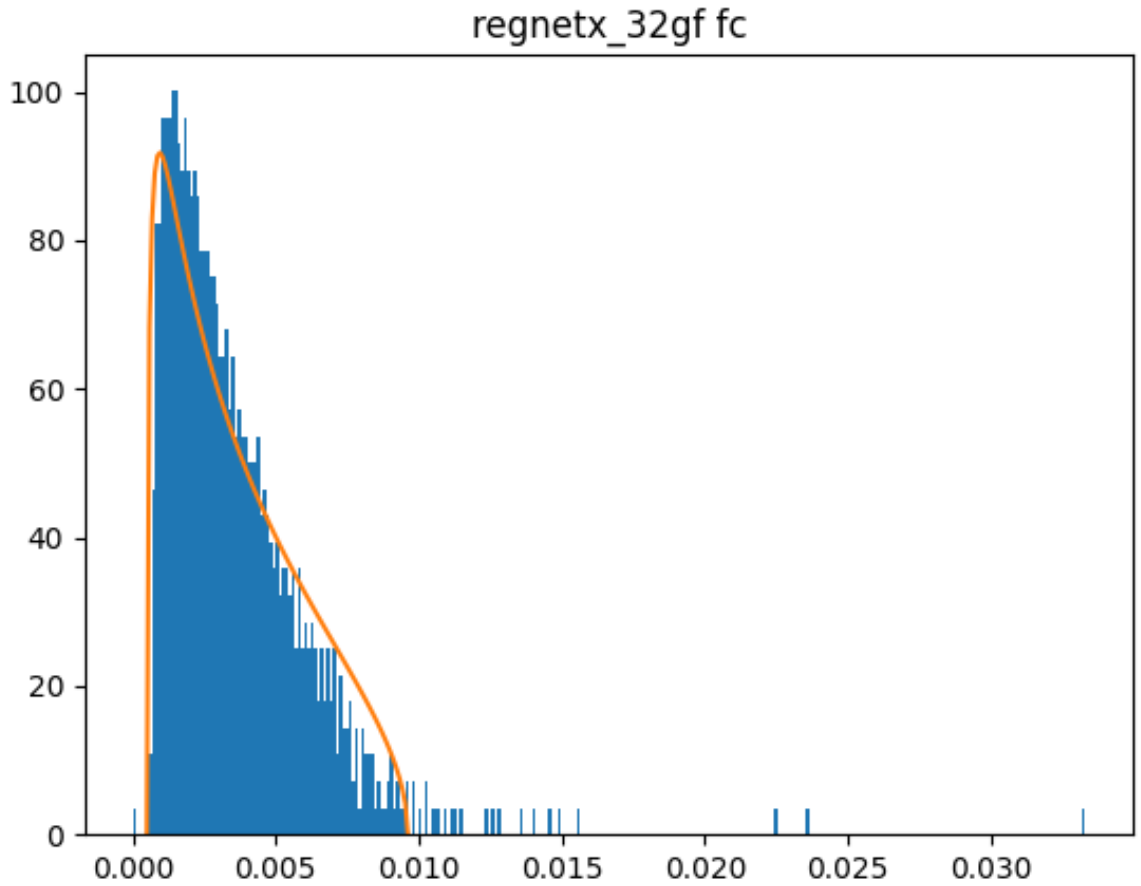
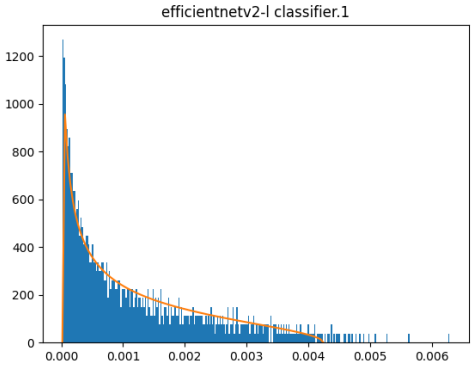
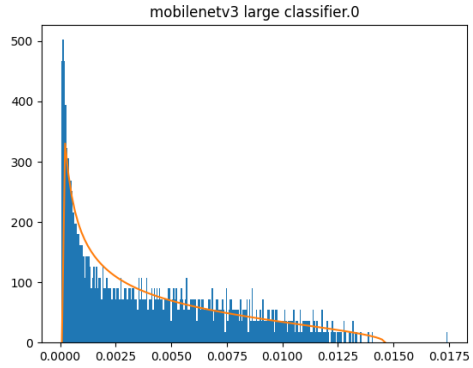
Eigenvector Self Regularization of Neural Networks

