Minimal Projection Constants and Deep Learning

Ryan Malthaner Weston Baines

Abstract

Minimum projection constants are an interesting problem to use in order to study the capacity of Neural Networks to learn functions which do not seem to exhibit any sort of natural structure. In the case of certain finite dimensional normed vector spaces, the minimum projection constant varies continuously with the choice of subspace. Several thousand networks are trained and it is determined experimentally that fully connected networks fail to learn the minimum projection constants. Evidence is found to support the viability of Neural Networks with skip connections to learn the minimum projection constants.

1. Introduction

Given a Banach space $(X, ||\cdot||_X)$ and a closed linear subspace $Y \subset X$, a map $P : X \to Y$ satisfying $P|_Y = I_Y$ is called a projection. Note that the set of projections from X onto Y forms a vector space, and one can define a norm on this space via:

$$||P|| := \sup_{x \in X - \{0\}} \frac{||Px||_X}{||x||_X} \tag{1}$$

when the space of projections from X onto Y is nonempty, one is often interested in finding projections of minimal norm. This is because given $x \in X$, the projection $Px \in Y$ can be considered as an approximation to x in the subspace Y. In this sense a projection of minimal norm gives for each $x \in X$ a best approximation Px in the subspace Y. This is especially useful when one wants to approximate elements in infinite dimensional spaces using finite subspaces. Finding minimal projections is not well understood in general however, and in most cases specific results have only been obtained for certain subspaces (B. Chalmers, 1990), (Lewicki, 1997), (G. Lewicki, 2007). Nonetheless, when the ambient space X under consideration has finite dimension, it can be shown that determining minimal projections can be cast as a linear programming problem (Foucart, 2016). This equivalence was explored in order to generate a method for conducting experimental studies of minimal projections, and we wish to expand on this idea by investigating the viability of deep learning as a means of determining minimal projections. If deep neural networks are able to approximate minimal projections, then by studying how this is achieved we may gain some more insight into this problem. Conversely if deep neural networks are unable to attain good approximations of minimal projections, then this can serve as an interesting example of the failure of deep learning, and by studying what aspect of the problem makes it unsuitable to deep neural networks, we may discover some important limitations.

2. Relevant Background

There are three primary pieces of relevant background for this work. First is the paper written in 1974 by Blatter and Cheney which gave explicit formulas for hyper-planes in the sequence spaces ℓ_2 and c_0 (Blatter & Cheney, 1974). In the context of this work, their theorem on c_0 will be applied to $\mathbb{R}^n$ equipped with the infinity norm $||x||_\infty$. In this way, we have that given $(f_1, f_2, ..., f_n) \in \mathbb{R}^n$ such

that $||f||_1 = 1$, the minimal projection constant λ of the projections onto the hyperplane orthogonal to f relative to the infinity norm is given by

$$\lambda(f^{\perp}) = \begin{cases} 1 & ||f||_{\infty} \geq 1/2 \\ 1 + \dfrac{1}{\sum_{i=1}^{n} \dfrac{|f_i|}{1-2|f_i|}} & ||f||_{\infty} < 1/2 \end{cases}. \tag{2}$$

After this work, there is the paper by Lewicki in 2000 which extended these results to partial formulas in the case when the subspace has codimension 2. These formulas reveal how difficult and non-trivial finding minimal projection constants can be, and we refer the interested reader to (Lewicki, 1997) for why mathematicians turned to more computational approaches for calculating these values.

This now leads us to the work by Foucart in 2016 which recast the entire system into a relatively simple and readily computable linear program which can be computed for any values $n, m \in \mathbb{N}$ where n is the dimension of the vector space $\mathbb{R}^n$ and m is the dimension of the subspace of $\mathbb{R}^n$ onto which the projection is considered. Utilizing the MATLAB convex optimization solver CVX, Foucart created a MATLAB package MinProj (code publicly available on GitHub) which directly calculates the minimal projection constants using both the ℓ_1 and ℓ_{∞} norms. (Foucart, 2016) In this paper, the package served as the means by which all data for training and validation was gathered.

