Channel Normalization in Convolutional Neural Networks avoids Vanishing Gradients

Zhenwei Dai[†] and Reinhard Heckel[*]

[*]Dept. of Electrical and Computer Engineering and Dept. of Statistics[†], Rice University

Abstract

Normalization layers are widely used in deep neural networks to stabilize training. In this paper, we consider the training of convolutional neural networks with gradient descent on a single training example. This optimization problem arrises in recent approaches for solving inverse problems such as the deep image prior or the deep decoder. We show that for this setup, channel normalization, which centers and normalizes each channel individually, avoids vanishing gradients, whereas without normalization, gradients vanish which prevents efficient optimization. We then demonstrate that this effect prevails in deep single-channel linear convolutional networks, and analytically show that without channel normalization, gradient descent takes at least exponentially many steps to come close to an optimum. Contrary, with channel normalization, the gradients remain bounded, thus avoiding exploding gradients.

1 Introduction

Deep learning and in particular convolutional neural networks have significantly improved the state-of-the-art in computer vision, image generation, and computational imaging, among many other fields. Deep neural networks are typically trained using first order methods such as gradient descent and the stochastic gradient methods. However, the corresponding loss function is non-convex and therefore, depending on the initialization, convergence to an optimum is not guaranteed, and first order methods sometimes suffer from vanishing or exploding gradients.

Normalization layers are widely used to avoid vanishing or exploding gradients, stabilize training, enable learning with higher rates, and enable faster convergence. The perhaps most popular technique is batch normalization [IS15]; but a number of (often closely related) variations and alternatives have been proposed such as layer normalization [LB+16], weight normalization [SK16], and instance normalization [Uly+16].

A variety of recent works have proposed different explanations for the success of normalization layers. While the original batch normalization paper [IS15] suggested that batch normalization aids optimization by reducing a quantity called internal covariate shift, Santurkar et al. [San+18] provide empirical evidence contradicting this explanation, and instead explain that batch normalization reparameterizes the underlying optimization problem and make its landscape significantly smoother. Kohler et al. [Koh+18] linked batch normalization to weight normalization [SK16], and pointed out that batch normalization accelerates the training process by splitting the optimization task into optimizing the length and direction of the parameters separately. We hasten to add that whether normalization layers are useful or not depends strongly on the architecture and initialization. For example carefully initialized deep residual networks can be trained successfully without any normalization layers [Zha+19].

In this paper, we study normalization techniques in the context of a convolutional generator network. Specifically, we study channel normalization, which is a special case of a number of the above mentioned normalization techniques. Channel normalization standardizes each channel in a convolutional neural network, individually for each training example, and scales and shifts the resulting vector with a (learned) scalar. Channel normalization is equivalent to batch normalization for a single training example (then the batch size is one) and to instance normalization.

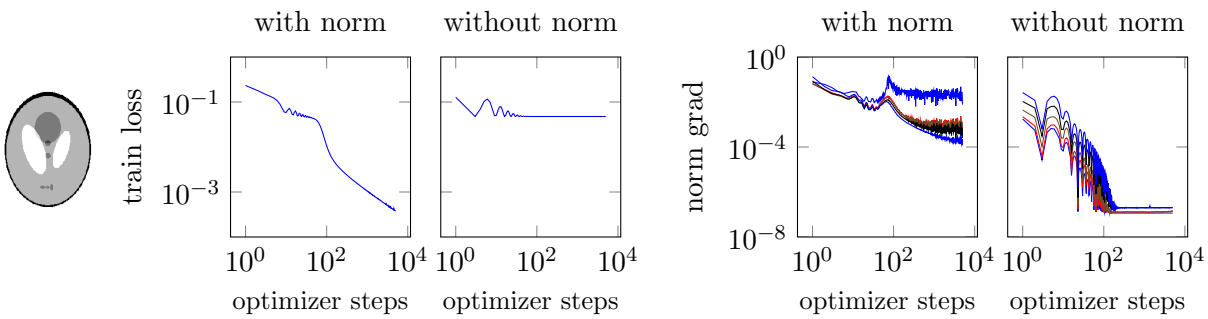

Figure 1: The train loss and the norm of the gradient in each of the 5 layers of a convolutional generator network with and without channel normalization for fitting the phantom MRI image: Without channel normalization the gradients vanish before gradient descent reaches a good solution.