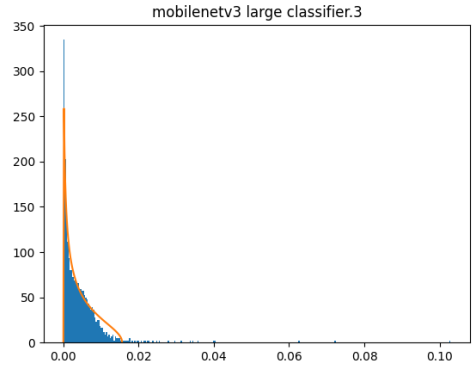
An Example

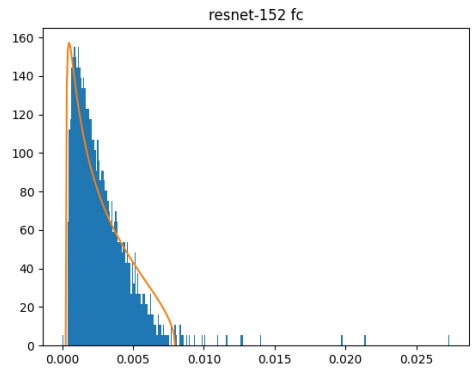

regnetx_32gf fc
100
80
60
40
20
0
0.000
0.005
0.010
0.015
0.020
0.025
0.030

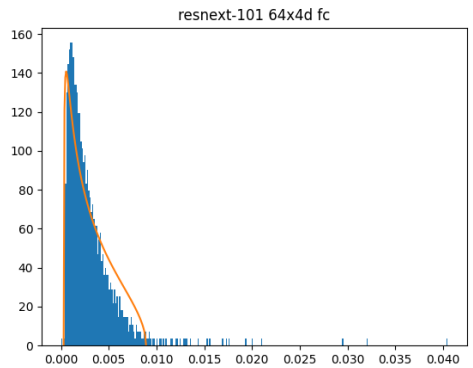
Spectrum of Linear Layers of Neural Networks

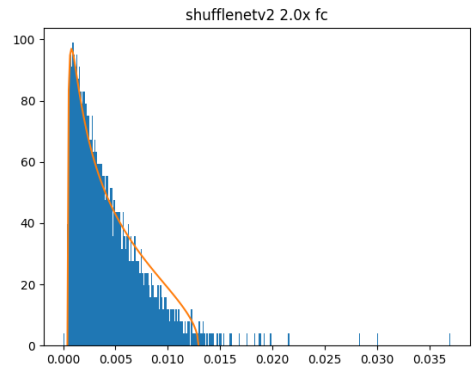












Tweaking the Weights

We first perform SVD on the $p \times n$ weight matrix W:

$$W = U\Sigma V^T = U_{p\times q}\Sigma_{q\times q}V^T_{n\times q} + Z,$$

with the singular values in Σ sorted from greatest to smallest and Z is the residual matrix. So

$$\frac{1}{n}WW^T = \frac{1}{n}U\Sigma^2U^T = HH^T + N$$

where

$$H = \frac{1}{\sqrt{n}}U_{p\times q}\Sigma_{q\times q}.$$

Here $U_{p\times q}$ consists of the top q left singular vectors and $\Sigma_{q\times q}$ consists of the top q singular values.

Tweaking the Weights

Provided that Z is noise, and that $\mathbb{E}W = B\Lambda\mathcal{V}^T$, then the left singular vectors of H_{JS} which is the JS correction of H will be a better estimator of B then $U_{p\times q}$.