3. Theoretical Guarantees

With these results in mind, it is now appropriate to discuss their practical implications for finding minimal projection constants. First, we consider the formula given by (2). Observe that the only possible points of discontinuity are when $||f||_{\infty} = 1/2$, but in this case, we observe that if $\alpha_i = \dfrac{|f_i|}{1-2|f_i|} = \infty$ for any f_i, then we must have that the summation will be zeroed out, giving a value of 1 and preserving continuity. Similarly, the formula also has a degree of smoothness outside of the case when $||f||_{\infty} = 1/2$. With this in mind, we can apply the classic work of Cybenko in (Cybenko, 1989) which states that there exists a neural network which can approximate this function to an arbitrary degree, however, we do not know whether or not this neural network can actually be found through standard neural network training methods. This is one of the core ideas that is investigated in this paper.

From here, we might ask whether or not we have any existence theorems for the other cases where we do not have a simple formula. Does a neural network exist that can approximate those values as well? Fortunately, the answer is a resounding yes, and it is largely due to the Foucart paper (Foucart, 2016) and a paper on linear programming by Martin (Martin, 1975).

In light of Cybenko's Universal Approximation Theorem, if it can be shown that the minimal projection constant varies continuously as the subspace upon which we are projecting varies, then the existence of a neural network which approximates the constants well is guaranteed. In the case of a finite dimensional space X, Foucart showed in (Foucart, 2016) that the projection constant minimization problem on $X = \ell_{\infty}^n$ is equivalent to the linear optimization problem:

$$\begin{aligned} &\text{minimize}_{d,P}\ d \text{ subject to} \\ &(U^T \bigotimes I_n)vec(P) = vec(U), \\ &(\tilde{U}^T \bigotimes \tilde{U}^T)vec(P) = 0 \\ &\text{and to } \sum_{j=1}^{n} |P_{ij}| \leq d, \text{ for all } i \in [[1, n]] \end{aligned} \tag{3}$$

Where U is the matrix representation of the subspace upon which we are projecting, $\tilde{U}$ is the matrix representation of the orthogonal complement of the subspace represented by U, P is the matrix representation of the projection map under consideration, and I_n is the $n \times n$ identity matrix. Furthermore in (Martin, 1975) Martin asserts that

the optimum value of a parametric linear programming problem is continuous, provided both the primal and dual problems have a bounded optimal solution set. But since the problem under consideration is a simple linear optimization problem, strong duality holds, and the primal and its dual both have bounded optimal solution sets, thus (3), viewed as a programming problem parametrized by the coefficients of U and $\tilde{U}$, has a continuous maximum.

With this in mind, we can now turn to the task of finding this neural network and getting a proper approximation of the function that sends subspaces to minimal projection constants. As we show in our results, knowing the existence of the neural network does not at all mean it is a simple task to find.

4. Results

4.1. Architectures & Experimental Setup

For this problem both fully connected networks and fully connected networks with skip connections were considered. Each network under consideration was trained on 200,000 training samples, and tested on 30,000 validation samples with a batch size of 1024 for 100 epochs. Various parameters were tested using a brute force grid search, and all architectures were implemented using Keras version 2.2.4 (Chollet et al., 2015) using Tensorflow 1.11.0 (Abadi et al., 2015) as the backend. All code and relevant datasets for this project are available for download at https://github.com/RyanMalt/ProjectionConstants.

4.1.1. Grid Search Parameters

A grid search was conducted in order to find an optimal neural network architecture for this problem. Table 4.1.1 outlines the parameters of the grid search.

Parameter	Minimum	Step Size	Maximum
Depth	2	1	50*
Width	50	50	550
L. Rate	0.001	variable	.04

Table 1. Grid Search Parameters (* for depths greater than 3 hidden layers, a fixed hidden layer width of 100 is used)

Regularization was not used in the grid search as preliminary tests showed that regularization severely impacted network performance. None of the architectures trained in preliminary testing suffered from over fitting, but early stopping was employed to skip networks stuck in poor learning regimes and capitalize on hardware resources. A patience of 10 epochs without at least .001 improvement in loss was used for early stopping when skip connections were not used, or depth was sufficiently low (less than 6), and a patience of 20 epochs was used otherwise.

4.1.2. Width and Depth

The results of the grid search are outlined in Figure 1 for the case 2-dimensional subspaces of ℓ_∞^4. Upon closer examination of the results of the grid search, we found that having sufficiently high width is the key parameter in this problem. Although additional hidden layers improve performance, the gains are only marginal, and begin to diminish if the depth is too large.