We first demonstrate that channel normalization in convolutional networks avoids exploding and vanishing gradients in the context of training a convolutional network on a single training example, as common for solving inverse problems [HH18; Uly+18]. Contrary, without channel normalization, the optimization does not converge to an optimum in a reasonable number of iterations. We then show analytically, for a special case of linear convolutional networks, that without channel normalization, gradient descent requires at least exponentially many steps to converge under mild initialization conditions.

The aforementioned works [IS15; San+18; Koh+18] have studied normalization techniques by focusing on shallow networks (i.e., networks with one hidden layer), since analytical gradient expressions for deep networks with non-linearities are intractable. Here, we sidestep this hurdle by exploring a simple model–linear convolutional neural networks with a single channel. Studying such a simple model is justified by noting that even for this simple model, normalization is crucial for fast convergence.

2 Channel normalization in convolutional networks

We start by introducing the channel normalization operation and then show empirically that it is critical for the optimization of convolutional generators trained on a single example.

The channel normalization operation normalizes each channel of a convolutional network individually. Let $\mathbf{z}_{ij}$ be the input of j-th channel and the i-th layer. Channel normalization performs the transformation $\mathbf{z}'_{ij} = \frac{\mathbf{z}_{ij} - \text{mean}(\mathbf{z}_{ij})}{\sqrt{\text{var}(\mathbf{z}_{ij}) + \epsilon}} \gamma_{ij} + \beta_{ij}$, where mean and var compute the empirical mean and variance, $\gamma_{i,j}$ and β_{ij} are parameters learned independently for each channel, and ϵ is a fixed small constant for numerical stability.

We consider a variant of the deep decoder introduced in [HH18], which in its original form consists of bilinear upsampling and convolutional operations, followed by channel normalization and ReLU layers. The network works well for image compression and for a variety of inverse problems, when trained or fitted to a single image only. Specifically, we consider an extremely simple convolutional generator consisting of $d = 5$ 3x3 convolutional layers, followed by channel normalization and ReLU activation functions. Each layer has $k = 32$ channels, and the last layer is a 1x1 convolutional layer mapping the k channels to a single, 256x256 grayscale output image. The input to the network is a 32x256x256 volume that is chosen randomly and is fixed (i.e., we do not optimize over the input). Given an image $\mathbf{x}^*$ we then fit the parameters of the network (i.e., the weights) by minimizing the loss $L(\mathbf{C}) = \|G(\mathbf{C}) - \mathbf{x}^*\|_2^2$ with respect to the network parameters

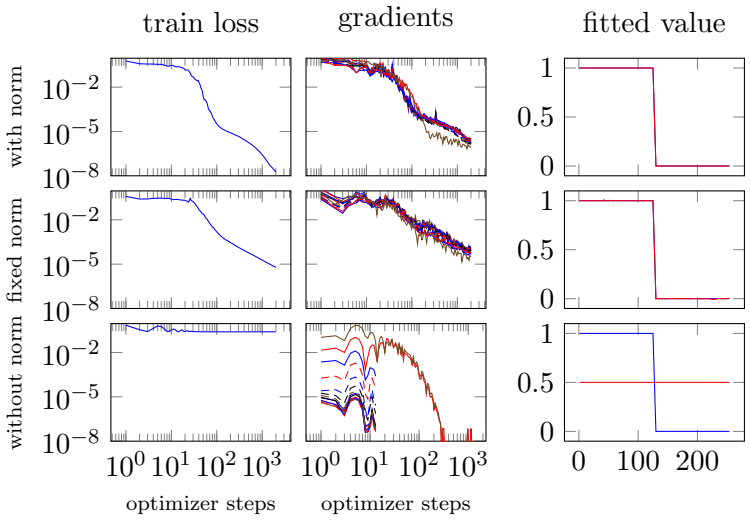

Figure 2: The left panel shows the training error and the norms of the gradients of each layer verses the iteration steps. The right panel shows the fitted response (red) and true response (blue).

C using plain gradient descent with a fixed stepsize.

Figure 1 shows the results for the phantom MRI image for a network with and without channel normalization. It can be seen that with channel normalization, the training loss converges rapidly and the gradients do not vanish. Contrary, without normalization, the network does not converge to a small error (even though it has the capacity to represent the image exactly), and the gradients vanish. This effect is not specific to the image (we have reproduced it using 100 randomly chosen images from imagenet), and it is also reproducible for a number of related convolutional generators, for example those including upsampling operations.