Tweaking the Weights

To correct the weights W, we apply JS correction to the space spanned by the columns of H. Here we are not scaling any directions (yet). Given priors $v_1, \ldots, v_l$ (linearly independent) p-vectors, take A to be the matrix with the prior vectors as columns.
The shrinkage target is given by

$$M = A(A^T A)^{-1} A^T H$$

which is the projection of the columns of H to the space spanned by the priors.

Tweaking the Weights

The variance of the noise is estimated by

$$\nu^2 = \frac{\mathrm{Tr}(N)}{n_+ - q} \qquad \left(\text{or} \quad \frac{Tr(N)}{n_+ - (1 + \frac{n_+}{p})q} \right)$$

where n_+ is the number of nonzero singular values. Our shrinkage parameter is given by

$$C = I - \nu^2 J^{-1}, \;\; J = (H - M)^T (H - M).$$

Our JS correction of H is given by

$$H_{JS} = HC + M(I - C).$$

Tweaking the Weights

We take S^2 to be the largest q sample eigenvalues, i.e.

$$S^2 = H^T H = \frac{1}{n}\Sigma^2_{q\times q}.$$

We correct the singular values by taking

$$\Phi = S^2\Psi^2, \;\; \Psi^2 = I - \nu^2 S^{-2}$$

where S^{-2} is the inverse of S^2. So the q corrected singular values are given by

$$\Sigma_{JS,q\times q} = (n\Phi)^{1/2}.$$

Many other eigenvalue shrinkages are possible.

Tweaking the Weights

To correct the weights W, we again apply SVD to H_{JS} to get the left singular vectors β_{JS}.
We replace the top q left singular vectors of U in the SVD of W by β_{JS} to get a new matrix U_{JS}.
The JS corrected weights W_{JS} is given by

$$W_{JS} = U_{JS}\Sigma V^T$$

without singular value corrections. With singular value corrections, we have

$$W_{JS} = U_{JS}\Sigma_{JS} V^T.$$

JS corrections

Taking $A = (e)$ which corresponds to taking the grand mean shrinkage, without singular value corrections, we have

q	0	noise	1	2	3	4	5	6	7	8	9	10	11
Top1	72.04	0.13	72.05	72.14	72.08	72.07	71.98	72.05	71.94	71.65	71.95	71.81	71.77
Top5	90.21	0.53	90.22	90.31	90.21	90.20	90.10	90.16	90.04	89.72	89.92	89.94	89.67

q	12	13	14	15	16	17	18	19	20	21	22	23	24
Top1	71.76	71.67	71.62	71.91	71.90	71.80	71.50	71.82	71.63	71.59	71.84	71.50	71.47
Top5	89.29	89.65	89.42	89.75	89.68	89.41	89.00	89.84	89.08	89.11	89.45	89.11	88.37