4.1.3. Learning Rates

Regardless of architecture, the value of the learning rate had the same effect, lower learning rates lead to slow but consistent loss improvement, while higher learning rates lead to more sporadic loss improvements where loss may not improve for several epochs before a significant improvement. Due to this sporadic loss improvement, some potentially high performing networks trained in the parameter search my have been terminated by early stopping, whereas architectures with a lower learning rate were much more likely to train through all 100 epochs.

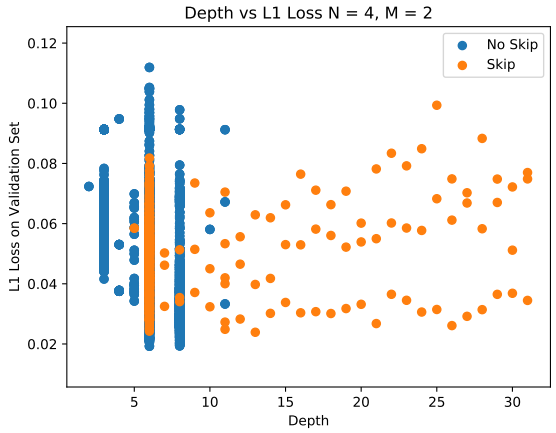


Figure 1. Best ℓ_1 losses achieved by networks of different depths. Note that due to early stopping several of the deeper networks without skip connections were terminated at high loss values, and thus are not shown in this plot.

4.2. Visualizations & Result Analysis

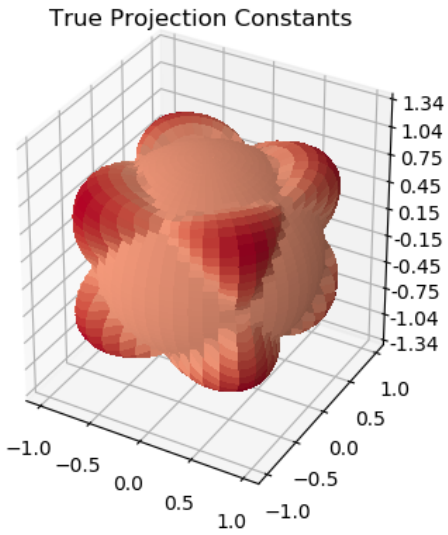
A key result obtained from our grid search was that the best performing network with respect to ℓ_1 loss was one without skip connections. In order to visualize how these networks are processing data several surface plots were made of the minimal projection constant as the subspace is varied over a 2-dimensional sphere (See Figure 2). The results show that the network without skip connections appears to assign every subspace the same value, so that on average it performs well, whereas the network with skip connections assigns different values for different subspaces, and thus appears to be learning some kind of structure to the problem.

In order to investigate why the network without skip connections seems to be simply averaging over the data, we generated several gray scale images from the weight matrices of different networks where the intensity of a given pixel is determined by the size of the associated weight (See Figure 3). We then zeroed out all pixels with an intensity less than 1 standard deviation above the mean intensity.

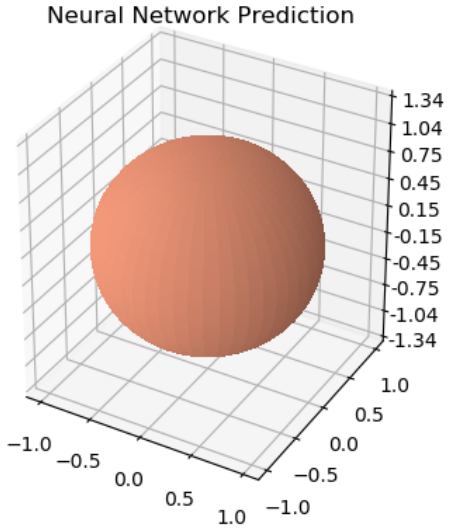
We found that a network trained on randomly labelled data produced weight matrices which have a strong striping pattern, whereas networks trained on more structured data and for which we can be more certain that structure is being learned, produce images that look more like random noise. In fact the networks without skip connections produce images which exhibit this striping pattern strongly, while the networks with skip connections produce images which look much more like random noise. Additionally, the networks with skip connections show a stronger density in the region corresponding to connections to the original inputs, indicating that memory of the original input may be important for learning relevant features for this problem. It was also observed throughout training that the networks with skip connections converged much more slowly, but because they are learning more interesting features, it may be worthwhile to increase the number of training epochs.

