Gaussian Predictions from Gradient Descent Training of Wide Neural Networks

Jaehoon Lee*, Lechao Xiao*, Jascha Sohl-Dickstein, Jeffrey Pennington

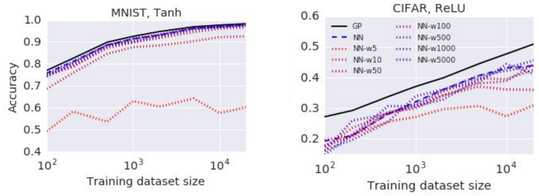
*Equal contribution, work done as part of Google AI Residency, g.co/airesidency

Summary

- Deep neural networks define complex systems for which many types of theoretical analysis are intractable. As in the physical sciences, investigating the extreme limits of such systems can often shed light on these hard problems. For neural networks, one such limit is of **infinite width**.
- There is a correspondence between multi-layer neural networks and Gaussian processes (GPs) (Neal, Lee & Bahri et al, Matthews et al):
  - Due to the Central Limit Theorem, an infinitely wide neural network with i.i.d. random parameters is a GP, with known mean and covariance function.
  - The correspondence enables ***Bayesian predictions*** to be made for deep neural networks, ***without ever instantiating a deep network***.
- We find in the infinite width limit gradient descent training dynamics for deep neural networks becomes **tractable**
  - NNGP kernel remains unchanged during training
  - Tangent kernel of the network is stable during training (Jacot et al)
  - Exact dynamics of *outputs* and *weights* assuming continuous time steepest descent
  - Residual randomness from initialization results in Gaussian distribution over test-set predictions during the full course of training
- **Assumptions**: 1) sufficiently wide networks 2) small learning rates

Motivation

Finding 1: As SGD trained neural networks become wider their accuracy approaches that of the equivalent Gaussian Process



(Lee & Bahri et al 2018)

Bayesian Inference VS gradient descent training

$$z^*|\mathcal{D}, x^* \sim \mathcal{N}(\bar{\mu}, \bar{K})$$
$$\bar{\mu} = K_{x^*,\mathcal{D}}(K_{\mathcal{D},\mathcal{D}} + \sigma_\epsilon^2 \mathbb{I}_n)^{-1}\boldsymbol{t}$$
$$\bar{K} = K_{x^*,x^*} - K_{x^*,\mathcal{D}}(K_{\mathcal{D},\mathcal{D}} + \sigma_\epsilon^2 \mathbb{I}_n)^{-1}K_{x^*,\mathcal{D}}^T$$

$$\Delta W = -\eta \frac{dL(f(\mathcal{D}; W)))}{dW}$$
$$f(x^*; W^*, \mathcal{D})$$

Core Result

Kernel regression to NNGP (Training only the readout layers)

- Random initialized networks with least square regression problem

$$L(\theta) = \frac{1}{2|\mathcal{D}|}||\Phi\theta - Y||^2$$

- For small enough learning rates, gradient descent becomes gradient flow equation

$$\frac{d\theta}{dt} = -\frac{\eta}{|\mathcal{D}|}\Phi^T(\Phi\,\theta - Y)$$

- Exact dynamics of readout weights are given by

$$\theta(t) = (I_n - P_\parallel)\theta_0 + (I_n - \exp(-\Phi^T\Phi\,\tfrac{\eta t}{|\mathcal{D}|}))\Phi^+ Y + \exp(-\Phi^T\Phi\,\tfrac{\eta t}{|\mathcal{D}|})P_\parallel\,\theta_0$$

- Test predictions are Gaussian distributed throughout training with mean and covariance

$$\mathbb{E}[f(x_T)](t) = K_{TD}\hat{K}_{DD}^{-1}\left(I - \exp(-K_{DD}\,\tfrac{\bar{\eta}t}{|\mathcal{D}|\sigma_\theta^2})\right)Y$$
$$Cov[f(x_T)](t) = K_{TT} - K_{TD}\left(I - \exp(-K_{DD}\,\tfrac{\bar{\eta}t}{|\mathcal{D}|\sigma_\theta^2})\right)\hat{K}_{DD}^{-1}K_{DT}$$

Tractable learning dynamics of wide neural networks

- As the network becomes wide, the change in each hidden neuron over training becomes infinitesimally small, and the NN is well approximated by its linearization:

$$\frac{df(\mathcal{D})}{dt} = -\frac{\eta}{|\mathcal{D}|}J_\mathcal{D}J_\mathcal{D}^T(f(\mathcal{D}) - Y) \qquad J_\mathcal{X} \equiv [\nabla_\theta f(x;\theta)]_{x\in\mathcal{X}}$$

- Solving this ODE gives the exact dynamics of the linear approximation of the NN:

$$f(\mathcal{D}) = Y + \exp(-J_\mathcal{D}J_\mathcal{D}^T\frac{\eta t}{|\mathcal{D}|})(f_0(\mathcal{D}) - Y)$$

- The dynamics of the prediction of a test point can be approximated as follows:

$$f(x) = f_0(x) - J_x J_\mathcal{D}^T(J_\mathcal{D}J_\mathcal{D}^T)^{-1}(I - e^{-J_\mathcal{D}J_\mathcal{D}^T\frac{\eta t}{|\mathcal{D}|}})(f_0(\mathcal{D}) - Y)$$

- The dynamics of the weights can be approximated similarly.

$$\omega \equiv W(t) - W(0) = J_\mathcal{D}^T(J_\mathcal{D}J_\mathcal{D}^T)^{-1}(e^{-J_\mathcal{D}J_\mathcal{D}^T\frac{\eta t}{|\mathcal{D}|}} - I)(f_0(\mathcal{D}) - Y)$$

- The predictions are Gaussian distributed throughout training whose mean and variance are tractable.

$$\Theta_{\mathcal{X}\mathcal{X}'} \equiv J_\mathcal{X}J_{\mathcal{X}'}^T$$
$$\mathbb{E}f(x) = \Theta_{x,\mathcal{D}}\Theta^{-1}(I - e^{-\frac{\eta}{|\mathcal{D}|}\Theta t})Y$$
$$\mathrm{Var}[f_\omega(x)] = K_{x,x} - 2\Theta_{x,\mathcal{D}}\Theta^{-1}(I - e^{-\eta\Theta t/|\mathcal{D}|})K_{x,\mathcal{D}}^T + \Theta_{x,\mathcal{D}}\Theta^{-1}(I - e^{-\eta\Theta t/|\mathcal{D}|})K_{\mathcal{D},\mathcal{D}}\Theta^{-1}(I - e^{-\eta\Theta t/|\mathcal{D}|})\Theta_{x,\mathcal{D}}^T$$