With singular value corrections, we have

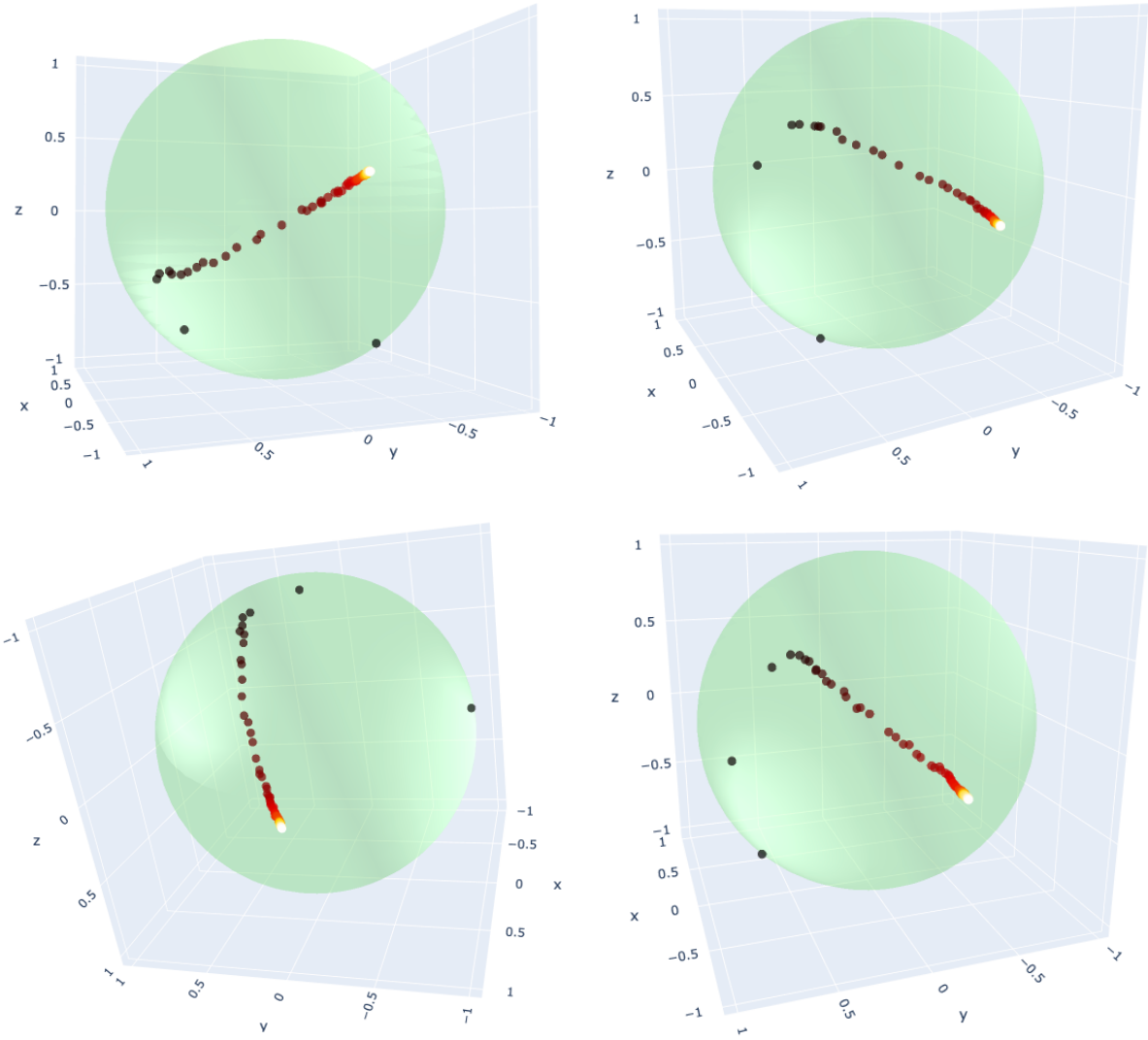
q	0	1	2	3	4	5	6	7	8
Top1	72.04	72.05	72.11	72.07	72.07	71.98	72.06	71.95	71.66
Top5	90.21	90.22	90.32	90.20	90.23	90.07	90.15	90.03	89.81

Leading eigenvector of input layer across epochs

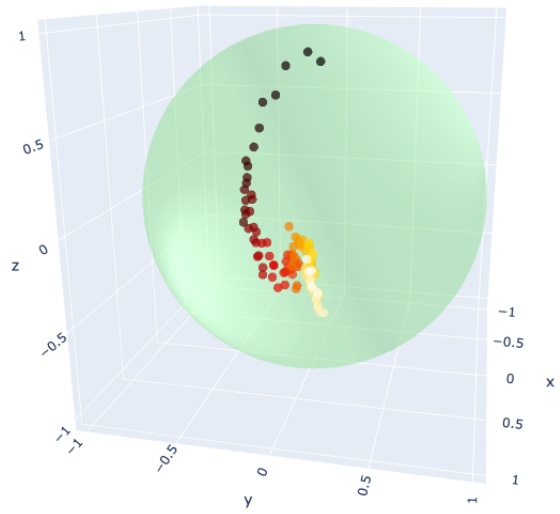
To have a better sense on how to make such corrections, we investigate how the eigenvectors evolve during training. We trained three layered perceptrons for 100 epochs on the CIFAR-10 dataset, with hyperparameters 3072,1024,256,10 and 3072,512,512,10.