4.2.1. Skip Connections

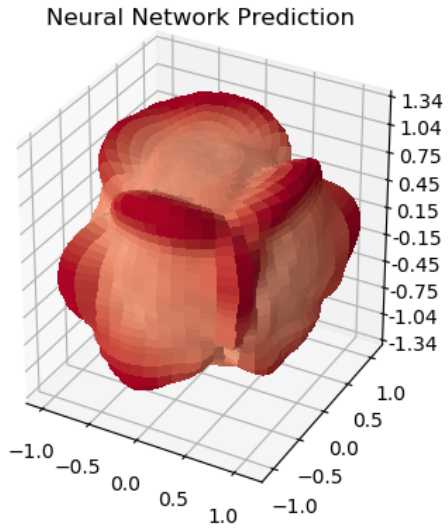
It was observed through our experiments that on average the best performing neural network was one without skip connections, yet this network seems to assign every subspace the same value. The best performing network with skip connections, in contrast, is assigning different values to different subspaces, albeit incorrect values. For this reason, despite their worse performance on average, we consider the networks with skip connections to be superior for this problem, and we believe they hold much more potential for accurately predicting projection constants. It is not entirely clear why the networks with skip connections exhibit more interesting behavior as there may be many contributing factors. In (Yarotsky, 2016) Yarotsky shows that with sufficient depth such networks are superior for the efficient approximation of functions in particular Sobolev spaces. This is due to the fact that the compositional structure of these networks allow for efficient approximation of e.g. the squaring function and hence multiplication by using the polarization identity, and from there more complex functions can be built. In the special cases for which we



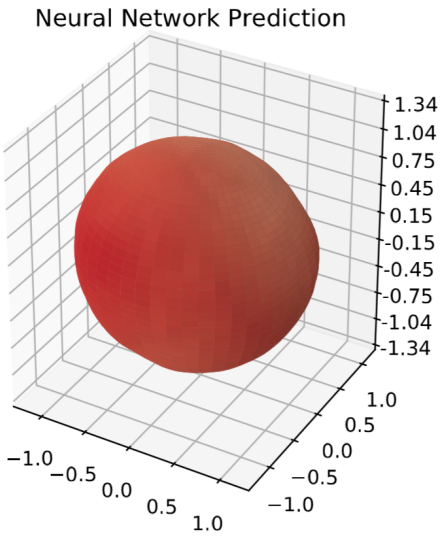
(a) True projection constants



(b) Neural Network without skip connections

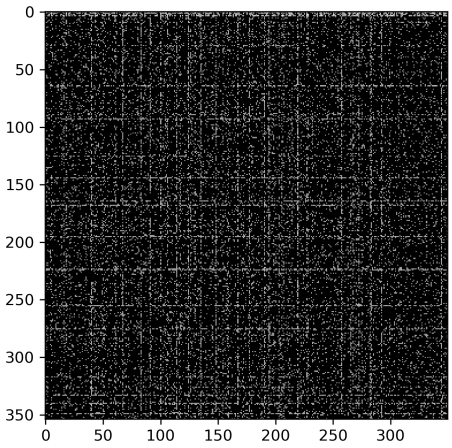


(c) Neural Network with skip connections

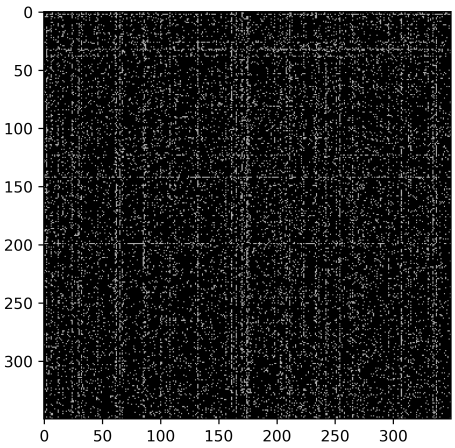


(d) Neural Network trained with scrambled data

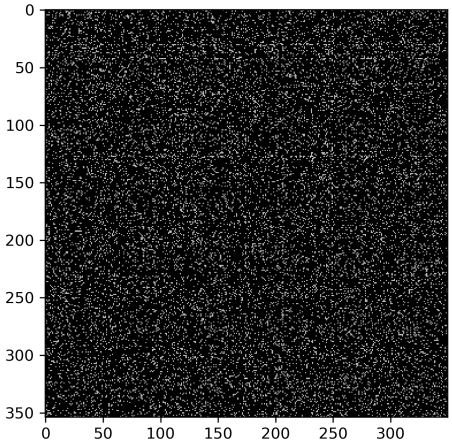
Figure 2. Predicted projection constants as subspace is varied over a 2-sphere. The distance of a point on the surface from the origin is the value of the projection constant for the associated subspace.