3 Isolating the effect of channel normalization

We next show that to achieve the stabilizing effect of channel normalization, the trainable coefficients γ_{ij} and β_{ij} do not need to be learned and can be set to one and zero, respectively. We also demonstrate that even for linear networks, channel normalization is critical to avoid vanishing gradients. This justifies studying linear networks in the next section.

Multiple Channels CNN: We consider a network again only consisting of convolutional layers followed by channel normalization and ReLU activation functions. We set the dimension of input/output vector to $n = 256$, the number of channels to $k = 4$, number of hidden layers to $d = 12$, and convolution kernel size to 3. The entries of the input vector $\mathbf{x}$ are sampled from a standard uniform distribution. As before, we minimize the least squares error with respect to the weight parameters using gradient descent with fixed step size. We consider three different normalization operations: the original channel normalization, a variant where $\gamma_{ij} = 1$ and $\beta_{ij} = 0$ (called fixed norm), and no normalization and fit a simple step function. The results in Figure 2 show that both versions of normalization enable efficient optimization, whereas without channel normalization the training error stops improving after a few iterations and the gradients vanish.

Single Channel Linear CNN: Next, we consider an even simpler network with only one channel and without activation functions. Without normalization, the network is linear. We set the dimension of input/output vector to $n = 64$, number of hidden layers $d = 10$, and convolution kernel size $k = 9$. The results are very similar to the previous experiment in Figure 2 (see Figure 3 in the appendix), and demonstrates the critical role of channel normalization.

4 Theoretical analysis

From the previous section, we know that channel normalization avoids vanishing gradients even for linear, one-layer convolutional neural networks, and that the scale and shift parameters can be set to one and zero. In this section, we provide theoretical justification for the stabilizing effect of channel normalization.

Throughout this section, we consider a single channel linear convolutional neural network with d layers, with output given as $f(\mathbf{x}, \mathbf{w}) = \prod_{i=1}^{d} \mathbf{W}_i \mathbf{x}$, where $\mathbf{W}_i \in \mathbb{R}^{n \times n}$ are circulant matrices implementing the convolution operation, and $\mathbf{w} = (\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_d)$ is the set of weights, given by the first columns of the respective circulant matrices $\mathbf{W}_i, \ldots, \mathbf{W}_d$ (which define all other entries of the matrices). We study gradient descent applied to the squared loss function $L(\mathbf{w}, \mathbf{x}, \mathbf{y}) = \frac{1}{2}\|\mathbf{y} - f(\mathbf{x}, \mathbf{w})\|_2^2$.

We start by showing that without channel normalization optimization is provably hard. Specifically, we show that without channel normalization, gradient descent needs at least exponentially many steps to converge under a standard initialization scheme.

Theorem 1. *Suppose that the signal* $\mathbf{y}$ *doesn't vanish, i.e.,* $\|\mathbf{y}\|/\|\mathbf{x}\| \geq dn^{d/2}\tau$*, where* τ *is a constant. Moreover, suppose that* $\mathbf{y}$ *is in the range of the generator* f *and that the initial weights are Gaussian random variables with zero mean and covariance matrix* $(1/n^2)\mathbf{I}$*). Then gradient descent with constant stepsize* $\eta \leq exp(\frac{1}{2}cd)$ *runs at least for* $\exp(\Omega(d))$ *steps to be* c' *close to optimal with probability larger than* $1 - \exp(-\Omega(d))$*. Here,* c *and* c' *are constants independent of* d*.*

The proof, deferred to the appendix, relies on diagonalizing the circulant matrices $\mathbf{W}_i$ using the Fourier transform. Then, the optimization problem reduces to n one-dimensional problems, and we can build on results by Shamir [Sha18] on the hardness of optimizing one-dimensional deep neural networks.

Theorem 1 shows that the number of steps to come close to the optimum is at least exponential in the network depth d, even when the stepsize is large (exponential in d). This can be interpreted as a case of gradient vanishing. Also not that if we initialize the weight away from 0, with another initialization scheme, then the norms of the gradients increase exponentially fast with the network depth, and the network becomes difficult to optimize due to exploding gradients.