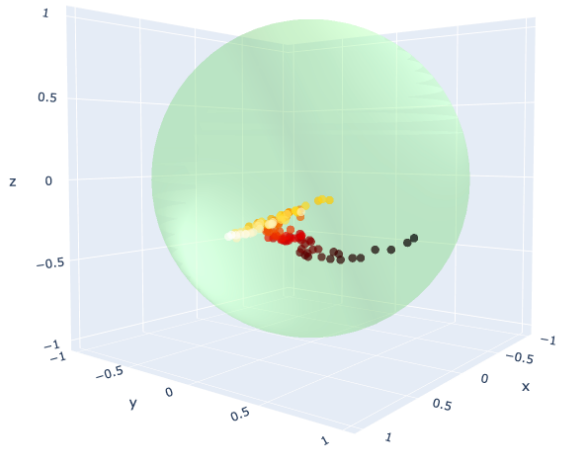
After each epoch, we collect the leading eigenvector which lies on a $p-1$ dimensional sphere. We apply dimension reduction by finding the best fitting 2-sphere.

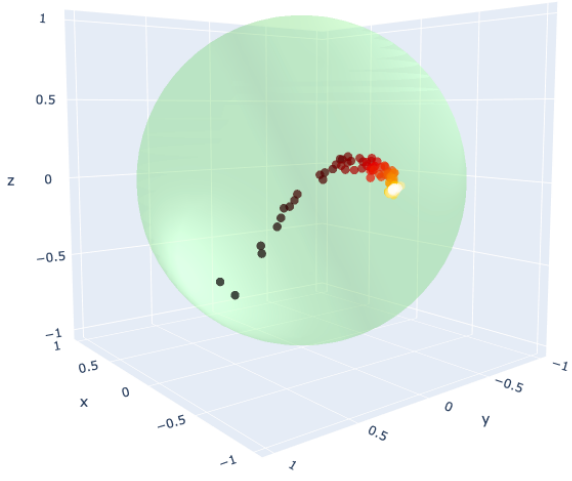
Leading eigenvector of input layer across epochs