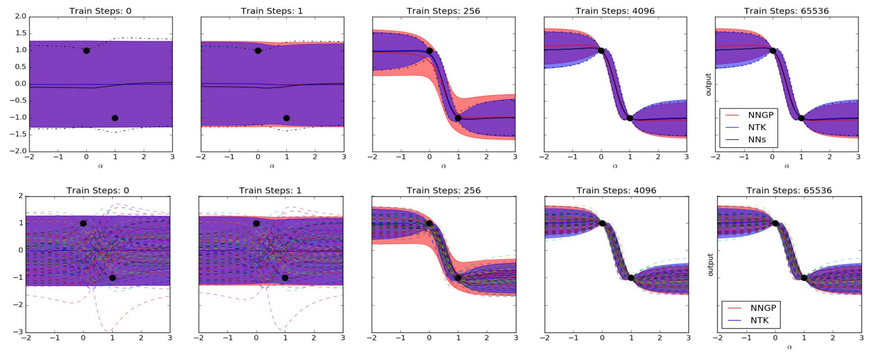
Reference

1. Neal, Radford M. Bayesian learning for neural networks. Ph.D. Thesis, University of Toronto, 1994.
2. Lee, J. and Bahri, Y., et al. "Deep neural networks as gaussian processes." *arXiv:1711.00165*, ICLR 2018.
3. Matthews, Alexander G. de G., et al. "Gaussian process behaviour in wide deep neural networks." *arXiv:1804.11271*, ICLR 2018.
4. Jacot, A. , et al. "Neural Tangent Kernel: Convergence and Generalization in Neural Networks." *arXiv:1806.07572* (2018).

Experiments

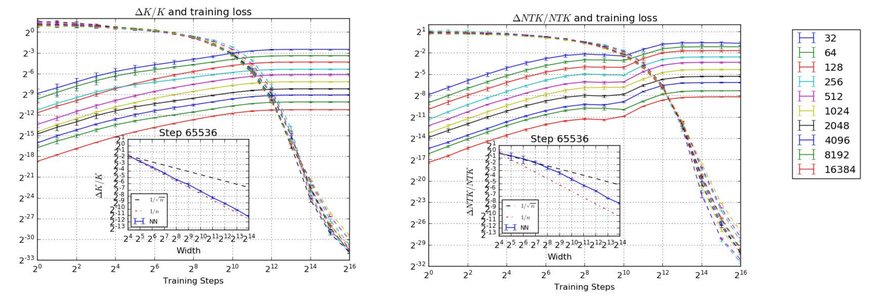
Posterior predictive distributions of ensembles of NN

(3 hidden layers, Tanh, CIFAR binary classification, width 8192, learning rate 0.5)



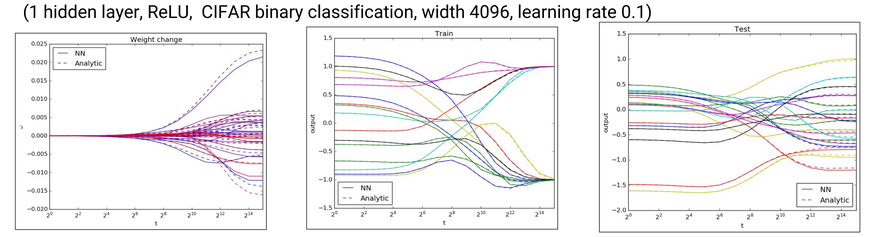
During training NNGP and tangent kernel change remains small

(1 hidden layer, Tanh/ReLU, CIFAR binary classification, learning rate=0.068)

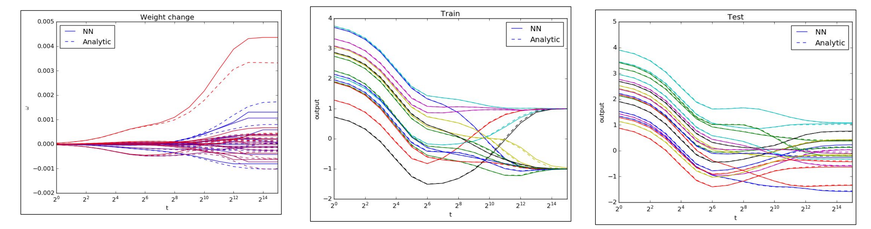


Exact dynamics of weights and outputs

(1 hidden layer, ReLU, CIFAR binary classification, width 4096, learning rate 0.1)



(3 hidden layers, ReLU, CIFAR binary classification, width 4096, learning rate 0.01)



Deep Neural Networks as Gaussian Processes

Jaehoon Lee*, Yasaman Bahri*, Roman Novak,
Samuel S. Schoenholz, Jeffrey Pennington, Jascha Sohl-Dickstein

*Equal contribution, work done as part of Google AI Residency, g.co/airesidency

ICLR 2018, https://arxiv.org/abs/1711.00165.
Code at: https//github.com/brain-research/nngp

Summary

- There is a correspondence between single-layer neural networks and Gaussian processes (GPs) (Radford Neal, 1994):
  - Due to the Central Limit Theorem, an infinitely wide neural network with i.i.d. random parameters is a GP, with known mean and covariance function.
- We build an analogous correspondence for multilayer networks, and find a ***GP that is equivalent to infinitely wide neural networks of any depth***:
  - We can compute the covariance function of this deep neural network GP (NNGP) iteratively for the desired depth. It is valid for generic nonlinearities.
- Bayesian inference for neural networks is typically intractable to implement exactly. The NNGP correspondence, however, enables ***Bayesian predictions (exact for regression problems!)*** to be made for deep neural networks, ***without ever instantiating a deep network***.

Implementation

Complexity of Kernel construction

- For some class of nonlinearities (e.g. ReLU) integration (*) can be done analytically.
- For generic nonlinearities, the Gaussian integration can be done numerically with n_g sampling points.
- Naive Kernel construction for n_{train} and n_{test} data points at depth L has compute cost

$$\mathcal{O}\left(n_g^2 L(n_{\text{train}}^2 + n_{\text{train}} n_{\text{test}})\right)$$

- Preprocess inputs to constant norm and generate precomputed grid of n_v variance and n_c correlations.

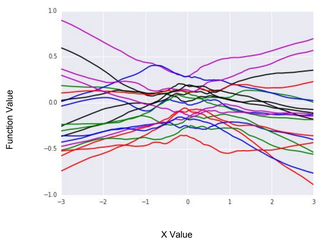
$$F_{ij} = \frac{\sum_{ab} \phi(u_a)\phi(u_b) \exp\left(-\frac{1}{2}\begin{bmatrix}u_a\\u_b\end{bmatrix}^T \begin{bmatrix}s_i & s_i c_j\\ s_i c_j & s_i\end{bmatrix}^{-1}\begin{bmatrix}u_a\\u_b\end{bmatrix}\right)}{\sum_{ab} \exp\left(-\frac{1}{2}\begin{bmatrix}u_a\\u_b\end{bmatrix}^T \begin{bmatrix}s_i & s_i c_j\\ s_i c_j & s_i\end{bmatrix}^{-1}\begin{bmatrix}u_a\\u_b\end{bmatrix}\right)}.$$

- Obtain Kernel elements by bi-linear interpolation over closest points in the grid.
- Compute cost is

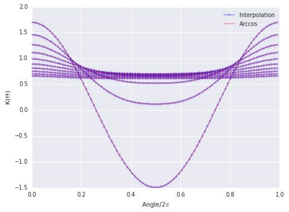
$$\mathcal{O}\left(n_g^2 n_v n_c + L(n_{\text{train}}^2 + n_{\text{train}} n_{\text{test}})\right)$$

- Can be implemented using accelerated tensor operations, and computation of K^L is typically faster than solving the system of linear equations.
- Deterministic and differentiable.
- Purely determined by hyperparameters of the deep neural network.

