the network. An orthogonal transformation Q is said to be an *equalizer* of U (equivalently, of the corresponding autoencoder) if UQ is equalized.

Next, we show that any matrix U can be equalized.

Theorem 2.2. Any weight matrix $\mathrm{U} \in \mathbb{R}^{d \times r}$ (equivalently, the corresponding autoencoder network $h_{\mathrm{U},\mathrm{U}}$) can be equalized. Furthermore, there exists a polynomial time algorithm (Algorithm 2) that returns an equalizer for a given matrix.

The key insight here is that if $\mathrm{G_U} := \mathrm{U}^\top \mathrm{U}$ is the Gram matrix associated with the weight matrix U, then $h_{\mathrm{U},\mathrm{U}}$ is equalized by Q if and only if all diagonal elements of $\mathrm{Q}^\top \mathrm{G_U} \mathrm{Q}$ are equal. More importantly, if $\mathrm{G_U} = \mathrm{V}\Lambda\mathrm{V}^\top$ is an eigendecomposition of $\mathrm{G_U}$, then for $\mathrm{w} = \frac{1}{\sqrt{r}} \sum_{i=1}^{r} \mathrm{v}_i$, it holds that $\mathrm{w}^\top \mathrm{G_U} \mathrm{w} = \frac{\mathrm{Tr}\,\mathrm{G_U}}{r}$; Proof of Theorem 2.2 uses this property to recursively equalize all diagonal elements of $\mathrm{G_U}$.

Algorithm 2 $\mathtt{EQZ}(\mathrm{U})$ equalizer of an auto-encoder $h_{\mathrm{U},\mathrm{U}}$

input $\mathrm{U} \in \mathbb{R}^{d \times r}$
$\mathrm{G} \leftarrow \mathrm{U}^\top \mathrm{U}$
$\mathrm{Q} \leftarrow \mathrm{I}_r$
for $i = 1$ to r **do**
  $[\mathrm{V}, \Lambda] \leftarrow \mathtt{eig}(\mathrm{G})$ {$\mathrm{G} = \mathrm{V}\Lambda\mathrm{V}^\top$ eigendecomposition}
  $\mathrm{w} = \frac{1}{\sqrt{r-i+1}} \sum_{i=1}^{r-i+1} \mathrm{v}_i$
  $\mathrm{Q}_i \leftarrow [\mathrm{w}\ \mathrm{w}_\perp]$ {$\mathrm{w}_\perp \in \mathbb{R}^{(r-i+1)\times(r-i)}$ orthonormal basis for the Null space of w}
  $\mathrm{G} \leftarrow \mathrm{Q}_i^\top \mathrm{G} \mathrm{Q}_i$ {Making first diagonal element zero}
  $\mathrm{G} \leftarrow \mathrm{G}(2 : \mathrm{end}, 2 : \mathrm{end})$ {First principal submatrix}
  $\mathrm{Q} \leftarrow \mathrm{Q} \begin{bmatrix} \mathrm{I}_{i-1} & 0 \\ 0 & \mathrm{Q}_i \end{bmatrix}$
end for
output Q {such that UQ is equalized}

Finally, we argue that the implicit bias induced by dropout is closely related to the notion of equalized network introduced above. In particular, our main result of the section states that the dropout enforces any globally optimal network to be equalized. Formally, we show the following.

Theorem 2.3. If U is a global optimum of Problem 4, then U is equalized. Furthermore, it holds that

$$R(\mathrm{U}) = \frac{\lambda}{r} \|\mathrm{U}\|_F^4.$$

Theorem 2.3 characterizes the effect of regularization induced by dropout in learning autoencoders with tied weights. It states that for any globally optimal network, the columns of the corresponding weight matrix have equal norms. In other words, dropout tends to give equal weights to all hidden nodes – it shows that dropout implicitly biases the optimal networks towards having hidden nodes with limited overall influence rather than a few important ones.

While Theorem 2.3 makes explicit the bias of dropout and gives a necessary condition for global optimality in terms of the weight matrix U_*, it does not characterize the bias induced in terms of the network (i.e. in terms of $\mathrm{U}_*\mathrm{U}_*^\top$). The following theorem completes the characterization by describing globally optimal autoencoder networks. Since the goal is to understand the implicit bias of dropout, we specify the global optimum in terms of the true concept, M.

Theorem 2.4. For any $j \in [r]$, let $\kappa_j := \frac{1}{j} \sum_{i=1}^{j} \lambda_i(\mathrm{M})$. Furthermore, define $\rho := \max\{j \in [r] : \lambda_j(\mathrm{M}) > \frac{\lambda j \kappa_j}{r + \lambda j}\}$. Then, if U_* is a global optimum of Problem 4, it satisfies that $\mathrm{U}_*\mathrm{U}_*^\top = \mathcal{S}_{\frac{\lambda \rho \kappa_\rho}{r + \lambda \rho}}(\mathrm{M})$.

