MEASURING THE INTRINSIC DIMENSION OF OBJECTIVE LANDSCAPES

August 27, 2022

Arturo Maiani

Abstract

This projects deals with the study of the intrinsic dimension of objective landscapes through constraints applied in the space of network parameters.
https://github.com/ArturoMaiani/DLAI-project/tree/main

1. Introduction

In this project we expand on the work of (Li et al., 2018) in which they trained DNNs whose parameters were constrained in a randomly oriented linear subspace. In this work we follow a research direction suggested by (Li et al., 2018) which consisted in exploring the use of nonlinear constraints instead of linear subspaces. In addition, a particular form for the matrix P was investigated.

2. Related work

The pioneeristic work from (Li et al., 2018) has started this research direction, by training on fixed randomly oriented subspaces. (Gressmann et al., 2020) changed the subspace orientation at each SGD step, obtaining higher classification accuracy but losing the global structure of the manifold on which the weights are traveling.

3. Method

A DNN is a function of some weights $w \in \mathcal{R}^N$ but in this case we want to make sure that $w = f(\theta)$ with $\theta \in \mathcal{R}^d$ and $d << N$. In (Li et al., 2018) a linear mapping of the form $w = \theta_0 + P\theta$ has been used, where P is a $N \times d$ matrix and θ_0 is an offset. In this work an hyperellipsoid has been tested:

$$\mathbf{w} = \theta_0 + \big[P_1 \mid P_2 \mid ... \mid P_d\big]\theta$$

$$\mathbf{w} = \theta_0 + \\ + \big(P_1, P_2, ..., P_{d+1}\big) \begin{pmatrix} r_1 cos(\theta_1) \\ ... \\ r_d sin(\theta_1) sin(\theta_2) ... cos(\theta_d) \\ r_{d+1} sin(\theta_1) sin(\theta_2) ... sin(\theta_d) \end{pmatrix}$$

As it can be seen with an hyperellipsoid we get a free extra column for the matrix P with the same number of free parameters. This may be insignificant since $d >> 1$, but still is an interesting fact.

3.1. First Structure used for matrix P

The structure used by (Li et al., 2018) was an orthonormal random matrix. In this project a matrix whose entries are extracted from gaussian distributions of zero mean and standard deviation around 0.1 (Hyperplane) and 0.08 (Hyperellipsoid) have been observed to be sufficient for the learning task.

3.2. Proposal of a second structure for matrix P

The following reasoning was taken from the Dynamic Linear Dimensionality Reduction explained at page 4 of (Li et al., 2021). Such way to choose P is based on analyzing the initial steps of the optimization process of the unconstrained network, as follows.
1) Collect T optimization steps on the unconstrained network, collect the weight values as columns in a matrix C.
2) By decomposing C with SVD we can find the main directions for such trajectory.
3) Then we can build matrix P as follows:

$$C = U\Sigma V^T \quad P = \big[\gamma \times U[0:k] \mid v_{k+1}, \ldots, v_d\big]$$

Where k can be chosen as how many columns of U we want, $v_{k+1}, \ldots, v_d$ are chosen randomly as section 3.1 and γ is a scaling coefficient equal to the average norm of v_j so to have all equal norm vectors in matrix P.
This structure is analyzed to see if having some vectors pointing in the "natural learning direction" does improve the outcome of the learning task.

Email: Arturo Maiani <maiani.1738271@studenti.uniroma1.it>.

Deep Learning and Applied AI 2022, Sapienza University of Rome, 2nd semester a.y. 2021/2022.

4. Experimental results

The results presented in this section are based on the classification task of the MNIST dataset. The dataset is composed of 60'000 images for training and 10'000 images for testing. The DNN network is a MLP composed of 2 hidden layers of 200 and 100 units. The input layer has 784 neurons and the output layer has 10, since we want to classify numbers from 0 to 9.
First of all a standard model is trained: the accuracy is 98 % and has been obtained by training for 5 epochs, using the Adam optimizer, a linear scheduler for the learning rate from 10^{-2} to 10^{-4} and using dropout (20 %) after the first layer. In addition in order to avoid the "dying ReLU" problem the activation function "LeakyReLU" has been used. The training time was very fast: 13 seconds.