Next, we comment on the effect the channel normalization on the gradients. Since our experiments have shown that fixing scale and shift parameters does not weaken the performance of channel normalization, we focus on the case with $\gamma_{ij} = 1$ and $\beta_{ij} = 0$. With the Fourier transformation, the circulant matrices $\mathbf{W}_i$ can be diagonalized (see appendix). As shown in the appendix, provided the input and output vectors $\mathbf{x}$ and $\mathbf{y}$ have been centered, the loss function with channel normalization $L_N(\mathbf{W}, \mathbf{x}, \mathbf{y})$ can written as

$$L_N(\mathbf{W}, \mathbf{x}, \mathbf{y}) = \frac{1}{2}\left\|\mathbf{F}^H\mathbf{y} - w_{d+1}\sqrt{n}\left(\prod_{i=1}^{d}\operatorname{diag}(\mathbf{F}^H\mathbf{w}_i)\right)\mathbf{F}^H\mathbf{x}/\left\|\prod_{i=1}^{d}\operatorname{diag}(\mathbf{F}^H\mathbf{w}_i)\mathbf{F}^H\mathbf{x}\right\|\right\|_2^2$$

where $\mathbf{w}_i$ is the first column of $\mathbf{W}_i$. To ensure the fitted function can exhaust $\mathbb{R}^n$, we added an extra scaling parameter $w_{d+1} \in \mathbb{R}$ to the fitted function. Now, we can analyze the gradient with channel normalization (fixed scale and shift).

Based on the expression above, it can be shown that the gradients of the loss function with normalization, given by $\nabla_{\mathbf{w}_k} L_N(\mathbf{W}, \mathbf{x}, \mathbf{y})$ are bounded at initialization (shown in an extended version of this manuscript). Thus, channel normalization avoid gradient explosition. Contrary, without normalization, the order of the gradient norm increases in the network depth. Specifically, for deep linear CNNs, the norm of the gradients increase/decrease exponentially fast as the order of network, which can easily lead to gradient explosion/vanishing.

References

[HH18] R. Heckel and P. Hand. "Deep decoder: Concise image representations from untrained non-convolutional networks". In: *arXiv preprint arXiv:1810.03982* (2018).

[IS15] S. Ioffe and C. Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *arXiv preprint arXiv:1502.03167* (2015).

[Koh+18] J. Kohler, H. Daneshmand, A. Lucchi, M. Zhou, K. Neymeyr, and T. Hofmann. "Exponential convergence rates for Batch Normalization: The power of length-direction decoupling in non-convex optimization". In: *arXiv:1805.10694* (2018).

[LB+16] J. Lei Ba, J. R. Kiros, and G. E. Hinton. "Layer normalization". In: *arXiv preprint arXiv:1607.06450* (2016).

[SK16] T. Salimans and D. P. Kingma. "Weight normalization: A simple reparameterization to accelerate training of deep neural networks". In: *Advances in Neural Information Processing Systems*. 2016, pp. 901–909.

[San+18] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry. "How does batch normalization help optimization?" In: *Advances in Neural Information Processing Systems*. 2018, pp. 2488–2498.

[Sha18] O. Shamir. "Exponential convergence time of gradient descent for one-dimensional deep linear neural networks". In: *arXiv preprint arXiv:1809.08587* (2018).

[Uly+18] D. Ulyanov, A. Vedaldi, and V. Lempitsky. "Deep image prior". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 9446–9454.

[Uly+16] D. Ulyanov, A. Vedaldi, and V. Lempitsky. "Instance normalization: The missing ingredient for fast stylization". In: *arXiv preprint arXiv:1607.08022* (2016).

[Zha+19] H. Zhang, Y. N. Dauphin, and T. Ma. "Fixup initialization: Residual learning without normalization". In: *arXiv preprint arXiv:1901.09321* (2019).