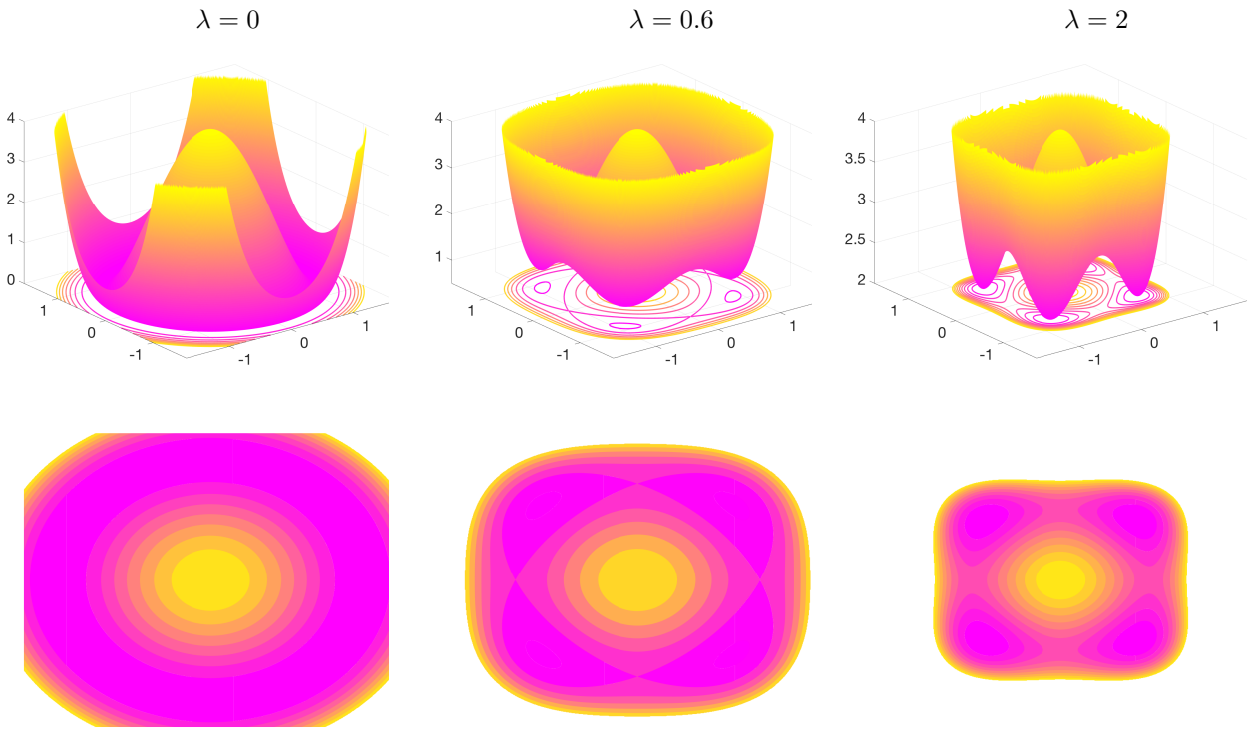

Figure 1: Optimization landscape (top) and contour plot (bottom) for a single hidden-layer linear autoencoder network with one dimensional input and output and a hidden layer of width $r = 2$ with dropout, for different values of the regularization parameter λ. Left: for $\lambda = 0$ the problem reduces to squared loss minimization, which is rotation invariant as suggested by the level sets. Middle: for $\lambda > 0$ the global optima shrink toward the origin. All local minima are global, and are equalized, i.e. the weights are parallel to the vector $(\pm 1, \pm 1)$. Right: as λ increases, global optima shrink further.

Remark 2.5. In light of Theorem 2.3, the proof of Theorem 2.4 entails solving the following optimization problem

$$\min_{\mathrm{U} \in \mathbb{R}^{d \times r}} \ell(\mathrm{U}, \mathrm{U}) + \frac{\lambda}{r} \|\mathrm{U}\|_F^4, \tag{5}$$

instead of Problem 4. This follows since the loss function $\ell(\mathrm{U}, \mathrm{U})$ is invariant under rotations, hence a weight matrix U cannot be optimal if there exists a rotation matrix Q such that $R(\mathrm{UQ}, \mathrm{UQ}) < R(\mathrm{U}, \mathrm{U})$. Now, while the objective in Problem 5 is a lower bound on the objective in Problem 4, by Theorem 2.2, we know that any weight matrix can be equalized. Thus, it follows that the minimum of the two problems coincide. Although Problem 5 is still non-convex, it is easier to study owing to a simpler form of the regularizer. Figure 1 shows how optimization landscape changes with different dropout rates for a single hidden layer linear autoencoder with one dimensional input and output and with a hidden layer of width two.

3. Single hidden-layer linear networks

Next, we consider the more general setting of a shallow linear network with a single hidden layer. Recall, that the goal is to find weight matrices U, V that solve

$$\min_{\mathrm{U} \in \mathbb{R}^{d_1 \times r}, \mathrm{V} \in \mathbb{R}^{d_2 \times r}} \ell(\mathrm{U}, \mathrm{V}) + \lambda \sum_{i=1}^{r} \|\mathrm{u}_i\|^2 \|\mathrm{v}_i\|^2. \tag{6}$$

As in the previous section, we note that the loss function is rotation invariant, i.e. $\ell(\mathrm{UQ}, \mathrm{VQ}) = \ell(\mathrm{U}, \mathrm{V})$ for any rotation matrix Q, however the regularizer is not invariant to rotations. Furthermore, it is easy to verify that both the loss function and the regularizer are invariant under rescaling of the incoming and outgoing weights to hidden neurons.

