Searching for a more minimal intrinsic dimension of objective landscapes[1]

Jonathan Lam & Richard Lee

December 17, 2020

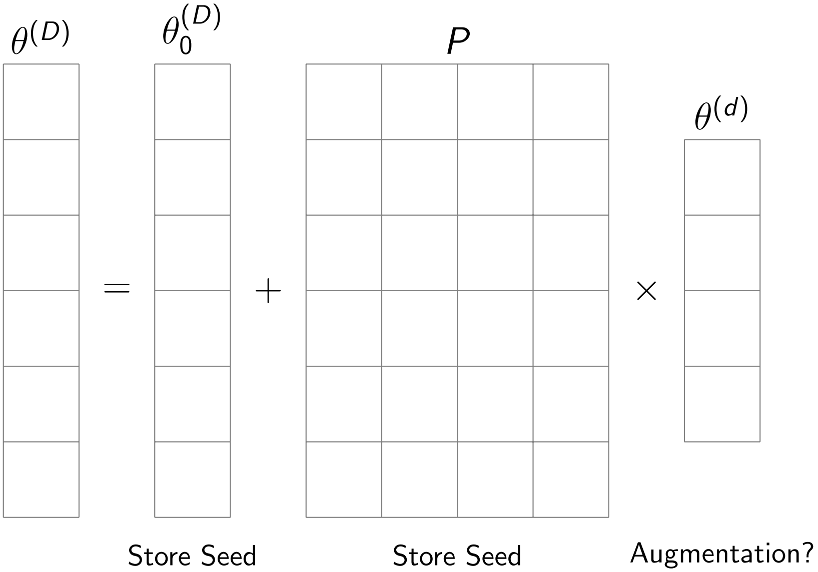
[1] https://github.com/jlam55555/intrinsic-dimension-projections

“Measuring the intrinsic dimension of objective landscapes”[2]

- Objective landscape (combination of learning problem + network architecture)
- Defines concept of “intrinsic weights”
- Proposed method of finding intrinsic weight of objective landscape by method of random linearly-projected weights
- Method to approximate minimum description length (MDL); can be used for model compression
- **Our goal**: to find a method to describe an objective landscape with even fewer weights

[2]Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. *Measuring the intrinsic dimension of objective landscapes*. In International Conference on Learning Representations, 2018.

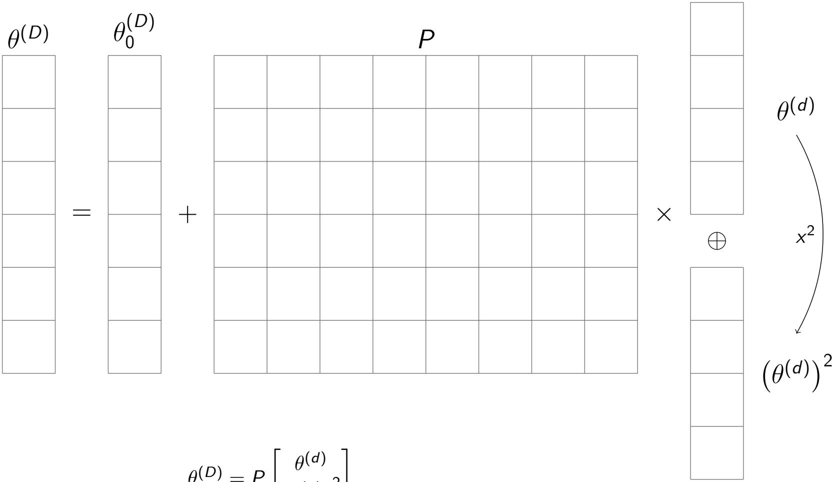
Method of random linearly-projected weights



Notation

- $\theta^{(D)}$: ordinary network weights; not stored as a `tf.Variable`, but rather the result of this calculation
- $\theta_0^{(D)}$: "base initialization weights" – like an initial bias; randomly initialized and non-trainable
- P : projection matrix; randomly initialized and non-trainable
- $\theta^{(d)}$: intrinsic weights; randomly initialized and trainable

Augmenting $\theta^{(d)}$ with squared terms



$$\theta^{(D)} = P \begin{bmatrix} \theta^{(d)} \\ \left(\theta^{(d)}\right)^2 \end{bmatrix}$$

What are random Fourier features (RFFs)?