Appendix

5 Convergence for linear one layer convolutional networks

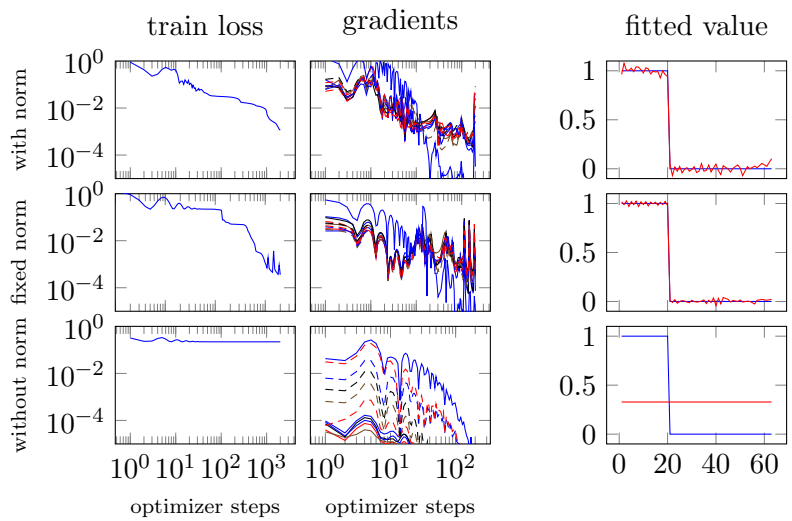

Figure 3: The left panel shows the training error and gradients of different layers verses the iteration steps. The right panel shows the fitted response (red) and true response (blue).

6 Proofs

Our proofs rely on diagonalizing the circulant matrices using the Fourier transform.

6.1 Linear single channel CNNs in the Fourier domain

Since the matrices $\mathbf{W}_i$ are circulant, they can be diagonalized with the Fourier transformation, which makes the loss function separable. Hence, the loss function linear CNN can be transformed to the sum of the loss functions of single channel deep linear neural networks. With $\mathbf{W}_i = \mathbf{F}\text{diag}(\sqrt{n}\mathbf{F}^H\mathbf{w}_i)\mathbf{F}^H$, where $\mathbf{F}$ is the unitary $n \times n$ discrete Fourier transform matrix and $\mathbf{w}_i$ is the first column of $\mathbf{W}_i$, the network output can be expressed as

$$f(\mathbf{x}, \mathbf{W}) = \prod_{i=1}^{d} \mathbf{W}_i\mathbf{x} = \prod_{i=1}^{d} \mathbf{F}\text{diag}(\sqrt{n}\mathbf{F}^H\mathbf{w}_i)\mathbf{F}^H\mathbf{x} = \sqrt{n}^d\mathbf{F}\left(\prod_{i=1}^{d}\text{diag}(\mathbf{F}^H\mathbf{w}_i)\right)\mathbf{F}^H\mathbf{x}.$$

With this expression, the loss function becomes

$$
\begin{aligned}
L(\mathbf{W};\mathbf{x},\mathbf{y}) = \frac{1}{2}\|\mathbf{y} - f(\mathbf{x},\mathbf{W})\|^2 &= \frac{1}{2}\left\|\mathbf{y} - \sqrt{n}^d \mathbf{F}\left(\prod_{i=1}^{d} \operatorname{diag}(\mathbf{F}^H \mathbf{w}_i)\right)\mathbf{F}^H\mathbf{x}\right\|^2 \\
&= \frac{1}{2}\left\|\mathbf{F}^H\mathbf{y} - \sqrt{n}^d \left(\prod_{i=1}^{d} \operatorname{diag}(\mathbf{F}^H \mathbf{w}_i)\right)\mathbf{F}^H\mathbf{x}\right\|^2 \\
&= \frac{1}{2}\sum_{j=1}^{n}\left\|\mathbf{f}_j^H\mathbf{y} - \sqrt{n}^d \left(\prod_{i=1}^{d} \mathbf{f}_j^H \mathbf{w}_i\right)\mathbf{f}_j^H\mathbf{x}\right\|^2,
\end{aligned}
$$

where $\mathbf{f}_j$ is the j-th column of the Fourier matrix $\mathbf{F}$.

6.2 Proof of Theorem 1

In this section, we show that gradient descent takes exponentially many steps to converge under (standard) Xavier initialization.

Assume the kernel size of i-th convolution layer is p ($p \geq 1$). Thus, $\mathbf{w}_i \in \mathbb{S}_p$, where $\mathbb{S}_p = \{\mathbf{z} = (z_1, z_2, \ldots, z_n) | z_1, \ldots, z_p \in \mathbb{R}, z_{p+1}, \ldots, z_n = 0\}$.

Assumption 1. *Assume $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_d$ are drawn independently and the first p entries of $\mathbf{w}_i$ are drawn i.i.d from a distribution with mean $\mu = 0$ and variance $\sigma = \frac{1}{n^2}$, which also satisfies*

1. $\mathrm{P}\left[\|\mathbf{w}_i\| \leq t\right] \leq c_1 t$
2. $\mathbb{E}\left[\|\mathbf{w}_i\|\right] \leq 1 - c_2$

where the constants $c_1, c_2 > 0$ are independent of d.