Remark 3.1 (Rescaling invariance)**.** The objective function in Problem (2) is invariant under rescaling of weight matrices, i.e. invariant to transformations of the form $\bar{\mathrm{U}} = \mathrm{UD}$, $\bar{\mathrm{V}} = \mathrm{VD}^{-1}$, where D is a diagonal matrix with positive entries. This follows since $\bar{\mathrm{U}}\bar{\mathrm{V}}^\top = \mathrm{UDD}^{-\top}\mathrm{V}^\top = \mathrm{UV}^\top$, so that $\ell(\bar{\mathrm{U}}, \bar{\mathrm{V}}) = \ell(\mathrm{U}, \mathrm{V})$, and also $R(\bar{\mathrm{U}}, \bar{\mathrm{V}}) = R(\mathrm{U}, \mathrm{V})$ since

$$\sum_{i=1}^{r} \|\bar{\mathrm{u}}_i\|^2 \|\bar{\mathrm{v}}_i\|^2 = \sum_{i=1}^{r} \|d_i \mathrm{u}_i\|^2 \|\frac{1}{d_i} \mathrm{v}_i\|^2 = \sum_{i=1}^{r} \|\mathrm{u}_i\|^2 \|\mathrm{v}_i\|^2.$$

As a result of rescaling invariance, $f(\bar{\mathrm{U}}, \bar{\mathrm{V}}) = f(\mathrm{U}, \mathrm{V})$. Now, following similar arguments as in the previous section,

we define $\mathrm{n}_{\mathrm{u,v}} = (\|\mathrm{u}_1\|\|\mathrm{v}_1\|, \ldots, \|\mathrm{u}_r\|\|\mathrm{v}_r\|)$, and note that

$$R(\mathrm{U},\mathrm{V}) = \lambda \sum_{i=1}^{r} \|\mathrm{u}_i\|^2\|\mathrm{v}_i\|^2 = \frac{\lambda}{r}\|1_r\|^2\|\mathrm{n}_{\mathrm{u,v}}\|^2$$
$$\geq \frac{\lambda}{r}\langle 1_r, \mathrm{n}_{\mathrm{u,v}}\rangle^2 = \frac{\lambda}{r}\left(\sum_{i=1}^{r}\|\mathrm{u}_i\|\|\mathrm{v}_i\|\right)^2,$$

where the inequality is due to Cauchy-Schwartz, and the lower bound is achieved if and only if $\mathrm{n}_{\mathrm{u,v}}$ is a scalar multiple of 1_r, i.e. *iff* $\|\mathrm{u}_i\|\|\mathrm{v}_i\| = \|\mathrm{u}_1\|\|\mathrm{v}_1\|$ for all $i = 1, \ldots, r$. This observation motivates the following definition.

Definition 3.2 (Jointly equalized weight matrices, equalized linear networks)**.** A pair of weight matrices $(\mathrm{U},\mathrm{V}) \in \mathbb{R}^{d_1\times r}\times\mathbb{R}^{d_2\times r}$ is said to be *jointly equalized* if $\|\mathrm{u}_i\|\|\mathrm{v}_i\| = \|\mathrm{u}_1\|\|\mathrm{v}_1\|$ for all $i \in [r]$. A single hidden-layer linear network is said to be equalized if the product of the norms of the incoming and outgoing weights are equal for all hidden nodes. Equivalently, a single hidden-layer network parametrized by weight matrices U,V, is equalized if U,V are jointly equalized. An orthogonal transformation $\mathrm{Q}\in\mathbb{R}^{r\times r}$ is an *equalizer* of a single hidden-layer network $h_{\mathrm{U,V}}$ parametrized by weight matrices U,V, if $h_{\mathrm{UQ,VQ}}$ is equalized. The network $h_{\mathrm{U,V}}$ (the pair(U,V)) then are said to be *jointly equalizable* by Q.

Note that Theorem 2.2 only guarantees the existence of an equalizer for an autoencoder with tied weights. It does not inform us regarding the existence of a rotation matrix that jointly equalizes a general network parameterized by a pair of weight matrices (U,V); in fact, it is not true in general that any pair (U,V) is jointly equalizable. Indeed, the general case requires a more careful treatment. It turns out that while a given pair of matrices (U,V) may not be jointly equalizable there exists a pair $(\tilde{\mathrm{U}},\tilde{\mathrm{V}})$ that is jointly equalizable and implements the same network function, i.e. $h_{\tilde{\mathrm{U}},\tilde{\mathrm{V}}} = h_{\mathrm{U,V}}$. Formally, we state the following result.

Theorem 3.3. For any given pair of weight matrices $(\mathrm{U},\mathrm{V})\in\mathbb{R}^{d_1\times r}\times\mathbb{R}^{d_2\times r}$, there exists another pair $(\tilde{\mathrm{U}},\tilde{\mathrm{V}})\in\mathbb{R}^{d_1\times r}\times\mathbb{R}^{d_2\times r}$ and a rotation matrix $\mathrm{Q}\in\mathbb{R}^{r\times r}$ such that $h_{\tilde{\mathrm{U}},\tilde{\mathrm{V}}} = h_{\mathrm{U,V}}$ and $h_{\tilde{\mathrm{U}},\tilde{\mathrm{V}}}$ is jointly equalizable by Q. Furthermore, for $\bar{\mathrm{U}} := \tilde{\mathrm{U}}\mathrm{Q}$ and $\bar{\mathrm{V}} := \tilde{\mathrm{V}}\mathrm{Q}$ it holds that $\|\bar{\mathrm{u}}_i\|^2 = \|\bar{\mathrm{v}}_i\|^2 = \frac{1}{r}\|\mathrm{U}\mathrm{V}^\top\|_*$ for $i = 1,\ldots,r$.