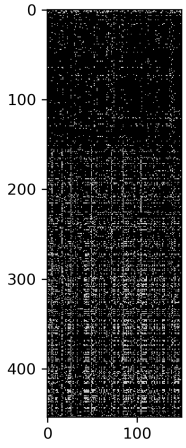
(a) Network with random weights first weight matrix



(b) Trained network without skip connections first weight matrix



(c) Trained network with skip connections first weight matrix



(d) Trained network with skip connections last weight matrix

Figure 3. gray-scale plots of weight matrices.

have formulas for the projection constants, such as the one previously discussed derived from the results in (Blatter & Cheney, 1974), the projection constants are at least locally smooth, and as previously discussed they are always continuous as a function of the subspace.

5. Conclusions and Future Work

5.1. Architecture

With more time or more powerful computing resources several deep networks with skip connections may be trained with several more epochs, which may result in a significant improvement performance. Instead of a brute force grid search, a more sophisticated search method, such as a Bayesian parameter search may be implemented in the future to capitalize on resources.

5.2. Linear Programming and Neural Networks

One of the more interesting directions for future work on this problem involves broadening the scope of the linear programs considered. Due to the work in (Martin, 1975), we know there exists a class of continuous functions whose values are determined entirely by solving a linear program at each particular point. While linear programming has come a long way in terms of efficiency, there are still bottlenecks in higher dimensions which have yet to be overcome. Since there is already knowledge that these functions are continuous and any amount of data necessary could be generated, it is conceivable that an active field of research applying neural networks to efficiently approximate these functions exists. Furthermore, the underlying structure of the problem (and perhaps a few additional assumptions) could grant a framework necessary to generate various theoretical results on matters such as optimal learning rates and architecture, the necessity of depth, and convergence rates with respect to quantity of data.

5.3. Skip Connections

A more thorough understanding of the expressive power and learning capacity of neural networks with skip connections is required to fully understand the differences observed in the performance of the different types of networks in this paper. Some specific results in this area are known (Yarotsky, 2016), but a more in depth investigation of these differences is beyond the scope of this work.

5.4. Visualizations

It may well prove helpful to investigate further ways of visualizing the information learned by neural networks, as the visualization of specific features will help to expose the limitations and strengths of the neural network's predictions.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `http://tensorflow.org/`. Software available from tensorflow.org.

B. Chalmers, F. M. Determination of a minimal projection from c[1, 1] onto the quadratics. *Numerical Functional Analysis and Optimization*, 11:1, 1990.

Blatter, J. and Cheney, E. W. Minimal projections on hyperplanes in sequence spaces. *Annali di Matematica Pura ed Applicata*, 101(1): 215–227, Dec 1974. ISSN 1618-1891. doi:

10.1007/BF02417105. URL https://doi.org/10.1007/BF02417105.

Chollet, F. et al. Keras. https://keras.io, 2015.

Cybenko, G. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, Dec 1989. ISSN 1435-568X. doi: 10.1007/BF02551274. URL https://doi.org/10.1007/BF02551274.

Foucart, S. Computation of minimal projections and extensions. *Numerical Functional Analysis and Optimization*, 37(2):159–185, 2016. doi: 10.1080/01630563.2015.1091014. URL https://doi.org/10.1080/01630563.2015.1091014.

G. Lewicki, L. S. Chalmersmetcalf operator and uniqueness of minimal projections. *Journal of Approximation Theory*, 148:71, 2007.

Lewicki, G. Minimal projections onto two dimensional subspaces of $\ell_\infty^{(4)}$. *Journal of Approximation Theory*, 88:92, 1997.

Martin, D. H. On the continuity of the maximum in parametric linear programming. *Journal of Optimization Theory and Applications*, 17(3):205–210, Nov 1975. ISSN 1573-2878. doi: 10.1007/BF00933875. URL https://doi.org/10.1007/BF00933875.

Yarotsky, D. Error bounds for approximations with deep relu networks. *CoRR*, abs/1610.01145, 2016. URL http://arxiv.org/abs/1610.01145.