Draws from NNGP Prior, GP(0, K^L)



Analytic vs numerical construction



Core Result

Deep Neural Networks as Gaussian Processes

- ***(Radford Neal, 1994)*** Consider: a single-layer NN with i.i.d. random parameters and nonlinearity ϕ. The output is a sum of random terms. As the network width N increases, by the multivariate Central Limit Theorem (CLT), the function becomes a Gaussian process (GP). It is governed by its mean and covariance function, which is known analytically or can be computed.
- ***(Our work)*** consider a deep neural network with i.i.d random parameters (here, zero mean). It computes iteratively:

$$z_i^l(x) = b_i^l + \sum_{j=1}^{N} W_{ij}^l x_j^l(x), \quad x_j^l(x) = \phi(z_j^{l-1}(x)).$$

- In the limit of infinite width, we can show by induction, using the CLT, that the final output is also a GP.
- The covariance function is a "deep kernel". The covariance kernel for layer ℓ depends on the covariance kernel for layer ℓ-1.

$$K^l(x,x') \equiv \mathbb{E}\left[z_i^l(x) z_i^l(x')\right] = \sigma_b^2 + \sigma_w^2 \mathbb{E}_{z_i^{l-1}\sim\mathcal{GP}(0,K^{l-1})}\left[\phi(z_i^{l-1}(x))\phi(z_i^{l-1}(x'))\right]. \quad (*)$$

- The relationship is deterministic – which we denote by a function F_ϕ -- and is known analytically (e.g. for ReLU) or can be computed numerically.

$$z_i^l \sim \mathcal{GP}(0, K^l)$$
$$K^l(x,x') = \sigma_b^2 + \sigma_w^2 F_\phi\left(K^{l-1}(x,x'),\, K^{l-1}(x,x),\, K^{l-1}(x',x')\right)$$

Bayesian Inference

- Consider dataset $\mathcal{D}$ of size n, consisting of input, target pairs (x, t). Let z denote the model output.
- Bayesian inference can be done in parameter space or, equivalently, in function space. The predictions for test point x^* are:

$$P(z^*|\mathcal{D}, x^*) = \int d\mathbf{z}\, P(z^*|\mathbf{z}, \mathbf{x}, x^*)\, P(\mathbf{z}|\mathcal{D}) = \frac{1}{P(\mathbf{t})}\int d\mathbf{z}\, P(z^*, \mathbf{z}|x^*, \mathbf{x})\, P(\mathbf{t}|\mathbf{z})$$

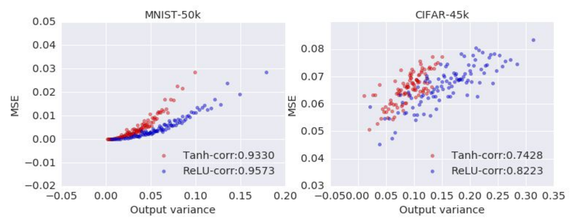
- The i.i.d. prior over parameters of a deep, infinitely wide neural network can be replaced with a GP prior over functions:

$$z^*, \boldsymbol{z}|x^*, \boldsymbol{x} \sim \mathcal{N}(0, \mathbf{K}) \qquad \mathbf{K} = \begin{bmatrix}K_{\mathcal{D},\mathcal{D}} & K_{x^*,\mathcal{D}}^T\\ K_{x^*,\mathcal{D}} & K_{x^*,x^*}\end{bmatrix}$$

- The integrals can be done exactly and the result, assur

$$z^*|\mathcal{D}, x^* \sim \mathcal{N}(\bar{\mu}, \bar{K})$$
$$\bar{\mu} = K_{x^*,\mathcal{D}}(K_{\mathcal{D},\mathcal{D}} + \sigma_\epsilon^2 \mathbb{I}_n)^{-1}\boldsymbol{t}$$
$$\bar{K} = K_{x^*,x^*} - K_{x^*,\mathcal{D}}(K_{\mathcal{D},\mathcal{D}} + \sigma_\epsilon^2 \mathbb{I}_n)^{-1} K_{x^*,\mathcal{D}}^T$$

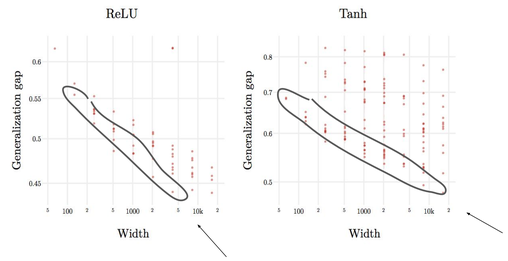
Finding 3: Gaussian Process uncertainty is predictive of test error



Experimental Findings

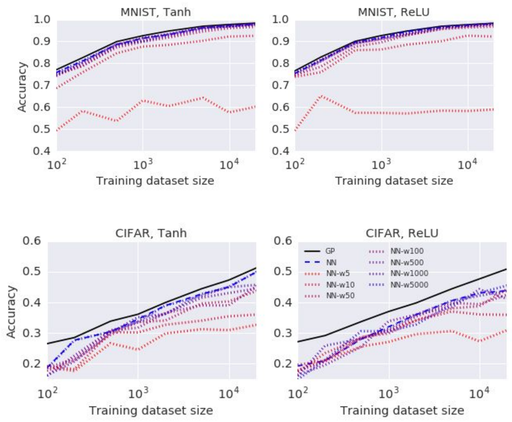
Observation: As neural networks become wider, they generalize better

- Trained many 5 layer networks of varying width with SGD on CIFAR-10
- Only networks which achieved 100% training accuracy plotted



Finding 1: As neural networks become wider their accuracy approaches that of the equivalent Gaussian Process

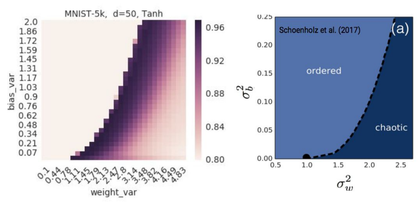
- GP : 10 independent outputs for each one-hot encoded class label
  - Grid search over weight/bias variances
- NN : Fully-connected network with varying depth/width trained on MSE loss
  - Hyperparameters optimized by Google Vizier



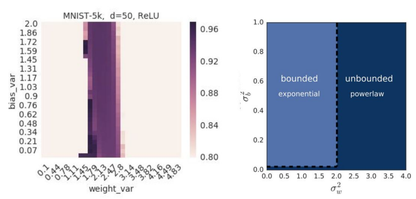
Finding 2: Gaussian Process performance governed by "Deep Information Propagation" phase diagram (S. Schoenholz, ICLR 2017)

- Training signal decay corresponds to degeneracy in Gaussian Process kernel

Tanh Phase Diagram



Relu Phase Diagram



Deep Neural Networks as Gaussian Processes

Jaehoon Lee*, Yasaman Bahri*, Roman Novak,
Samuel S. Schoenholz, Jeffrey Pennington, Jascha Sohl-Dickstein