Theorem 3.3 implies that for any network $h_{\mathrm{U,V}}$ there exists an equalized network $h_{\bar{\mathrm{U}},\bar{\mathrm{V}}}$ such that $h_{\bar{\mathrm{U}},\bar{\mathrm{V}}} = h_{\mathrm{U,V}}$. Hence, it is always possible to reduce the objective by equalizing the network, and a network $h_{\mathrm{U,V}}$ is globally optimal only if it is equalized.

Theorem 3.4. If (U,V) is a global optimum of Problem 6, then U,V are jointly equalized. Furthermore, it holds that

$$R(\mathrm{U},\mathrm{V}) = \frac{\lambda}{r}\left(\sum_{i=1}^{r}\|\mathrm{u}_i\|\|\mathrm{v}_i\|\right)^2 = \frac{\lambda}{r}\|\mathrm{U}\mathrm{V}^\top\|_*^2$$

Remark 3.5. As in the case of autoencoders with tied weights in Section 2, a complete characterization of the implicit bias of dropout is given by considering the global optimality in terms of the network, i.e. in terms of the product of the weight matrices $\mathrm{U}\mathrm{V}^\top$. Not surprisingly, even in the case of single hidden-layer networks, dropout promotes sparsity, i.e. favors low-rank weight matrices.

Theorem 3.6. For any $j\in[r]$, let $\kappa_j := \frac{1}{j}\sum_{i=1}^{j}\lambda_i(\mathrm{M})$. Furthermore, define $\rho := \max\{j\in[r]:\ \lambda_j(\mathrm{M}) > \frac{\lambda j\kappa_j}{r+\lambda j}\}$. Then, if $(\mathrm{U}_*,\mathrm{V}_*)$ is a global optimum of Problem 6, it satisfies that $\mathrm{U}_*\mathrm{V}_*^\top = \mathcal{S}_{\frac{\lambda\rho\kappa_\rho}{r+\lambda\rho}}(\mathrm{M})$.

4. Geometry of the Optimization Problem

While the focus in Section 2 and Section 3 was on understanding the implicit bias of dropout in terms of the global optima of the resulting regularized learning problem, here we focus on computational aspects of dropout as an optimization procedure. Since dropout is a first-order method (see Algorithm 1) and the landscape of Problem 4 is highly non-convex, we can perhaps only hope to find a *local* minimum, that too provided if the problem has no degenerate saddle points (Lee et al., 2016; Ge et al., 2015). Therefore, in this section, we pose the following questions: *What is the implicit bias of dropout in terms of local minima? Do local minima share anything with global minima structurally or in terms of the objective? Can dropout find a local optimum?*

For the sake of simplicity of analysis, we focus on the case of autoencoders with tied weight as in Section 2. We show in Section 4.1 that (a) local minima of Problem 4 inherit the same implicit bias as the global optima, i.e. all local minima are equalized. Then, in Section 4.2, we show that for sufficiently small regularization parameter, (b) there are no spurious local minima, i.e. all local minima are global, and (c) all saddle points are non-degenerate (see Definition 4.2).

4.1. Implicit bias in local optima

We begin by recalling that the loss $\ell(\mathrm{U},\mathrm{U})$ is rotation invariant, i.e. $\ell(\mathrm{UQ},\mathrm{UQ}) = \ell(\mathrm{U},\mathrm{U})$ for any rotation matrix Q. Now, if the weight matrix U were not equalized, then there exist indices $i,j\in[r]$ such that $\|\mathrm{u}_i\| > \|\mathrm{u}_j\|$. We show that it is easy to design a rotation matrix (equal to identity everywhere expect for columns i and j) that moves mass from u_i to u_j such that the difference in the norms of the corresponding columns of UQ decreases strictly while leaving the norms of other columns invariant. In other words, this rotation strictly reduces the regularizer and hence the objective. Formally, this implies the following result.

Lemma 4.1. All local optima of Problem 4 are equalized, i.e. if U is a local optimum, then $\|\mathrm{u}_i\| = \|\mathrm{u}_j\|\ \forall i,j\in[r]$.

Lemma 4.1 unveils a fundamental property of dropout. As

soon as we perform dropout in the hidden layer – *no matter how small the dropout rate* – all local minima become equalized.

4.2. Landscape properties

Next, we characterize the solutions to which dropout (i.e. Algorithm 1) converges. We do so by understanding the optimization landscape of Problem 4. Central to our analysis, is the following notion of *strict saddle property*.

Definition 4.2 (Strict saddle point/property)**.** Let $f : \mathcal{U} \to \mathbb{R}$ be a twice differentiable function and let $\mathrm{U} \in \mathcal{U}$ be a critical point of f. Then, U is a *strict saddle point* of f if the Hessian of f at U has at least one negative eigenvalue, i.e. $\lambda_{\min}(\nabla^2 f(\mathrm{U})) < 0$. Furthermore, f satisfies *strict saddle property* if all saddle points of f are strict saddle.

Strict saddle property ensures that for any critical point U that is not a local optimum, the Hessian has a significant negative eigenvalue which allows first order methods such as gradient descent (GD) and stochastic gradient descent (SGD) to escape saddle points and converge to a local minimum (Lee et al., 2016; Ge et al., 2015). Following this idea, there has been a flurry of works on studying the landscape of different machine learning problems, including low rank matrix recovery (Bhojanapalli et al., 2016), generalized phase retrieval problem (Sun et al., 2016), matrix completion (Ge et al., 2016), deep linear networks (Kawaguchi, 2016), matrix sensing and robust PCA (Ge et al., 2017) and tensor decomposition (Ge et al., 2015), making a case for global optimality of first order methods.

