Random Matrix Theory and Machine Learning

Manvitha Ponnapati

May 2018

This project provides a broad perspective on the use of random matrix theory techniques to study neural networks. From our study, we found that there are three different matrices related to neural networks that are of interest

1. Activation functions [1]
2. Hessians of the loss functions[2]
3. Jacobians of the loss functions[3]

In order, to understand why it's relevant to study eigenvalue spectrum of these matrices, it helps to first understand how eigenvalues and eigenvectors control gradient descent. If X is the input matrix to a network then eigenvalues of the covariance matrix $X'X$ influence the convergence of loss in neural networks. During the learning process for a neural network,the goal is to minimize the loss E(W) described by

$$\mathbf{E(W) = (1/2m) * \sum_{p=1}^{m} \left(Y^p - f(W, X^p) \right)^2}$$

where m is the number of samples with n features and p is the p^{th} sample. Y^p is the actual value and $f(W, X^p)$is the output of the layer. Using gradient descent, the iterative update step changes this weight function using

$$W(p+1) = W(p) - \alpha \nabla E$$

where the α is the step size of update and drives the learning rate of the network.
The loss function above can be written in it's quadratic form as

$$\mathbf{E(w) = (1/2) * W^T A W - B^T W}$$

where A is the covariance matrix of the input

$$\mathbf{A_{ij} = (1/m) * \sum_{k=1}^{m} x_i^k x_j^k}$$

and

$$\mathbf{B_i = (1/m) * \sum_{k=1}^{m} y_k x_i^k}$$

Then the solution space W^* that minimizes E(W) is AW = B and using this in ∇E the weight update becomes

$$\mathbf{W(p+1) = W(p) - \alpha(Aw^k - b)}$$

We can view this in the space of eigenvectors of A by using eigenvalue decomposition (Note: A is symmetric) .

$$\mathbf{A = Q \Lambda Q'}$$

By using this decomposition we can perform a change of basis to write x^p as

$$\mathbf{x^p} = \mathbf{Q^T}(\mathbf{w^p} - \mathbf{w}^*)$$

$\mathbf{x_i^{p+1}} = \mathbf{x_i^p} - \alpha\lambda_i\mathbf{x_i^p} = (\mathbf{1} - \alpha\lambda_\mathbf{i})^{\mathbf{p+1}}x_i^0$ and finally we can write

$$\mathbf{w^p} - \mathbf{w}^* = \sum_{\mathbf{i}}^{\mathbf{n}} \mathbf{x_i^0}(\mathbf{1} - \alpha\lambda_\mathbf{i})^\mathbf{p}\mathbf{q_i}$$

α here is the learning rate/step size

As you can clearly see the rate of convergence depends on the eigenvalues. For most step sizes , the largest eigenvalues will converge faster[4]. Code eigenvalues_convergence.jl will allow you to see this dependence of convergence on the eigenvalues

I hope the above discussion convinced you why eigenvalue spectrum can be important to the learning task as we navigate through the layers. The rest of this report is organized as follows

Section 1: Neural Network Loss Surfaces
Section 2: Nonlinear random matrix theory for deep learning
Section 3: Dynamical isometry in neural networks

Section 1: Neural Network Loss Surfaces [1]

Motivation : Applying moments method in random matrix theory to study deep learning architectures
Key contribution : Derive a representation for Stieltjes transform for , which can be used to find the spectral density of gram matrix M, where m is the number of samples and n is the number of parameters. These results are obtained assuming both n,m -> and n\m =, a finite constant
Notation : Input is represented by X $R^{n_0 x m}$and X can be made of i.i.d elements with mean 0 and variance$(\sigma_x)^2$, m is the number of samples,$n_0$is the number of features. Weights (we are studying a single layer case) is represented by W $R^{n_1 x n_0}$ W can be made of i.i.d elements with mean 0 and variance $\frac{(\sigma_w)^2}{n_0}$ where $\phi = \frac{n_0}{m}\psi = \frac{n_0}{n_1}$
Z = WX is the matrix before pointwise activation function is applied . f is the activation function with zero mean and finite moments . Two parameters are defined on f ,η and τ where

$$\eta = \int \frac{dz}{\sqrt{2\pi}} e^{-\frac{z^2}{2}} f(\sigma_w, \sigma_x, z)^2 \quad \tau = \left[\sigma_w \sigma_x \int \frac{dz}{\sqrt{2\pi}} e^{-\frac{z^2}{2}} f'(\sigma_w, \sigma_x, z)\right]^2$$