*Equal contribution, work done as part of Google AI Residency, g.co/airesidency

To appear in ICLR 2018, https://arxiv.org/abs/1711.00165.
Code at: https//github.com/brain-research/nngp

Summary and Background

Summary

- There is a correspondence between single-layer neural networks and Gaussian processes (GPs) (Neal 1994):
  - Due to the Central Limit Theorem, an infinitely wide neural network with i.i.d. random parameters is a GP, with known mean and covariance function.
- We build an analogous correspondence for deep neural networks:
  - We can compute the covariance function of the deep neural network GP (NNGP) iteratively for the desired depth. It is valid for generic nonlinearities.
- Bayesian inference for neural networks is typically intractable to implement exactly. The NNGP correspondence, however, enables *exact Bayesian predictions* to be made through a few matrix computations, without any training.
- We study the performance of this approach on MNIST and CIFAR-10, for a range of hyperparameter settings including depth, nonlinearity, and weight/bias priors, and compare against fully-connected neural networks (NNs) trained with stochastic gradient-based optimization.
- The NNGP predictions are competitive and often outperform NN predictions. NN performance approaches that of the NNGP as width is increased.
- The Bayesian nature of the GP gives uncertainty estimates which are predictive of the true error on test points.
- We relate the performance to the understanding of signal propagation in random networks.

Deep Neural Networks as Gaussian Processes

- Consider: a single-layer NN with i.i.d. random parameters (Neal, 1994) and nonlinearity ϕ. The output is a sum of random terms. As the network width N increases, by the multivariate Central Limit Theorem (CLT), the function becomes a Gaussian process (GP). It is governed by its mean and covariance function, which is known analytically or can be computed.
- Consider a deep neural network with i.i.d random parameters (here, zero mean). It computes iteratively:

$$z_i^l(x) = b_i^l + \sum_{j=1}^{N} W_{ij}^l x_j^l(x), \quad x_j^l(x) = \phi(z_j^{l-1}(x)).$$

- In the limit of infinite width, we can show by induction, using the CLT, that the final output is also a GP.
- The covariance function is a "deep kernel". The covariance kernel for layer ℓ depends on the covariance kernel for layer ℓ-1.

$$K^l(x,x') \equiv \mathbb{E}\left[z_i^l(x) z_i^l(x')\right] = \sigma_b^2 + \sigma_w^2 \, \mathbb{E}_{z_i^{l-1} \sim \mathcal{GP}(0, K^{l-1})}\left[\phi(z_i^{l-1}(x))\phi(z_i^{l-1}(x'))\right]. \quad (*)$$

- The relationship is deterministic – which we denote by a function F_ϕ – and is known analytically (e.g. for ReLU) or can be computed numerically.

$$z_i^l \sim \mathcal{GP}(0, K^l)$$
$$K^l(x,x') = \sigma_b^2 + \sigma_w^2 \, F_\phi\Big(K^{l-1}(x,x'),\, K^{l-1}(x,x),\, K^{l-1}(x',x')\Big)$$

Bayesian Inference

- Consider dataset $\mathcal{D}$ of size n, consisting of input, target pairs (x, t). Let z denote the model output.
- Bayesian inference can be done in parameter space or, equivalently, in function space. The predictions for test point x^* are:

$$P(z^*|\mathcal{D}, x^*) = \int d\boldsymbol{z}\, P(z^*|\boldsymbol{z}, \boldsymbol{x}, x^*)\, P(\boldsymbol{z}|\mathcal{D}) = \frac{1}{P(\mathbf{t})} \int d\boldsymbol{z}\, P(z^*, \boldsymbol{z}|x^*, \boldsymbol{x})\, P(\mathbf{t}|\boldsymbol{z}).$$

- The i.i.d. prior over parameters of a deep, infinitely wide neural network can be replaced with a GP prior over functions:

$$z^*, \boldsymbol{z}|x^*, \boldsymbol{x} \sim \mathcal{N}(0, \mathbf{K}) \qquad \mathbf{K} = \begin{bmatrix} K_{\mathcal{D},\mathcal{D}} & K_{x^*,\mathcal{D}}^T \\ K_{x^*,\mathcal{D}} & K_{x^*,x^*} \end{bmatrix}$$

- The integrals can be done exactly and the result, assuming Gaussian observation noise, is:

$$z^*|\mathcal{D}, x^* \sim \mathcal{N}(\bar{\mu}, \bar{K})$$
$$\bar{\mu} = K_{x^*,\mathcal{D}}(K_{\mathcal{D},\mathcal{D}} + \sigma_\epsilon^2 \mathbb{I}_n)^{-1} \boldsymbol{t}$$
$$\bar{K} = K_{x^*,x^*} - K_{x^*,\mathcal{D}}(K_{\mathcal{D},\mathcal{D}} + \sigma_\epsilon^2 \mathbb{I}_n)^{-1} K_{x^*,\mathcal{D}}^T$$

Implementation

Complexity of Kernel construction

- For some class of nonlinearities (e.g. ReLU) integration (*) can be done analytically.
- For generic nonlinearities, the Gaussian integration can be done numerically with n_g sampling points.
- Naive Kernel construction for n_{train} and n_{test} data points at depth L has compute cost

$$\mathcal{O}\left(n_g^2 L (n_{\text{train}}^2 + n_{\text{train}} n_{\text{test}})\right)$$

- Preprocess inputs to constant norm and generate precomputed grid of n_v variance and n_c correlations.

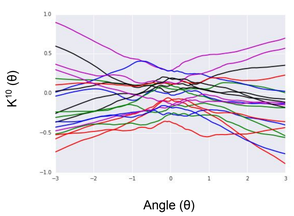
$$F_{ij} = \frac{\sum_{ab} \phi(u_a)\phi(u_b) \exp\left(-\frac{1}{2}\begin{bmatrix} u_a \\ u_b \end{bmatrix}^T \begin{bmatrix} s_i & s_i c_j \\ s_i c_j & s_i \end{bmatrix}^{-1} \begin{bmatrix} u_a \\ u_b \end{bmatrix}\right)}{\sum_{ab} \exp\left(-\frac{1}{2}\begin{bmatrix} u_a \\ u_b \end{bmatrix}^T \begin{bmatrix} s_i & s_i c_j \\ s_i c_j & s_i \end{bmatrix}^{-1} \begin{bmatrix} u_a \\ u_b \end{bmatrix}\right)}.$$

- Obtain Kernel elements by bi-linear interpolation over closest points in the grid.
- Compute cost is

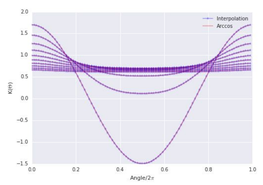
$$\mathcal{O}\left(n_g^2 n_v n_c + L(n_{\text{train}}^2 + n_{\text{train}} n_{\text{test}})\right)$$

- Can be implemented using accelerated tensor operations, and computation of K^L is typically faster than solving the system of linear equations.
- Deterministic and differentiable.
- Purely determined by hyperparameters of the deep neural network.