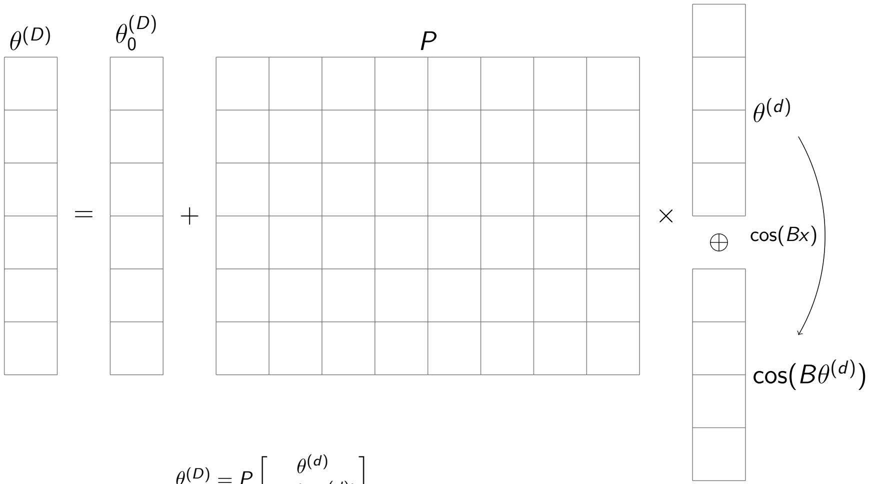
RFFs are a nonlinear many-to-many mapping that can be used to help capture different frequency components.

$$\gamma(\vec{v}) = \begin{bmatrix} a_1 \cos(2\pi \vec{b}_1^T \vec{v}) \\ a_1 \sin(2\pi \vec{b}_1^T \vec{v}) \\ a_2 \cos(2\pi \vec{b}_2^T \vec{v}) \\ a_2 \sin(2\pi \vec{b}_2^T \vec{v}) \\ \vdots \\ a_m \cos(2\pi \vec{b}_M^T \vec{v}) \\ a_m \sin(2\pi \vec{b}_M^T \vec{v}) \end{bmatrix}$$

We can append these to our intrinsic weights again:

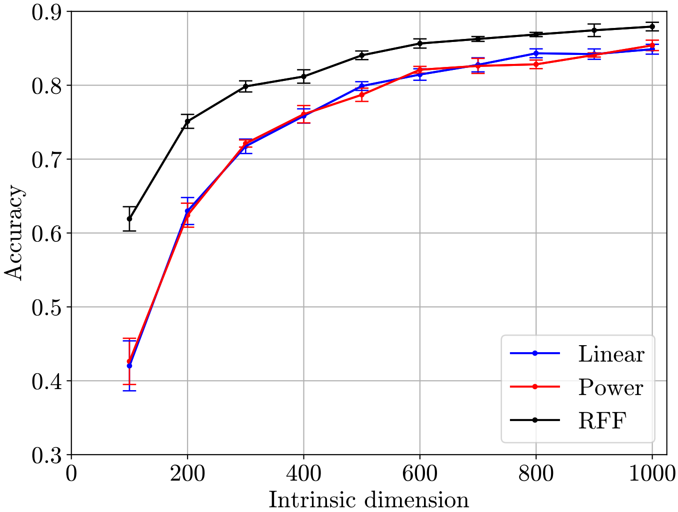
$$\theta^{(D)} = P \begin{bmatrix} \theta^{(d)} \\ \cos\left(B\theta^{(d)}\right) \\ \sin\left(B\theta^{(d)}\right) \end{bmatrix}$$

Example: Augmenting $\theta^{(d)}$ with RFF terms

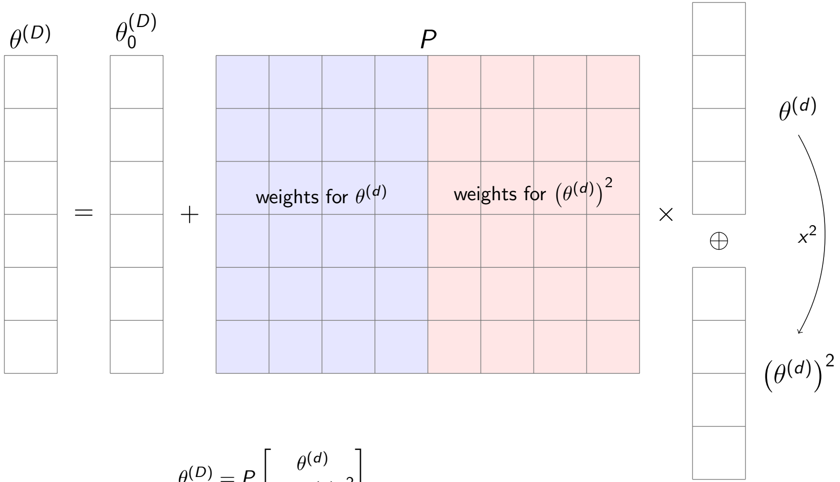


$$\theta^{(D)} = P \begin{bmatrix} \theta^{(d)} \\ \cos(B\theta^{(d)}) \end{bmatrix}$$