The assumptions hold for some widely used distributions, like the distribution $\mathcal{N}(0, \frac{1}{n^2})$ in the statement of the theorem.

Next, we show that with the initialization satisfying Assumption 1, gradient descent takes exponential time to converge with high probability.

The key idea of the proof of Theorem 1 is to show that if we start from such a random initialization $\mathbf{w} = (\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_d)$, then gradient descent must take exponential many steps to escape a ball of radius r around $\mathbf{w}$, defined as

$$
\mathcal{B}(\mathbf{w}, r) = \left\{\mathbf{v} = (\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_d) | \sum_{i=1}^{d} \|\mathbf{v}_i - \mathbf{w}_i\|^2 \leq r^2\right\}.
$$

We then show that the function value evaluated at any point inside the ball is sub-optimal, i.e., for $\mathbf{v} \in \mathcal{B}(\mathbf{w}, r)$, $L(\mathbf{v}) \geq c'$, where c' is a constant. This concludes the proof.

To show that there exist a radius $r > 0$ such that gradient descent takes at least exponential steps to escape the ball $\mathcal{B}(\mathbf{w}, r)$, we first note that the number of iterations required to escape a ball of radius r is at least $r/(\eta \sup_{\mathbf{v}\in\mathcal{B}(\mathbf{w},r)} \|\nabla L(\mathbf{v})\|)$, since at each step, gradient descent can at most move by $\eta \sup_{\mathbf{v}\in\mathcal{B}(\mathbf{w},r)} \|\nabla L(\mathbf{v})\|$ away from the initialization.

The following lemma provides an upper bound on $\|\nabla L(\mathbf{v})\|$, which enables us to show that the number of iterations must be large.

Lemma 1. *For all* $\mathbf{v} \in B(\mathbf{w}, r)$*, if* $\mathbf{w}$ *satisfies, for some* $\alpha, \delta > 0$*,* $\max_k \left(\prod_{i \neq k} \|\mathbf{w}_i\|\right) \leq \alpha$ *and* $\min_i \|\mathbf{w}_i\| \geq \delta$*, it holds that* $\|\nabla L(\mathbf{v})\| \leq \|D(\mathbf{v})\| \left(\alpha\sqrt{d}\right) \exp\left(\frac{\sqrt{d}}{\delta} r\right)$*, where* $L(\mathbf{v}) = \frac{1}{2}\|D(\mathbf{v})\|^2$ *and* $D(\mathbf{v}) = \mathbf{F}^H\mathbf{y} - \sqrt{n}^d \prod_{i=1}^d \operatorname{diag}(\mathbf{F}^H\mathbf{v}_i)\mathbf{F}^H\mathbf{x}$*.*

Lemma 1 guarantees that in a ball $B(\mathbf{w}, r)$ around the initialization, the corresponding gradient is strictly upper bounded. So, for a stepsize not too large, the update of each step is also upper bounded. Then, we show with high probability, that we can find a radius r that is much larger than the updates in each step.

Evoking Lemma 1, the number of steps required to escape the ball $\mathcal{B}(\mathbf{w}, r)$ is at least $\frac{r}{O(\eta\alpha\sqrt{d}\exp(\frac{\sqrt{d}}{\delta}r))}$. Next, we show that we can upper bound α and lower bound δ with high probability given Assumption 1. And under this condition, there exists r such that gradient descent takes exponential steps to escape the ball $B(\mathbf{w}, r)$.

With Assumption 1 and Markov's inequality, we have, for $t > 0$, that

$$\Pr\left(\left\|\prod_{i \neq k}(\mathbf{f}_j^H\mathbf{w}_i)\right\| \leq t\right) \leq \frac{\mathbb{E}(\prod_{i \neq k}\left\|\mathbf{f}_j^H\mathbf{w}_i\right\|)}{t} \leq \frac{(1-c_2)^{d-1}}{t}.$$

Let $\exp(-4c) = 1 - c_2$, $t = \exp(-2cd)$, where $c > 0$. Then we obtain

$$\Pr\left(\left\|\prod_{i \neq k}(\mathbf{f}_j^H\mathbf{w}_i)\right\| \leq \exp(-2cd)\right) \leq \frac{\exp(-4c(d-1))}{\exp(-2cd)} = O(\exp(-2cd)). \tag{1}$$