Draws from NNGP Prior, GP(0, K^L)



Analytic vs numerical construction



Experiments

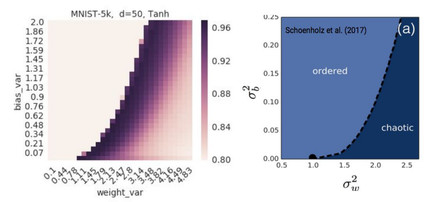
Description

- MNIST / CIFAR-10 dataset: 50k/10k/10k, 45k/5k/10k split, vary size of training set
- Nonlinearity: Relu, hyperbolic tangent(tanh), ...
- GP : 10 independent outputs for each one-hot encoded class labels
  - Grid search over weight/bias variances
- NN : Fully-connected network with varying depth/width trained on MSE loss
  - Hyperparameter optimized using Google Vizier (learning rate, weight/bias variances)

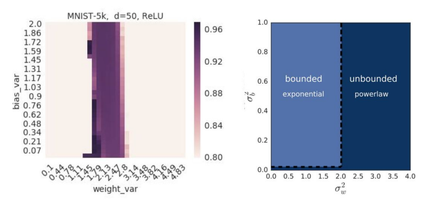
Table 1: The NNGP often outperforms finite width networks. Test accuracy on MNIST and CIFAR-10 datasets. The reported NNGP results correspond to the best performing depth, σ_w^2, and σ_b^2 values on the validation set. The traditional NN results correspond to the best performing depth, width and optimization hyperparameters. Best models for a given training set size are specified by (depth-width-σ_w^2-σ_b^2) for NNs and (depth–σ_w^2-σ_b^2) for GPs.

Num training	Model (ReLU)	Test accuracy	Model (tanh)	Test accuracy
MNIST:1k	NN-2-5000-3.19-0.00	0.9252	NN-2-1000-0.60-0.00	0.9254
	GP-20-1.45-0.28	**0.9279**	GP-20-1.96-0.62	0.9266
MNIST:10k	NN-2-2000-0.42-0.16	0.9771	NN-2-2000-2.41-1.84	0.9745
	GP-7-0.61-0.07	0.9765	GP-2-1.62-0.28	**0.9773**
MNIST:50k	NN-2-2000-0.60-0.44	0.9864	NN-2-5000-0.28-0.34	0.9857
	GP-1-0.10-0.48	0.9875	GP-1-1.28-0.00	**0.9879**
CIFAR:1k	NN-5-500-1.29-0.28	0.3225	NN-1-200-1.45-0.12	0.3378
	GP-7-1.28-0.00	0.3608	GP-50-2.97-0.97	**0.3702**
CIFAR:10k	NN-5-2000-1.60-1.07	0.4545	NN-1-500-1.48-1.59	0.4429
	GP-5-2.97-0.28	**0.4780**	GP-7-3.48-2.00	0.4766
CIFAR:45k	NN-3-5000-0.53-0.01	0.5313	NN-2-2000-1.05-2.08	0.5034
	GP-3-3.31-1.86	**0.5566**	GP-3-3.48-1.52	0.5558

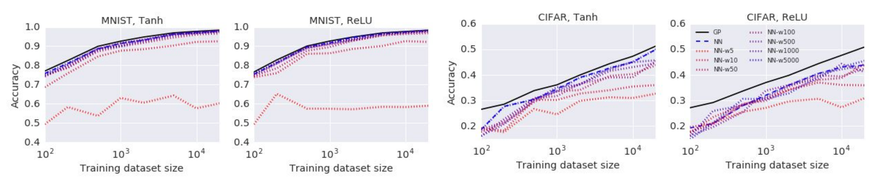
Tanh Phase Diagram



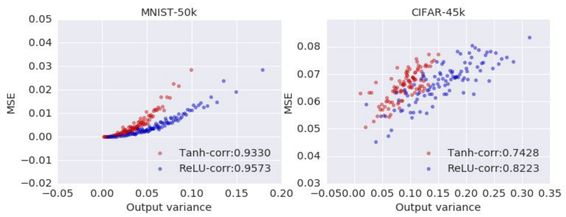
Relu Phase Diagram



Neural networks with increasing width approach Gaussian Process performance



Uncertainty estimates are predictive of test error



Future directions

- Scaling GP: structured approximation to kernel, variational approximation
- Extend to other architectures: CNN / RNN / Residual connection / Batch norm
- Model selection with likelihoods

References

Neal, Radford M. Bayesian learning for neural networks. Ph.D. Thesis, University of Toronto, 1994.
Williams, Christopher. Computing with infinite networks. NIPS 1997.
Cho, Y. and Saul, L.K. Kernel methods for deep learning. NIPS 2009.
Schoenholz, S. S., et al. Deep information propagation. ICLR 2017.
Golovin, D. et al. Google Vizier: A Service for Black-Box Optimization. KDD 2017.

Deep Gaussian Processes and Deep Neural Networks

Jaehoon Lee*, Yasaman Bahri*, Roman Novak,
Samuel S. Schoenholz, Jeffrey Pennington, Jascha Sohl-Dickstein

*Work done as part of Google Brain Residency, g.co/brainresidency

Summary and Background

Summary

- There is a correspondence between single-layer neural networks and Gaussian processes (GPs) (Neal 1994, Williams 1997):
  - Due to the Central Limit Theorem, an infinitely wide neural network with i.i.d random parameters is a GP, with known mean and covariance function.
- We build an analogous correspondence for deep neural networks and deep Gaussian processes:
  - We can compute the covariance function of the deep GP iteratively for the desired depth. It is valid for generic nonlinearities.
- Bayesian inference for neural networks is typically intractable to implement exactly. The deep GP correspondence, however, enables *exact Bayesian predictions* to be made through a few matrix computations, without any training!
- We study the performance of this approach on MNIST and CIFAR-10, for a range of hyperparameter settings including depth, nonlinearity, and weight/bias priors.
- We compare against conventional fully-connected neural networks (NNs) trained with stochastic gradient-based optimization. The deep GP computation is competitive and often outperforms NN.
- The NN performance approaches that of the deep GP as network width is increased.

Deep Gaussian Processes and Deep Neural Networks

- Consider a single-layer NN with i.i.d. random parameters (Williams, 1997) and nonlinearity ϕ. As the network width M increases, by the Central Limit Theorem, the function becomes a Gaussian process. It is governed by its mean and covariance function.

$$z_i^1 = b_i^1 + \sum_{j=1}^{M} W_{ij}^1 \phi\left(b_j^0 + \sum_k W_{jk}^0 x_k\right) \qquad K^1(x,x') \equiv \mathbb{E}\left[z_i^1(x) z_i^1(x')\right] = \sigma_b^2 + \sigma_w^2 \, \mathbb{E}_{W^0,b^0}\left[\phi(x;W^0,b^0)\phi(x';W^0,b^0)\right]$$

- Consider a deep neural network with i.i.d random parameters. It computes iteratively:

$$z_i^l = b_i^l + \sum_j W_{ij}^l \phi(z_j^{l-1})$$

- In the limit of large width, the neural network output is a *deep Gaussian proces* $z_i^l \sim \mathcal{GP}(0, K^l)$
- We construct the covariance function iteratively:

$$K^l(x,x') \equiv \mathbb{E}\left[z_i^l(x) z_i^l(x')\right] = \sigma_b^2 + \sigma_w^2 \, \mathbb{E}_{\boldsymbol{z}_{x,x'}^{l-1}}\left[\phi(z_i^{l-1}(x))\phi(z_i^{l-1}(x'))\right]$$