4.1. Linear subspace

The first constraint taken into exam is that of the linear subspace, proposed by (Li et al., 2018). The way in which the intrinsic dimension is determined is through a threshold of 90 % w.r.t. the performance of the unconstrained network, which consists of a threshold accuracy of 88%.
In addition simulations have shown that the offset vector θ_0 was not necessary. The constrained networks have been trained for two epochs with the Adam optimizer with a fixed learning rate of $5e^{-4}$.
Since matrix P is generated randomly, we can see that we got lucky for $d = 725$, obtaining a staggering 96% accuracy. In general we can conclude that for the MNIST dataset the Intrinsic Dimension falls between 700 and 750, confirming the results of (Li et al., 2018). In addition, the training time for a single epoch is around 2 minutes since we need to perform a big matrix vector multiplication at each forward step.

4.2. Hyperellipsoid

Such manifold is not as simple as a linear subspace, this fact makes training quite longer. In addition the radius for each direction is obtained from a uniform distribution between $[0.075, 0.3]$. The Adam optimizer with learning rate equal to $6e^{-3}$ has proven to give some results in this case. Such models have proven to be much harder to train since 1) the time for training on a single epoch becomes 3 minutes and 2) the training required more epochs. In addition the model is very sensitive to random choice of parameters, meaning that in many cases even with $d = 800$ the test accuracy was very low. After some trials it can be concluded that with such model the intrinsic dimension was $d = 750$, of around 50 units grater w.r.t. the linear constraint, hence performing worse.

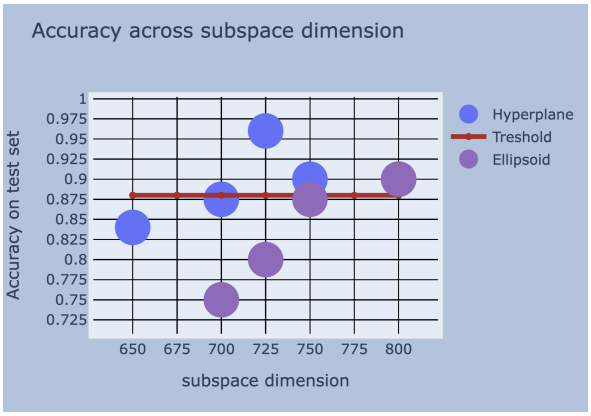

Figure 1. Test Accuracy vs Subspace Dimension

4.3. SVD hyperplane vs Vanilla hyperplane comparison

Such comparison has been done on a (30,30) MLP (20 K parameters) instead of (200,100) (170 K parameters) since SVD computations in the second case became unfeasible. The test has been done using d=750, for each method a statistical sample of 20 trainings has been performed. The scaling coefficient $\gamma = ||\mathcal{N}(0, 0.1, (20K, 1))||$ was ≈ 300 being the norm of a 20K dimensional vector whose entries were zero mean gaussians of 0.1 standard deviation. As it can be seen the presence of deterministic vectors decrease the variance of the learning but the mean accuracy has not improved, once again confirming that Vanilla is better w.r.t. my strange ideas.

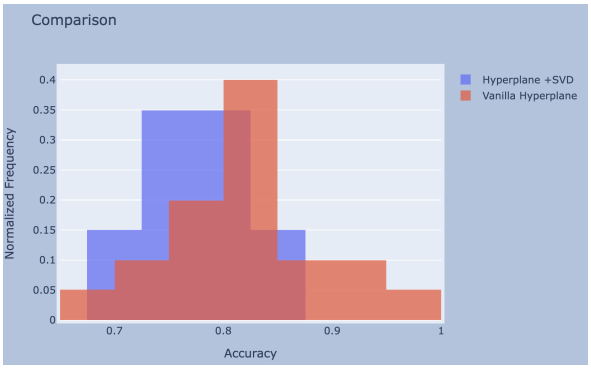

Figure 2. SVD+hyperplane vs Vanilla Hyperplane accuracy distribution

5. Conclusions

In this work a hyperplane constraint and a hyperellipsoid constraint have been examined in the context of the investigation of intrinsic dimension of a deep learning task. In addition to that a variation in the structure of matrix P has been tested. In both cases the Vanilla Hyperplane has performed better.

References

Gressmann, F., Eaton-Rosen, Z., and Luschi, C. Improving neural network training in low dimensional random bases. *CoRR*, abs/2011.04720, 2020. URL https://arxiv.org/abs/2011.04720.

Li, C., Farkhoor, H., Liu, R., and Yosinski, J. Measuring the intrinsic dimension of objective landscapes. *CoRR*, abs/1804.08838, 2018. URL http://arxiv.org/abs/1804.08838.

Li, T., Tan, L., Tao, Q., Liu, Y., and Huang, X. Low dimensional landscape hypothesis is true: Dnns can be trained in tiny subspaces, 2021. URL https://arxiv.org/abs/2103.11154.