Again by Assumption 1, it holds for all i that $\Pr(\|\mathbf{w}_i\| \leq \exp(-cd)) \leq O(\exp(-cd))$. Then, by the union bound,

$$\Pr(\min_i \|\mathbf{w}_i\| \leq \exp(-cd)) \leq \sum_{i=1}^d \Pr\left(\|\mathbf{w}_i\| \leq \exp(-cd)\right) \leq O(d\exp(-cd)). \tag{2}$$

Finally, by Assumption 1, Markov's inequality, and a union bound

$$\Pr(\max_i \|\mathbf{w}_i\| \geq \exp(cd)) \leq \sum_{i=1}^d \Pr(\|\mathbf{w}_i\| \geq \exp(cd)) \leq O(d\exp(-cd)). \tag{3}$$

Thus, by a union bound, with probability at least $1 - O(\exp(-2cd)) - O(d\exp(-cd)) - O(d\exp(-cd)) = 1 - O(d\exp(-cd))$, the conditions i) $\left\|\prod_{i \neq k}(\mathbf{f}_j^H\mathbf{w}_i)\right\| \geq \exp(-2cd)$, ii) $\min_i \|\mathbf{w}_i\| \geq \exp(-cd)$, and ii) $\max_i \|\mathbf{w}_i\| \leq \exp(cd)$ are satisfied simultaneously.

With i-iii) satisfied, the number of steps in $\mathcal{B}(\mathbf{w}, r)$ is at least $\frac{r}{O(\eta\alpha\sqrt{d}\exp(\frac{\sqrt{d}}{\delta}r))} = \frac{r}{O(\exp(-\frac{3}{2}cd + \frac{\sqrt{d}}{\delta}r))}$. If we let $r = O(\frac{\delta}{\sqrt{d}}) = O(\frac{\exp(-cd)}{\sqrt{d}})$, then $\frac{r}{O(\exp(-\frac{3}{2}cd + \frac{\sqrt{d}}{\delta}r))} = O(\exp(\frac{1}{2}cd))$, which increases exponentially in d.

It remains to prove that for any $\mathbf{v} \in \mathcal{B}(\mathbf{w}, r)$ (with r as chosen above), the loss function $L(\mathbf{v})$ is lower bounded away from the global minimum. Towards this goal, we note that, for $\mathbf{v} \in \mathcal{B}(\mathbf{w}, r)$,

$$\max_i \|\mathbf{v}_i\| \leq \max_i \|\mathbf{w}_i\| + r = O(\exp(cd)).$$

Hence,

$$\left\|\prod_{i=1}^{d}\mathbf{f}_j^H\mathbf{v}_i\right\| \le \prod_{i=1}^{d}\|\mathbf{f}_j^H\mathbf{v}_i\| \le \left(\prod_{i\neq k}\|\mathbf{f}_j^H\mathbf{v}_i\|\right)\left(\max_k\|\mathbf{v}_k\|\right) \le O(\exp(-cd)).$$

Therefore, when d is large, $\left\|\prod_{i=1}^{d}\mathbf{f}_j^H\mathbf{v}_i\right\| \le 1$. From the assumption $\|\mathbf{y}\| - \|\mathbf{x}\|d\sqrt{n}^d \ge \tau$, there exists $t \in [n]$ such that $\left\|\mathbf{f}_t^H\mathbf{y}\right\|_2 - \sqrt{n}^d\left\|\mathbf{f}_t^H\mathbf{x}\right\|_2 \ge \tau$. So, we have

$$\begin{aligned}\left\|\mathbf{f}_t^H\mathbf{y} - \sqrt{n}^d\prod_{i=1}^{d}\left(\mathbf{f}_t^H\mathbf{v}_i\right)\mathbf{f}_t^H\mathbf{x}\right\|_2 &\ge \left\|\mathbf{f}_t^H\mathbf{y}\right\|_2 - \left\|\sqrt{n}^d\prod_{i=1}^{d}\left(\mathbf{f}_t^H\mathbf{v}_i\right)\mathbf{f}_t^H\mathbf{x}\right\|_2 \\ &\ge \left\|\mathbf{f}_t^H\mathbf{y}\right\|_2 - \sqrt{n}^d\prod_{i=1}^{d}\left\|\mathbf{f}_t^H\mathbf{v}_i\right\|_2\left\|\mathbf{f}_t^H\mathbf{x}\right\|_2 \\ &\ge \left\|\mathbf{f}_t^H\mathbf{y}\right\|_2 - \sqrt{n}^d\left\|\mathbf{f}_t^H\mathbf{x}\right\|_2 > \tau\end{aligned}$$