- For example, for the Relu activation function this covariance function can be calculated exactly (Cho and Saul, 2009):

$$K^l(x,x') = \sigma_b^2 + \frac{\sigma_w^2}{2\pi}\sqrt{K^{l-1}(x,x)K^{l-1}(x',x')}\, J_1(\theta_{x,x'}^{l-1})$$
$$J_1(\theta) \equiv \sin\theta + (\pi - \theta)\cos\theta$$
$$\theta_{x,x'}^l \equiv \cos^{-1}\left(\frac{K^l(x,x')}{\sqrt{K^l(x,x)K^l(x',x')}}\right)$$

Bayesian Inference

- Consider dataset D of size N and test point (x_{N+1}, t_{N+1}).
- Bayesian inference using a model parameterized by $\boldsymbol{\theta}$:

$$P(y_{N+1}|\mathcal{D}, x_{N+1}) = \frac{1}{P(\mathcal{D})}\int d\boldsymbol{\theta} P(y_{N+1}|\boldsymbol{\theta}, x_{N+1})\, P(\mathcal{D}|\boldsymbol{\theta})\, P(\boldsymbol{\theta})$$

- Equivalently, we can formulate in the hypothesis space y(x) directly.

$$P(y_{N+1}|\mathcal{D}, x_{N+1}) = \int d\boldsymbol{y}\, P(y_{N+1}|\boldsymbol{y}, x_{N+1})\, P(\boldsymbol{y}|\mathcal{D}) = \frac{1}{P(\mathcal{D})}\int d\boldsymbol{y}\, P(y_{N+1}, \boldsymbol{y})\, P(\mathcal{D}|\boldsymbol{y})$$

- We can replace the prior over parameters with a prior over functions. An i.i.d. prior over parameters of a *deep*, *infinitely* wide neural network is a deep Gaussian process prior over functions.

$$\mathbf{K} = \begin{bmatrix} K_{\mathcal{D},\mathcal{D}} & K_{N+1,\mathcal{D}}^T \\ K_{N+1,\mathcal{D}} & K_{N+1,N+1} \end{bmatrix}$$

- This simplifies the computation and the exact result is

$$y_{N+1}|\mathcal{D}, x_{N+1} \sim \mathcal{N}(\bar{\mu}, \bar{K})$$
$$\bar{\mu} = K_{N+1,\mathcal{D}}(K_{\mathcal{D},\mathcal{D}} + \sigma_\epsilon^2 \mathbf{I})^{-1}\boldsymbol{t}$$
$$\bar{K} = K_{N+1,N+1} - K_{N+1,\mathcal{D}}(K_{\mathcal{D},\mathcal{D}} + \sigma_\epsilon^2 \mathbf{I})^{-1}K_{N+1,\mathcal{D}}^T$$

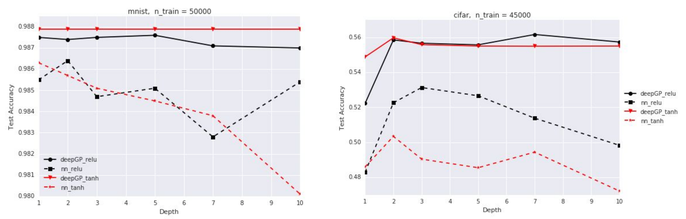
Experiments

Description

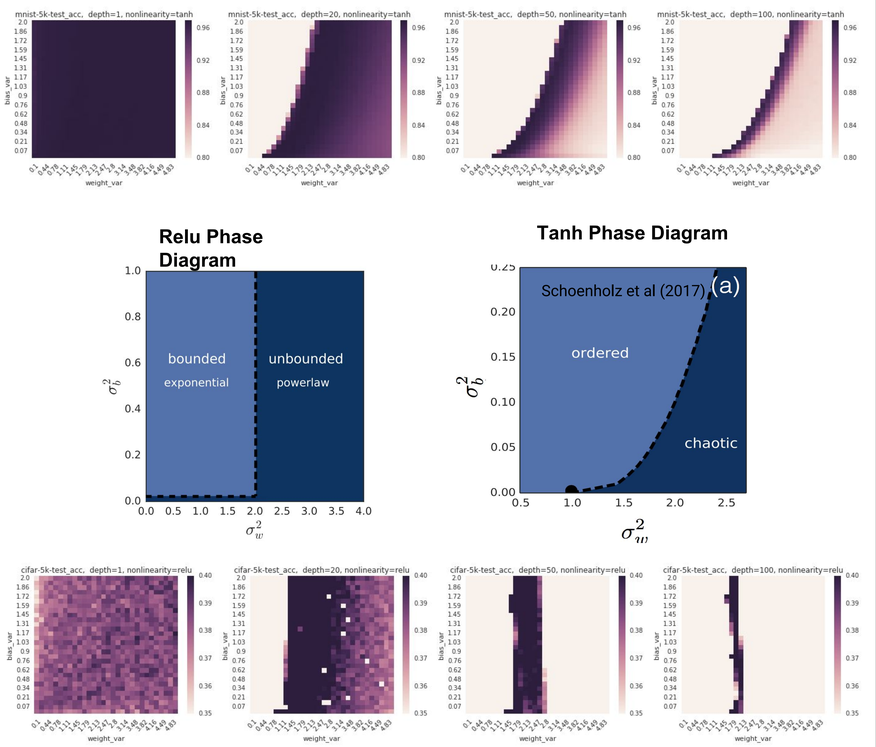
- MNIST / CIFAR-10 dataset: 50k/10k/10k, 45k/5k/10k split, vary size of training set
- Nonlinearity: Relu, hyperbolic tangent(tanh), ...
- GP : 10 independent outputs for each one-hot encoded class labels
  - Efficient numerics using interpolation grid to construct Kernels
  - Grid search over weight/bias variances
- NN : Fully-connected network with varying depth/width trained on MSE loss
  - Hyperparameter optimized using Google Vizier (learning rate, weight/bias variances)

Table 1: Results on MNIST and CIFAR-10 datasets. For a given training set size, the best performing result is highlighted. The reported deep GP results correspond to the best performing depth, σ_w^2, and σ_b^2 values on the validation set. The traditional NN results correspond to the best performing depth, width and optimization hyperparameters.