For the special case of no regularization (i.e. $\lambda = 0$; equivalently, no dropout), Problem 4 reduces to standard squared loss minimization which has been shown to have no spurious local minima and satisfy strict saddle property (see, e.g. (Baldi and Hornik, 1989; Jin et al., 2017)). However, the regularizer induced by dropout can potentially introduce new spurious local minima as well as degenerate saddle points. Our next result establishes that that is not the case, at least when the dropout rate is sufficiently small.

Theorem 4.3. For regularization parameter $\lambda < \frac{r\lambda_r(\mathrm{M})}{\sum_{i=1}^{r}\lambda_i(\mathrm{M}) - r\lambda_r(\mathrm{M})}$, (a) all local minima of Problem 4 are global, and (b) all saddle points are strict saddle points.

A couple of remarks are in order. First, Theorem 4.3 guarantees that any critical point U that is not a global optimum is a strict saddle point, i.e. $\nabla^2 f(\mathrm{U}, \mathrm{U})$ has a negative eigenvalue. This property allows first order methods, such as dropout given in Algorithm 1, to escape such saddle points. Second, note that the guarantees in Theorem 4.3 hold when the regularization parameter λ is sufficiently small. Assumptions of this kind are common in the literature (see, for example (Ge et al., 2017)). While this is a *sufficient* condition for the result in Theorem 4.3, it is not clear if it is *necessary*.

5. Matrix Factorization with Dropout

The optimization problem associated with learning a shallow network, i.e. Problem 6, is closely related to the optimization problem for matrix factorization. Recall that in matrix factorization, given a matrix $\mathrm{M} \in \mathbb{R}^{d_1 \times d_2}$, one seeks to find factors U, V that minimize $\ell(\mathrm{U}, \mathrm{V}) = \|\mathrm{M} - \mathrm{U}\mathrm{V}^\top\|_F^2$. Matrix factorization has recently been studied with dropout by Zhai and Zhang (2015); He et al. (2016) and Cavazza et al. (2018) where at each iteration of gradient descent on the loss function, the columns of factors U, V are dropped independently and with equal probability. Following Cavazza et al. (2018), we can write the resulting problem as

$$\min_{\mathrm{U}\in\mathbb{R}^{d_1\times r}, \mathrm{V}\in\mathbb{R}^{d_2\times r}} \|\mathrm{M} - \mathrm{U}\mathrm{V}^\top\|_F^2 + \lambda \sum_{i=1}^{r} \|\mathrm{u}_i\|^2 \|\mathrm{v}_i\|^2, \quad (7)$$

which is identical to Problem 6. However, there are two key distinctions. First, we are interested in stochastic optimization problem whereas the matrix factorization problem is typically posed for a given matrix. Second, for the learning problem that we consider here, it is unreasonable to assume access to the true model (i.e. matrix M). Nonetheless, many of the insights we develop here as well as the technical results and algorithmic contributions apply to matrix factorization. Therefore, the goal in this section is to bring to bear the results in Sections 2, 3 and 4 to matrix factorization.

We note that Theorem 3.6 and Theorem 3.3, both of which hold for matrix factorization, imply that there is a polynomial time algorithm to solve the matrix factorization problem. In order to find a global optimum of Problem 7, we first compute the optimal $\bar{\mathrm{M}} = \tilde{\mathrm{U}}\tilde{\mathrm{V}}^\top$ using shrinkage-thresholding operation (see Theorem 3.6). A global optimum $(\bar{\mathrm{U}}, \bar{\mathrm{V}})$ is then obtained by joint equalization of $(\tilde{\mathrm{U}}, \tilde{\mathrm{V}})$ (see Theorem 3.3) using Algorithm 2. The whole procedure is described in Algorithm 3. Few remarks are in order.

Remark 5.1 (Computational cost of Algorithm 3)**.** It is easy to check that computing $\rho, \bar{\mathrm{M}}, \tilde{\mathrm{U}}$ and $\tilde{\mathrm{V}}$ requires computing a rank-r SVD of M, which costs $O(d^2 r)$, where

Algorithm 3 Polynomial time solver for Problem 7

input Matrix $\mathrm{M} \in \mathbb{R}^{d_2 \times d_1}$ to be factorized, size of factorization r, regularization parameter λ

1: $\rho \leftarrow \max\{j \in [r] : \lambda_j(\mathrm{M}) > \frac{\lambda_j \kappa_j}{r + \lambda_j}\}$, where $\kappa_j = \frac{1}{j}\sum_{i=1}^{j}\lambda_i(\mathrm{M})$ for $j \in [r]$.
2: $\bar{\mathrm{M}} \leftarrow \mathcal{S}_{\frac{\lambda\rho\kappa_\rho}{r+\lambda\rho}}(\mathrm{M})$
3: $(\mathrm{U}, \Sigma, \mathrm{V}) \leftarrow \mathtt{svd}(\bar{\mathrm{M}})$
4: $\tilde{\mathrm{U}} \leftarrow \mathrm{U}\Sigma^{\frac{1}{2}}, \tilde{\mathrm{V}} \leftarrow \mathrm{V}\Sigma^{\frac{1}{2}}$
5: $\mathrm{Q} \leftarrow \mathtt{EQZ}(\tilde{\mathrm{U}})$ {Algorithm 2}
6: $\bar{\mathrm{U}} \leftarrow \tilde{\mathrm{U}}\mathrm{Q}, \bar{\mathrm{V}} \leftarrow \tilde{\mathrm{V}}\mathrm{Q}$