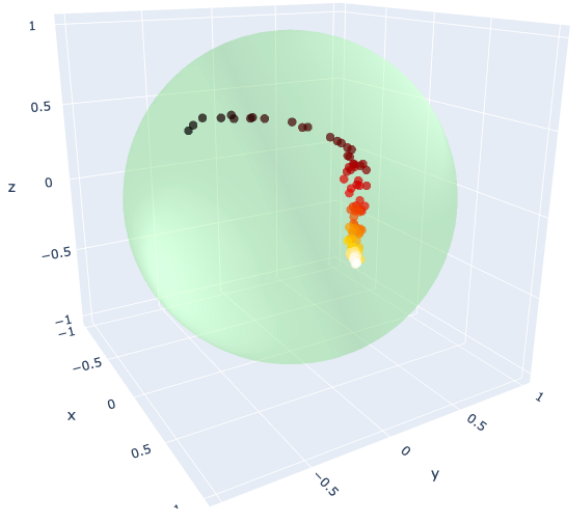


Leading eigenvector of hidden layer across epochs

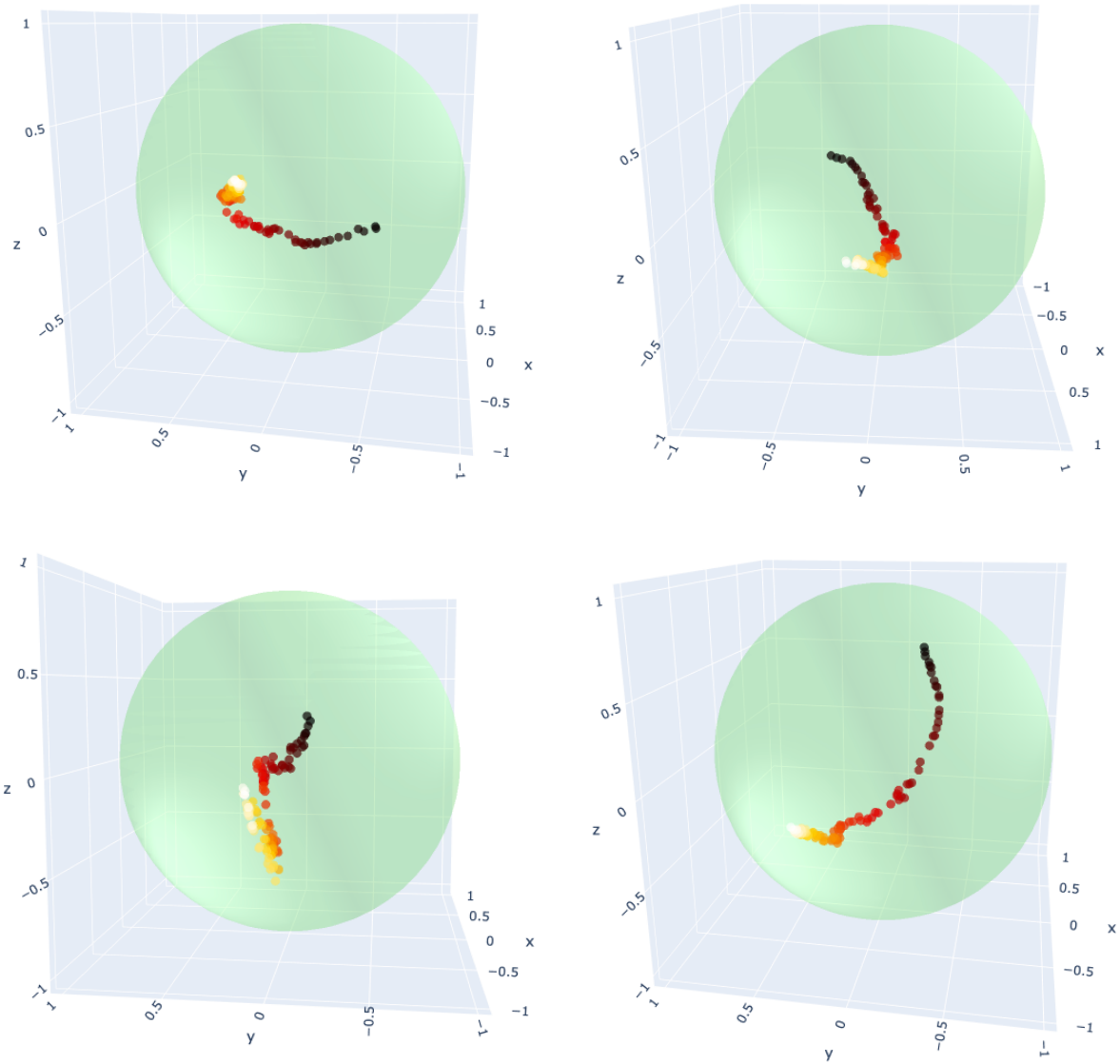








Leading eigenvector of output layer across epochs



Fitting scores

There are two natural notions of variances in such a fit and the sum of it would be called mixed variance following Huckemann–Ziezold, the first being the *RSS* given by the sum of the squared (Euclidean) distances between the original points and the projection onto the 2-sphere.
The other variance, variance on the sphere is similar to the Euclidean case, but we first have to define a notion of mean. A point $p \in S^l$ is called a Frechet mean of the points $x_1, \ldots, x_n \in S^l$ if it minimizes the function:

$$p \mapsto \sum_{i=1}^{n} d^2_{S^l}(x_i, p).$$

In most cases, the Frechet mean is unique.

Fitting scores

So now we can define the variance on the sphere given by the sum of the squared distances between the projected points and the Frechet mean μ. The mixed variance is then defined by:

$$Var = \sum_{i=1}^{k} d^2_{\mathbb{R}^p}(x_i, Proj_{S^2} x_i) + \sum_{i=1}^{k} d^2_{S^2}(Proj_{S^2} x_i, \mu).$$

Our fitting score is then defined as:

$$R^2 = 1 - \frac{RSS}{Var}.$$

Fitting scores

Here are the S^2 fitting scores:

Runs	Input layer	Hidden layer	Output layer
0	0.9318	0.8741	0.8964
1	0.9254	0.7272	0.8427
2	0.9234	0.8477	0.8672
3	0.9387	0.9223	0.9629

If we focus on the input layer, and restrict to epochs 3-100 in run 0-2 and epochs 4-100 in run 3, we get S^1 fitting scores 0.9502, 0.9169, 0.9128 and 0.9328 respectively.