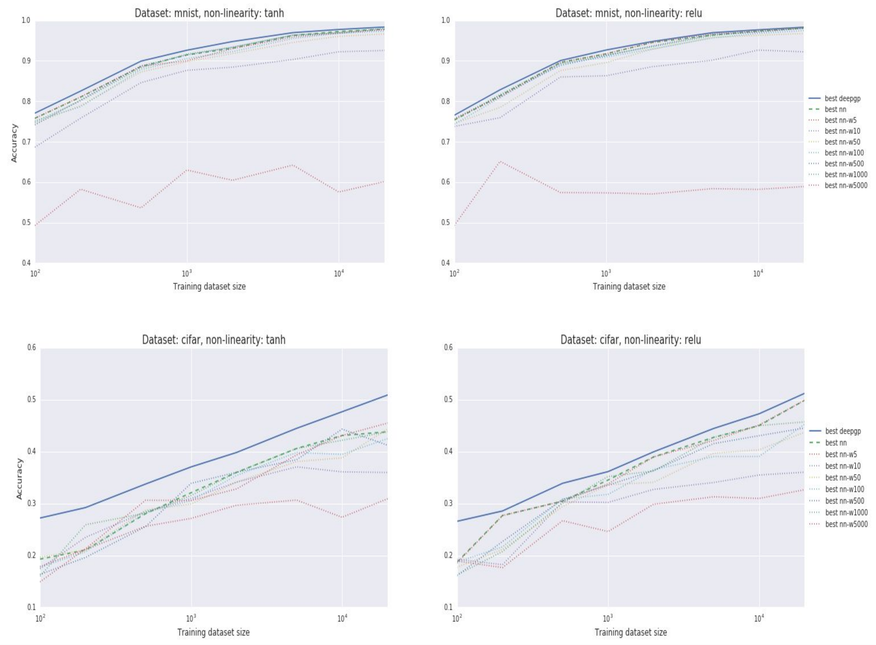
Num training	Model (relu)	Test accuracy	Model (tanh)	Test accuracy
mnist-100	NN-2-5000-0.10-0.11	**0.7786**	NN-1-500-1.48-0.61	0.7766
	GP-100-1.79-0.83	0.7735	GP-100-3.14-0.97	0.7736
mnist-1k	NN-2-5000-3.19-0.00	0.9252	NN-2-1000-0.60-0.00	0.9254
	GP-20-1.45-0.28	**0.9279**	GP-20-1.96-0.62	0.9266
mnist-10k	NN-2-2000-0.42-0.16	0.9771	NN-2-2000-2.41-1.84	0.9745
	GP-7-0.61-0.07	0.9765	GP-2-1.62-0.28	**0.9773**
mnist-50k	NN-2-2000-0.60-0.44	0.9864	NN-2-5000-0.28-0.34	0.9857
	GP-1-0.10-0.48	0.9875	GP-1-1.28-0.00	**0.9879**
cifar-100	NN-5-500-1.88-1.00	0.2586	NN-2-200-3.22-2.09	0.2470
	GP-3-4.49-0.97	0.2673	GP-10-3.65-1.17	**0.2718**
cifar-1k	NN-5-500-1.29-0.28	0.3225	NN-1-200-1.45-0.12	0.3378
	GP-7-1.28-0.00	0.3608	GP-50-2.97-0.97	**0.3702**
cifar-10k	NN-5-2000-1.60-1.07	0.4545	NN-1-500-1.48-1.59	0.4429
	GP-5-2.97-0.28	**0.4780**	GP-7-3.48-2.00	0.4766
cifar-45k	NN-3-5000-0.53-0.01	0.5313	NN-2-2000-1.05-2.08	0.5034
	GP-3-3.31-1.86	**0.5566**	GP-3-3.48-1.52	0.5558



Experiments



Gaussian process as a limit of wide neural networks



References

Williams, Christopher. Computing with infinite networks. NIPS 1997.
Neal, Radford M. Bayesian learning for neural networks. Ph.D. Thesis, University of Toronto, 1994.
Mackay, David JC. Gaussian processes -- a replacement for supervised neural networks? NIPS Tutorial, 1997.
Schoenholz, S. S., et al. Deep information propagation. ICLR 2017.
Cho, Y. and Saul, L.K. Kernel methods for deep learning. NIPS 2009.
Golovin, D. et al. Google Vizier: A Service for Black-Box Optimization. KDD 2017.

PCA of High Dimensional Random Walks with Comparison to Neural Network Training

Joseph Antognini*, Jascha Sohl-Dickstein

*Work done as part of Google AI Residency, g.co/airesidency

Summary

- Deep neural networks (NNs) are high dimensional objects which makes visualizing the training process difficult. One approach some authors have proposed is to perform principal components analysis (PCA) on the trajectory of the parameters of the NNs and visualize the projection onto the lowest PCA dimensions (e.g., Lorch 2016; Lipton 2016; Li et al. 2018). When projected onto the lowest PCA components the trajectory appears smooth and contains a large fraction of the variance (typically over 80% in the first two PCA components).

- We show that when PCA is applied to a random walk in the limit of an infinite number of dimensions:
  - Approximately 60% of the variance is in the first PCA component; 80% of the variance is in the first two PCA components.
  - The projection of a random walk onto any two PCA components is a Lissajous curve.
  - These results are *independent* of the noise distribution.
  - These results also apply to a random walk with momentum
  - These results also apply to a random walk in a quadratic well (initialized near the origin) in the early stages of the trajectory.

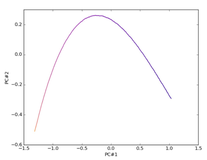
- We perform PCA on the parameter trajectories of a linear model trained on wCIFAR-10, a small NN trained on MNIST, and ResNet-50 trained on ImageNet and find that the PCA projected trajectories of the parameters closely resemble PCA-projected random walks, especially in the early stages of training.

Motivation

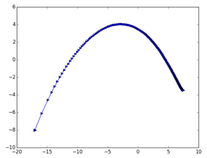
- NN training is a high dimensional stochastic process. This makes understanding and visualizing the training procedure difficult.

- Several authors have visualized NN training by performing PCA on the training trajectory and projecting onto the first few components (e.g., Lorch 2016; Lipton 2016; Li et al. 2018). These authors have found:

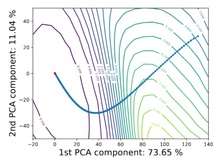
  - Although the trajectory when projected onto random dimensions looks similar to a random walk, the PCA projected trajectories are very smooth.
  - A large fraction of the variance is in the first few PCA components (between 80--90% in the first few).
  - This suggests an interpretation that although there is a stochastic component to NN training, in general the NN predominantly moves in a small number of dimensions, and does so in a consistent manner.



Lorch (2016)



Lipton (2016)



Li et al. (2016)

- *We consider what happens when you apply the same technique to a high dimensional random walk and find qualitatively the same results.* This suggests that this technique indicates that upon PCA, NN training can be effectively modeled as a biased random walk.