Augmenting $\theta^{(d)}$ with power, RFF terms

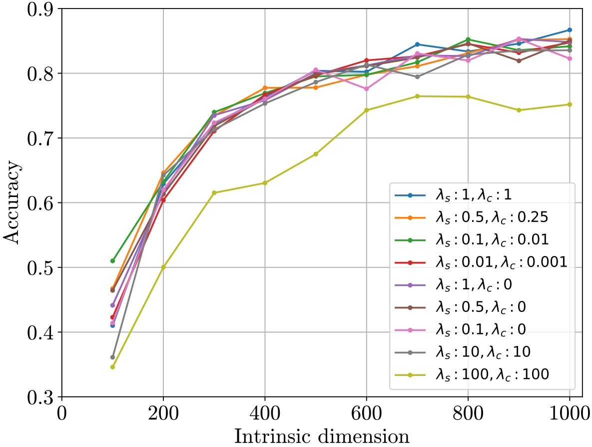


Varying initialization of P: motivation



$$\theta^{(D)} = P \begin{bmatrix} \theta^{(d)} \\ \lambda_s \left(\theta^{(d)}\right)^2 \end{bmatrix}$$

Varying the initialization of P

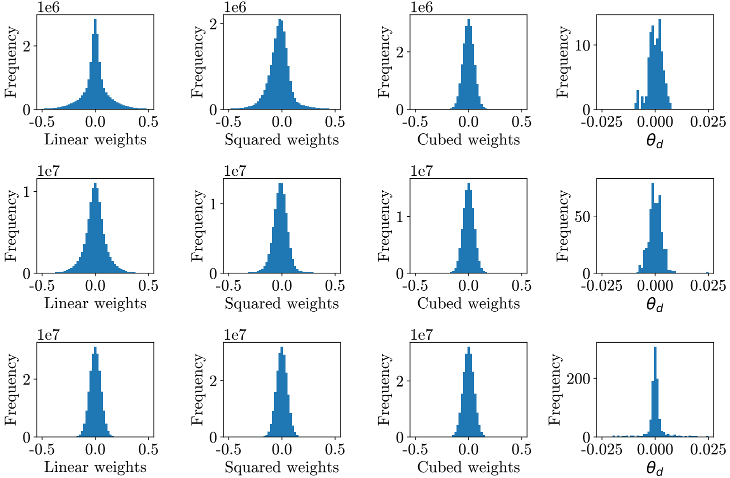


Trained P and $\theta^{(d)}$ weights

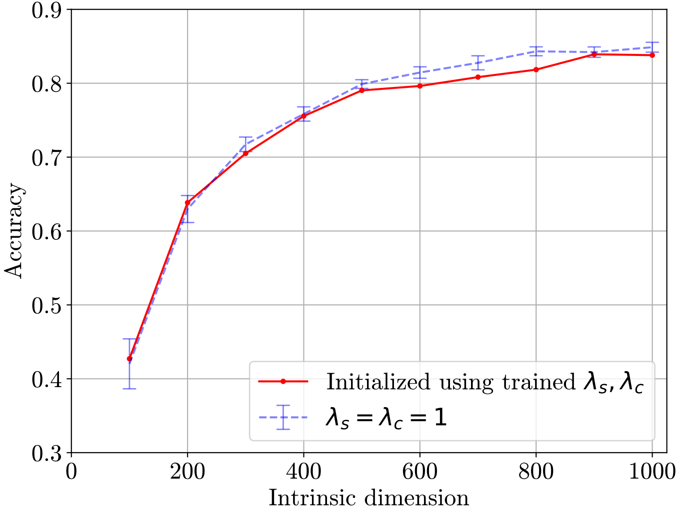
$d = 100$

$d = 500$

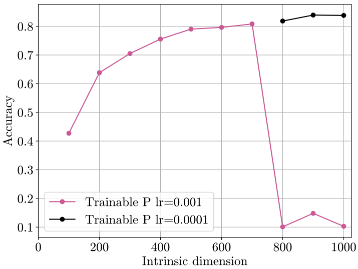
$d = 1000$



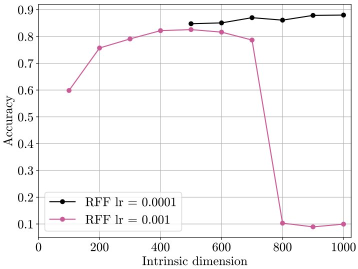
Initializing P with trained distributions



Bad convergence → decreased learning rates

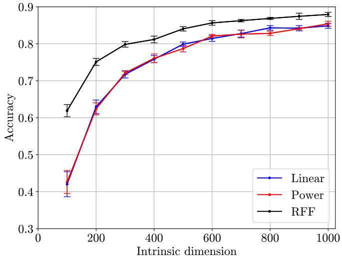


(a) Trainable Projection Matrix

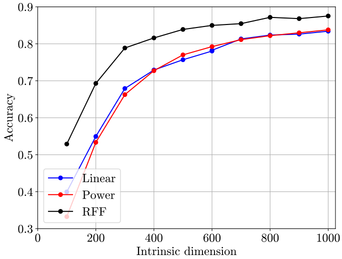


(b) Random Fourier Augmentation

Normalize P?



(c) Non-normalized Projection Matrix



(d) Normalized Projection Matrix

Conclusions and future research

- RFF $>$ linear $\approx$ power terms
- Still have a lot to try: different data, larger models, different layer types, etc.
- Compression? Practicality?