Johnson–Lindenstrauss lemma

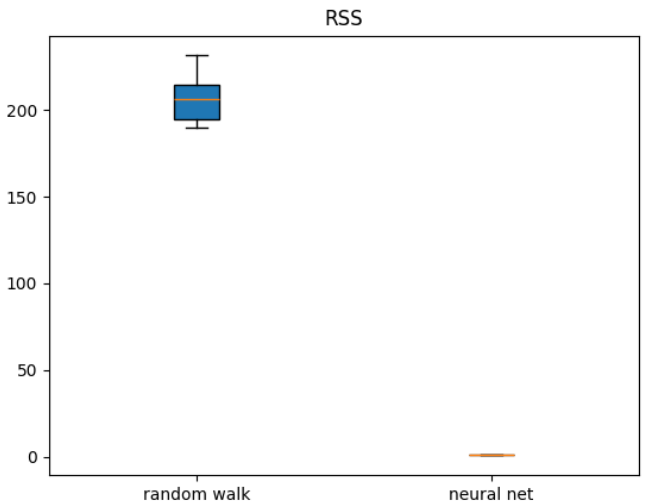
Given $0 < \epsilon < 1$, and N points $x_1, \ldots, x_N$ in $\mathbb{R}^n$, and an integer $k > 8 \ln N/\epsilon^2$, there exists a linear map $f : \mathbb{R}^n \to \mathbb{R}^k$ such that

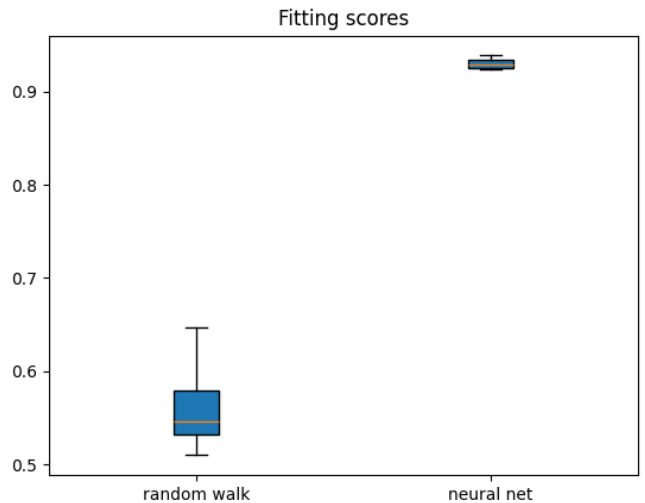
$$1 - \epsilon \leq \frac{||f(x_i) - f(x_j)||^2}{||x_i - x_j||^2} \leq 1 + \epsilon,$$

for $i \neq j$.
What this means is that we can find maps that preserves pairwise distances.

Crude benchmark

We performed 1000 random walks with 100 steps on a 3071 dimensional sphere and performed dimension reduction to the best fit 2-sphere. We obtained their respective RSS and fitting scores (inflated).





Bias vectors

We also performed PCA on the bias vectors of the neural networks. Here are the variances explained in the first component:

Runs	Input layer	Hidden layer	Output layer
1	0.9885	0.9851	0.9833
2	0.9881	0.9869	0.9883
3	0.9841	0.9823	0.9865

Future work

1. Run a lot more different examples with different architectures, hyperparameters, optimizers, seeds
2. Examine the behavior of convex optimization problems and see whether what we observe is just how optimization algorithms operate
3. Examine also what happens to the other spiked eigenvectors, our results seem to suggest that neural network training is low dimensional which the recent work by Mao et al. also suggests
4. How does this help finding better targets to perform JS corrections?