output $\bar{\mathrm{U}}, \bar{\mathrm{V}}$ {global optimum of Problem 7}

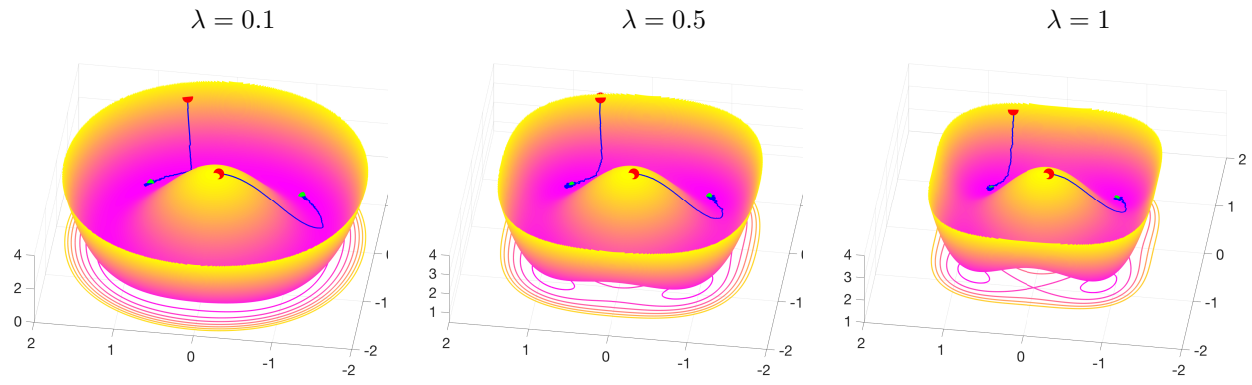

Figure 2: Convergence of dropout (Algorithm 1) from two different initialization (marked in red circles) to a global optimum of Problem 4 (marked in green circles), for the simple case of scalar M (one dimensional input and output) and $r = 2$. It can be seen that dropout quickly converges to a global optimum, which is equalized (i.e. weights are parallel to $(\pm 1, \pm 1)$) regardless of the value of the regularization parameter, $\lambda = 0.1$ (left), $\lambda = 0.5$ (middle) and $\lambda = 1.0$ (right).

$d = \max\{d_1, d_2\}$. Algorithm 2 entails computing $\mathrm{G_U} = \mathrm{U}^\top \mathrm{U}$, which costs $O(r^2 d)$ and the cost of each iterate of Algorithm 2 is dominated by computing the eigendecomposition which is $O(r^3)$. Overall, the computational cost of Algorithm 3 is $O(d^2 r + d r^2 + r^4)$.

Remark 5.2 (Universal Equalizer)**.** While Algorithm 2 is efficient (only linear in the dimension) for any rank r, there is a more effective equalization procedure when r is a power of 2. In this case, we can give a universal equalizer which works simultaneously for all matrices in $\mathbb{R}^{d \times r}$. Let $\mathrm{U} \in \mathbb{R}^{d \times r}$, $r = 2^k$, $k \in \mathbb{N}$ and let $\mathrm{U} = \mathrm{W}\Sigma\mathrm{V}^\top$ be its full SVD. The matrix $\tilde{\mathrm{U}} = \mathrm{UQ}$ is equalized, where $\mathrm{Q} = \mathrm{VZ}_k$ and

$$\mathrm{Z}_k := \begin{cases} 1 & k = 1 \\ 2^{\frac{-k+1}{2}} \begin{bmatrix} \mathrm{Z}_{k-1} & \mathrm{Z}_{k-1} \\ -\mathrm{Z}_{k-1} & \mathrm{Z}_{k-1} \end{bmatrix} & k > 1 \end{cases}.$$

Finally, we note that Problem 7 is an instance of regularized matrix factorization which has recently received considerable attention in the machine learning literature (Ge et al., 2016; 2017; Haeffele and Vidal, 2017). These works show that the saddle points of a class of regularized matrix factorization problems have certain "nice" properties (i.e. escape directions characterized by negative curvature around saddle points) which allow variants of first-order methods such as perturbed gradient descent (Ge et al., 2015; Jin et al., 2017) to converge to a local optimum. Distinct from that line of research, we completely characterize the set of global optima of Problem 7, and provide a polynomial time algorithm to find a global optimum.

The work most similar to the matrix factorization problem we consider in this section is that of Cavazza et al. (2018), with respect to which we make several important contributions: (I) Cavazza et al. (2018) characterize optimal solutions only in terms of the product of the factors, and not in terms of the factors themselves, whereas we provide globally optimal solutions in terms of the factors; (II) Cavazza et al. (2018) require the rank r of the desired factorization to be variable and above some threshold, whereas we consider fixed rank-r factorization for any r; (III) Cavazza et al. (2018) can only find low rank solutions using an adaptive dropout rate, which is not how dropout is used in practice, whereas we consider any fixed dropout rate; and (IV) we give an efficient poly time algorithm to find optimal factors.

6. Empirical Results

Dropout is a popular algorithmic technique used for avoiding overfitting when training large deep neural networks. The goal of this section is not to attest to the already well-established success of dropout. Instead, the purpose of this section is to simply confirm the theoretical results we showed in the previous section, as a proof of concept.