Core Result

Preliminaries

An n-dimensional random walk can be written as follows:

$$\mathbf{x}_t = \mathbf{x}_{t-1} + \boldsymbol{\xi}_t, \qquad \boldsymbol{\xi}_t \sim \mathcal{P},$$

where $\mathbf{x}_0 = \mathbf{0}$ and $\mathcal{P}$ is an arbitrary probability distribution. We can also write this in matrix form:

$$\mathbf{SX} = \mathbf{R}, \qquad \mathbf{S} \equiv \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ -1 & 1 & 0 & \ddots & \vdots \\ 0 & -1 & 1 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & -1 & 1 \end{pmatrix},$$

Performing PCA is equivalent to finding the eigenvalues and eignevectors of the following matrix:

$$\hat{\mathbf{X}}\hat{\mathbf{X}}^T = \mathbf{C}\mathbf{S}^{-1}\mathbf{R}\mathbf{R}^T\mathbf{S}^{-T}\mathbf{C}, \qquad \hat{\mathbf{X}} = \mathbf{C}\mathbf{X}, \quad \mathbf{C} \equiv \mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^T.$$

In the limit that $d \gg n$ we will have $\mathbf{R}\mathbf{R}^T \to \mathbf{I}$ because the off diagonal terms will be $\mathbb{E}[\xi_i]^2 = 0$ whereas the on diagonal terms will be $\mathbb{E}[\xi^2] = \sum_{i=0}^{d} \mathbb{V}[\xi_i] = 1$

Asymptotic convergence to circulant matrices

In the limit that $n \to \infty$ banded Toeplitz matrices become asymptotically convergent to circulant matrices. This implies that they have the same inverses and distribution of eigenvalues and eigenvectors as the corresponding circulant matrices.

The eigenvalues of a circulant matrix with entries $c_0, c_1, \ldots$ are

$$\lambda_{\text{circ},k} = c_0 + c_{n-1}\omega_k + c_{n-2}\omega_k^2 + \ldots + c_1\omega_k^{n-1},$$

Using this, we can derive the variance in each PCA component:

$$\lambda_{\hat{\mathbf{X}}\hat{\mathbf{X}}^T,k} = \frac{1}{2}\left[1 - \cos\left(\frac{\pi k}{n}\right)\right]^{-1}$$

This implies that the explained variance ratio of each component is:

$$\boxed{\rho_k = \frac{6}{\pi^2 k^2}}$$

Since the eigenvectors of circulant matrices are simply Fourier modes, we show that the projection onto a PCA component is given by:

$$\boxed{\mathbf{X}_{\text{PCA},k} = \sqrt{\frac{2\lambda_k}{n}}\cos\left(\frac{\pi k t}{n}\right)}$$

This implies that the projection of a high dimensional random walk onto PCA components is a Lissajous curve.

Generalizations

PCA variances of a random walk with momentum:

$$\begin{aligned} \mathbf{v}_t &= \gamma\mathbf{v}_{t-1} + \boldsymbol{\xi}_t \\ \mathbf{x}_t &= \mathbf{x}_{t-1} + \mathbf{v}_t. \end{aligned}$$

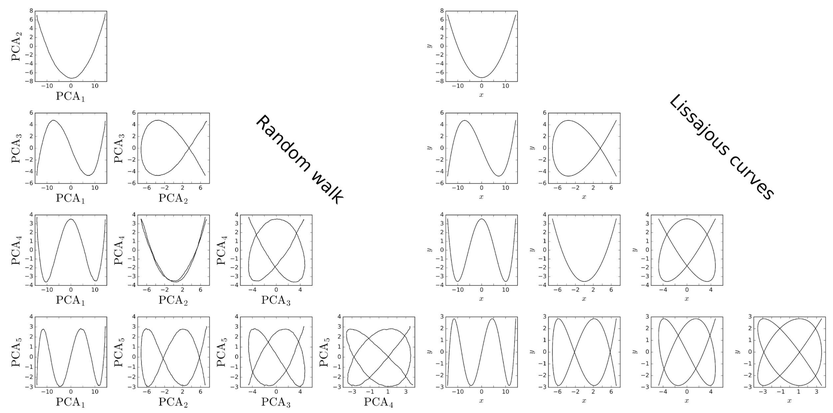
$$\lambda_k = \frac{1}{2}\left[1 + \gamma + \gamma^2 - (1+\gamma)^2\cos\left(\frac{\pi k}{n}\right) + \gamma\cos\left(\frac{2\pi k}{n}\right)\right]^{-1}$$

PCA variances of an Ornstein-Uhlenbeck process: $\mathbf{x}_t = (1-\alpha)\mathbf{x}_{t-1} + \boldsymbol{\xi}_t$,

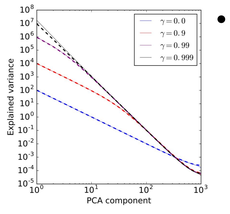
$$\lambda_{\text{OU},k} = \left[1 + (1-\alpha)^2 - 2(1-\alpha)\cos\left(\frac{2\pi k}{n}\right)\right]^{-1} \simeq \left[\frac{4\pi^2 k^2(1-\alpha)}{n^2} + \alpha^2\right]^{-1}$$

Experiments

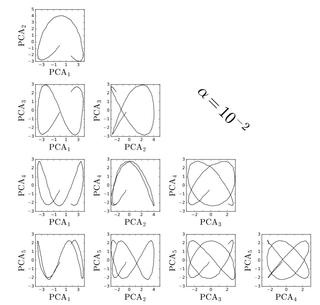
- High dimensional random walks projected onto PCA components are Lissajous curves:



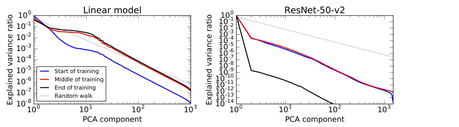
- The explained variance of each PCA component we observe closely matches our predictions for both a high dimensional random walk and a high dimensional random walk with momentum:

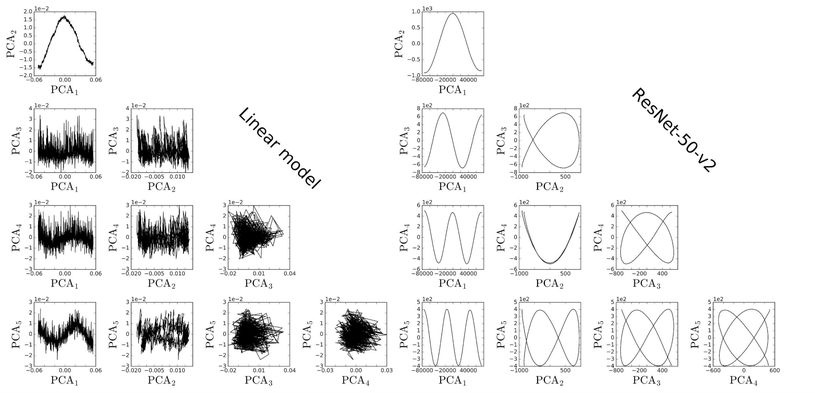


- The PCA projected trajectories of an Ornstein-Uhlenbeck process resemble perturbed Lissajous curves:



- PCA projections of the training trajectory of a large neural network resemble Lissajous curves
  - Trained a linear model on CIFAR-10 for 10,000 steps, collected parameters from the middle 1000 steps
  - Trained ResNet-50-v2 on ImageNet for 150,000 steps, collected parameters from the middle 15,000 steps
- Explained variance ratio distribution is comparable to a random walk early in training with the exception of the first component.





References

- Lorch,E., Visualizing deep network training trajectories with pca. In *ICML Workshop on Visualization for Deep Learning*, 2016
- Lipton, Z., Stuck in a what? Adventures in weight space. *arXiv preprint arXiv:1602.07320*, 2016
- Li, H., Xu, Z., Taylor, G., and Goldstein, T., Visualizing the loss landscape of neural nets. In *International Conference on Learning Representations*, 2018