Thus, for $\mathbf{v} \in \mathcal{B}(\mathbf{w}, r)$, the loss function obeys $L(\mathbf{v}) = \sum_{j=1}^{n}\left\|\mathbf{f}_j^H\mathbf{y} - \sqrt{n}^d\prod_{i=1}^{d}\left(\mathbf{f}_j^H\mathbf{v}_i\right)\mathbf{f}_j^H\mathbf{x}\right\|_2 > \tau$ and is thus lower bounded away from zero-training error.

6.2.1 Proof of Lemma 1

Let $\mathbf{v} \in \mathcal{B}(\mathbf{w}, r)$ as defined previously. We have

$$\|\nabla L(\mathbf{v})\|^2 \le \sum_{k=1}^{d}\|\nabla_{\mathbf{v}_k}L(\mathbf{v})\|^2 \quad = \sum_{k=1}^{d}\|D(\mathbf{v})\|\left(\prod_{i\neq k}\|\mathbf{v}_i\|\right) \le \sum_{k=1}^{d}\|D(\mathbf{v})\|\left(\max_k\prod_{i\neq k}\|\mathbf{v}_i\|\right). \tag{4}$$

The lemma now follows from $\max_k\left(\prod_{i\neq k}\|\mathbf{v}_i\|\right) \le \left(\alpha\sqrt{d}\right)\exp\left(\frac{\sqrt{d}}{\delta}r\right)$.

Define $\mathbf{r}_i = \mathbf{v}_i - \mathbf{w}_i$, for notational convenience, and note that $\sum_{i=1}^{n}\|\mathbf{r}_i\|^2 \le r^2$ since $\mathbf{v} \in \mathcal{B}(\mathbf{w}, r)$. Then, we have

$$\begin{aligned}\prod_{i\neq k}\|\mathbf{v}_i\| &= \prod_{i\neq k}\|\mathbf{w}_i + \mathbf{r}_i\| \\ &\le \left(\prod_{i\neq k}\|\mathbf{w}_i\|\right)\left[\prod_{i\neq k}\left(1 + \frac{\|\mathbf{r}_i\|}{\|\mathbf{w}_i\|}\right)\right] \\ &\le \left(\prod_{i\neq k}\|\mathbf{w}_i\|\right)\left[\prod_{i\neq k}\left(1 + \frac{\|\mathbf{r}_i\|}{\delta}\right)\right] \\ &= \left(\prod_{i\neq k}\|\mathbf{w}_i\|\right)\exp\left[\sum_{i\neq k}\log\left(1 + \frac{\|\mathbf{r}_i\|}{\delta}\right)\right] \le \left(\prod_{i\neq k}\|\mathbf{w}_i\|\right)\exp\left[\sum_{i\neq k}\frac{\|r_i\|}{\delta}\right]\end{aligned} \tag{5}$$

We also have

$$r^2 \ge \sum_{i=1}^{d}\|\mathbf{r}_i\|^2 \ge \frac{\left(\sum_{i=1}^{d}\|\mathbf{r}_i\|_1\right)^2}{d}. \tag{6}$$

Inserting Equation (5) into Equation (4), we get

$$\prod_{i\neq k} \|\mathbf{v}_i\| \leq \left(\prod_{i\neq k} \|\mathbf{w}_i\|\right) \exp\left[\sum_{i=1}^{d} \frac{\|r_i\|}{\delta}\right] \leq \left(\prod_{i\neq k} \|\mathbf{w}_i\|\right) \exp\left(\frac{\sqrt{d}r}{\delta}\right) \leq \alpha \cdot \exp\left(\frac{\sqrt{d}r}{\delta}\right).$$

Application of this inequality in (**??**) yields $\|\nabla L(\mathbf{v})\| \leq \sqrt{d} \cdot \max_k \|\nabla_k L(\mathbf{v})\| \leq \|D(\mathbf{v})\| \left(\alpha\sqrt{d}\right) \exp\left(\frac{\sqrt{d}}{\delta} r\right)$, which concludes the proof.