We begin with a toy example in order to visually illustrate the optimization landscape. We use dropout to learn a simple linear auto-encoder with one-dimensional input and output (i.e. a network represented by a scalar $\mathrm{M} = 2$) and a single hidden layer of width $r = 2$. The input features are sampled for a standard normal distribution. Figure 2 shows the optimization landscape along with the contours of the level sets, and a trace of iterates of dropout (Algorithm 1). The initial iterates and global optima (given by Theorem 2.4) are shown by red and green dots, respectively. Since at any global optimum the weights are equalized, the optimal weight vector in this case is parallel to the vector $(\pm 1, \pm 1)$. We see that dropout converges to a global minimum.

For a second illustrative experiment, we use Algorithm 1 to train a shallow linear network, where the input $\mathrm{x} \in \mathbb{R}^{80}$

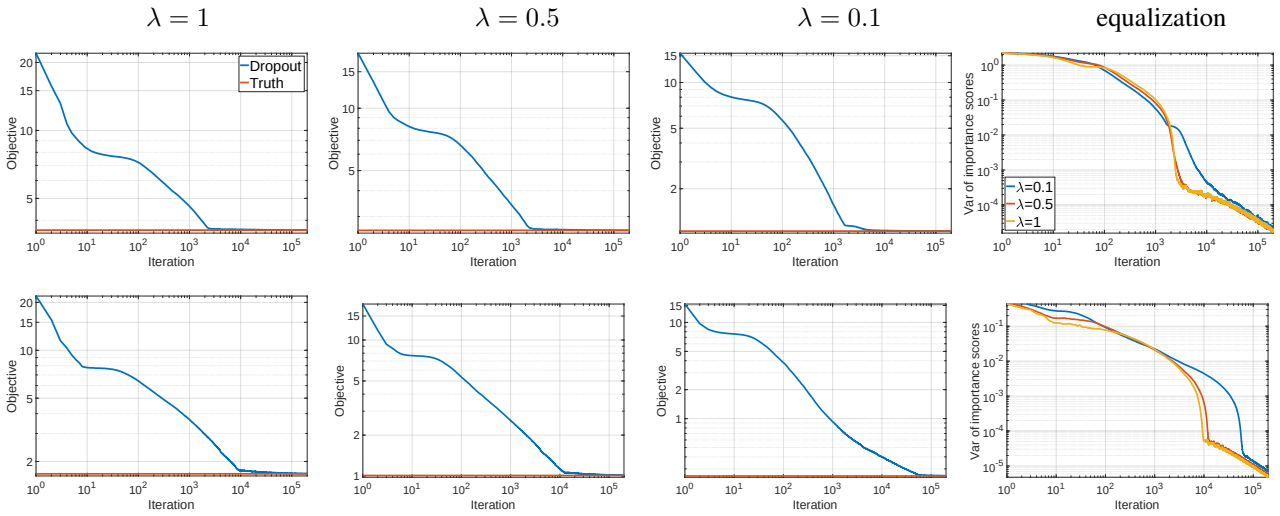

Figure 3: Dropout converges to global optima for different values of $\lambda \in \{0.1, 0.5, 1\}$ and different widths of the hidden layer $r = 20$ (top) and $r = 80$ (bottom). The right column shows the variance of the product of column-wise norms for each of the weight matrices. As can be seen, the weight matrices become equalized very quickly since variance goes to zero.

is distributed according to the standard Normal distribution. The output $\mathrm{y} \in \mathbb{R}^{120}$ is generated as $\mathrm{y} = \mathrm{M}\mathrm{x}$, where $\mathrm{M} \in \mathbb{R}^{120\times 80}$ is drawn randomly by uniformly sampling the right and left singular subspaces and with a spectrum decaying exponentially. Figure 3 illustrates the behavior of Algorithm 1 for different values of the regularization parameter ($\lambda \in \{0.1, 0.5, 1\}$), and for different sizes of factors ($r \in \{20, 80\}$). The curve in blue shows the objective value for the iterates of dropout, and the line in red shows the optimal value of the objective (i.e. objective for a global optimum found using Theorem 3.6). All plots are averaged over 50 runs of Algorithm 1 (averaged over different random initializations, random realizations of Bernoulli dropout, as well as random draws of training examples).

To verify that the solution found by dropout actually has equalized factors, we consider the following measure. At each iteration, we compute the "importance scores", $\alpha_t^{(i)} = \|\mathrm{u}_{ti}\|\|\mathrm{v}_{ti}\|,\ i \in [r]$, where u_{ti} and v_{ti} are the i-th columns of U_t and V_t, respectively. The rightmost panel of Figure 3 shows the variance of $\alpha_t^{(i)}$'s, over the hidden nodes $i \in [r]$, at each iterate t. Note that a high variance in α_t corresponds to large variation in the values of $\|\mathrm{u}_{ti}\|\|\mathrm{v}_{ti}\|$. When the variance is equal to zero, all importance scores are equal, thus the factors are equalized. We see that iterations of Algorithm 1 decrease this measure monotonically, and the larger the value of λ, the faster the weights become equalized.

7. Discussion

There has been much effort in recent years to understand the theoretical underpinnings of dropout (see Baldi and Sadowski (2013); Gal and Ghahramani (2016); Wager et al. (2013); Helmbold and Long (2015)). In this paper, we study the implicit bias of dropout in shallow linear networks. We show that dropout prefers solutions with minimal path regularization which yield strong capacity control guarantees in deep learning. Despite being a non-convex optimization problem, we are able to fully characterize the global optima of the dropout objective. Our analysis shows that dropout favors low-rank weight matrices that are equalized. This theoretical finding confirms that dropout as a procedure uniformly allocates weights to different subnetworks, which is akin to preventing co-adaptation.

We characterize the optimization landscape of learning autoencoders with dropout. We first show that the local optima inherit the same implicit bias as global optimal, i.e. all local optima are equalized. Then, we show that for sufficiently small dropout rates, there are no spurious local minima in the landscape, and all saddle points are non-degenerate. These properties suggest that dropout – as an optimization procedure – can efficiently converge to a globally optimal solution specified by our theorems.

Understanding dropout in shallow linear networks is a prerequisite for understanding dropout in deep learning. We see natural extensions of our results in two directions: 1) shallow networks with non-linear activation function such as rectified linear units (ReLU) which have been shown to enable faster training (Glorot et al., 2011) and are better understood in terms of the family of functions represented by ReLU-nets (Arora et al., 2018), and 2) exploring the global optimality in deeper networks, even for linear activations.

Acknowledgements

This research was supported in part by NSF BIGDATA grant IIS-1546482 and NSF grant IIS-1618485.

References

Abraham Adrian Albert and Benjamin Muckenhoupt. On matrices of trace zero. *The Michigan Mathematical Journal*, 4(1):1–3, 1957.

Raman Arora, Amitabh Basu, Poorya Mianjy, and Anirbit Mukherjee. Understanding deep neural networks with rectified linear units. In *International Conference on Learning Representations (ICLR)*, 2018. URL https://arxiv.org/abs/1611.01491.

Pierre Baldi and Kurt Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural networks*, 2(1):53–58, 1989.

Pierre Baldi and Peter J Sadowski. Understanding dropout. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2814–2822, 2013.

Srinadh Bhojanapalli, Behnam Neyshabur, and Nati Srebro. Global optimality of local search for low rank matrix recovery. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3873–3881, 2016.

Rich Caruana, Steve Lawrence, and C Lee Giles. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *Advances in Neural Information Processing Systems (NIPS)*, pages 402–408, 2001.

Jacopo Cavazza, Benjamin D. Haeffele, Connor Lane, Pietro Morerio, Vittorio Murino, and Rene Vidal. Dropout as a low-rank regularizer for matrix factorization. *Int. Conf. on Artificial Intelligence and Statistics (AISTATS)*, 2018.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Int. Conf. Machine Learning (ICML)*, 2016.

Rong Ge, Furong Huang, Chi Jin, and Yang Yuan. Escaping from saddle pointsonline stochastic gradient for tensor decomposition. In *Conf. Learning Theory (COLT)*, 2015.

Rong Ge, Jason D Lee, and Tengyu Ma. Matrix completion has no spurious local minimum. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

Rong Ge, Chi Jin, and Yi Zheng. No spurious local minima in nonconvex low rank problems: A unified geometric analysis. *arXiv preprint arXiv:1704.00708*, 2017.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proc. Intl. Conf. on Artificial Intelligence and Statistics (AISTATS)*, 2011.

Benjamin D Haeffele and Rene Vidal. Structured low-rank matrix factorization: Global optimality, algorithms, and applications. *arXiv preprint arXiv:1708.07850*, 2017.

Zhicheng He, Jie Liu, Caihua Liu, Yuan Wang, Airu Yin, and Yalou Huang. Dropout non-negative matrix factorization for independent feature learning. In *Int. Conf. on Computer Proc. of Oriental Languages*. Springer, 2016.

David P Helmbold and Philip M Long. On the inductive bias of dropout. *Journal of Machine Learning Research (JMLR)*, 16:3403–3454, 2015.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, pages 448–456, 2015.

Chi Jin, Rong Ge, Praneeth Netrapalli, Sham M Kakade, and Michael I Jordan. How to escape saddle points efficiently. *arXiv preprint arXiv:1703.00887*, 2017.

William Kahan. Only commutators have trace zero, 1999.

Kenji Kawaguchi. Deep learning without poor local minima. In *Adv in Neural Information Proc. Systems (NIPS)*, 2016.

Jason D Lee, Max Simchowitz, Michael I Jordan, and Benjamin Recht. Gradient descent converges to minimizers. *arXiv preprint arXiv:1602.04915*, 2016.

Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. Norm-based capacity control in neural networks. In *Conf. on Learning Theory (COLT)*, pages 1376–1401, 2015.

Steven J Nowlan and Geoffrey E Hinton. Simplifying neural networks by soft weight-sharing. *Neural computation*, 4 (4):473–493, 1992.

Nathan Srebro, Jason Rennie, and Tommi S Jaakkola. Maximum-margin matrix factorization. In *Advances in Neural Information Processing Systems (NIPS)*, 2005.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research (JMLR)*, 15(1), 2014.

Ju Sun, Qing Qu, and John Wright. A geometric analysis of phase retrieval. In *IEEE International Symposium on Information Theory (ISIT)*, pages 2379–2383, 2016.

Stefan Wager, Sida Wang, and Percy S Liang. Dropout training as adaptive regularization. In *Advances in Neural Information Processing Systems (NIPS)*, 2013.

Shuangfei Zhai and Zhongfei Zhang. Dropout training of matrix factorization and autoencoder for link prediction in sparse graphs. In *Proc. of SIAM International Conference on Data Mining (ICDM)*, pages 451–459, 2015.