Key result : Using the expansion of Stieltjes transform into moments of M and computing the moments using the trace of M^k for k moment, the paper obtains Stieltjes transform of the gram matrix M as

$$G(z) = \frac{\psi}{z} P(\frac{1}{z\psi}) + \frac{1-\psi}{z}$$

where

$$P = 1 + (\eta - \tau)tP_\psi P_\phi + \frac{P_\psi P_\phi t\tau}{1 - P_\psi P_\phi t\tau}$$

and

$$P_\phi = 1 + (P-1)\phi, P_\psi = 1 + (P-1)\psi$$

Clearly G(z) depends on the two parameters η and τ of the activation function. The paper then continues to study what happens to the spectral density of the gram matrix and uses that to obtain a set of conditions for designing better activation functions

Case 1 : if $\eta = \tau$ Using hermite expansion, η can be expressed in terms of coefficients of the hermite expansion as $\eta = \sum f_n(x)^2$ and $\tau = f_1^2$ and it is clear that $\eta >= \tau$ and they are equal when $f(x) = f_1 H_1(x) = f_1 x$. Therefore in this the activation function is a linear function which means gram matrix is just WXX'W' and since X is i.i.d the spectral density and G(z) reduce to that of a Wishart matrix

Case 2 : if $\tau = 0$ the equation becomes $zG^2 + \big((1 - \frac{\psi}{\phi})z - 1\big) + \frac{\psi}{\phi} = 0$ this is the stieltjes transform of the MP distribution with parameter $\frac{\phi}{\psi}$ and for a particular limiting case when $\psi = 1$ i.e when $n_0 = n_1$, the spectrum of XX^T and YY^T are the same

The activation functions that belong to case 2 are the ones that preserve the eigenvalue spectrum/singular values as the matrix propagates through the network. The code attached to the report allows you observe these properties

1. Code ***eigenvalue_singular_value.jl*** ->You can observe how different the spectrum is compared to the Marcenko pastur distribution results for an activation function τ = 0 . For various number of epochs.
2. Code ***activation_functions.jl*** -> Will plot several other proposed activation functions in the graph
3. Code ***experiments_mnist_new_activation.jl*** -> Will run a simple mnist classification task with the activation function from this paper and a regular relu to see that they both perform similarly in this task.

Section 2: Neural Network Loss Surfaces [2]

Motivation : Provide a practical setting to study loss surfaces of the neural networks.
Key contribution : Describe loss surfaces in terms of eigenvalue spectrum of the hessian matrix at it's critical points and achieve an empirical function for energy of minimizers.
Discussion:
2.1 Hessian of the loss function H can be decomposed into two components
$H = H_0 + H_1$

H_0 is related to the jacobian by $H_0 = 1/m * [J * J']_{\alpha,\beta}$ where α, β are parameters related to the network.

Primary assumptions in Pennington 17

1. The matrices H0 and H1 are freely independent
2. The residuals are i.i.d. normal random variables with tunable variance governed by , eiμ ~ N (0, 2). This assumption allows the gradient to vanish in the large m limit, specifying our analysis to critical points.
3. The data features are i.i.d. normal random variables.
4. The weights are i.i.d. normal random variables.
5. At critical points, J and H1 are i.i.d normal random variables

Under these assumptions, H_0 becomes a real wishart matrix and H_1 becomes a real wigner matrix. Therefore, spectrum of H depends on the spectrum of H_0 and H_1. If we assume scaling factors of $\sigma_{H_0} = 1$ and $\sigma_{H_1} = \sqrt{(2 * \epsilon)}$. The semi circle part of the distribution can be described by

$$\rho_{SC} = (1/2\pi\sigma^2) * \sqrt{4\sigma^2 - \lambda^2} if \mid \lambda \mid \leq 2\sigma \ .$$

and

$$\rho_{\mathrm{MP}}(\lambda; \sigma, \phi) = \begin{cases} \rho(\lambda) & \text{if } \phi < 1 \\ (1 - \phi^{-1})\delta(\lambda) + \rho(\lambda) & \text{otherwise} \end{cases}$$

As you can see SC part of the hessian doesn't depend on the parameter related to input size ϕ . Using this decomposition of H and also the properties of R-transform , we can describe the R transform for H as $\frac{1}{1 - z\phi} + 2\epsilon * z$ and Stieltjes transform G(z) then satisfies the following cubic equation

$$2\epsilon\phi G_H^3 - (2\epsilon + z\phi)G_H^2 + (z + \phi - 1)G_H^1 + 1 = 0$$

where ϵ is the energy of the critical point. You can see that as ϵ increases the G(z) starts describing the behavior of a semicircle and more and more negative eigenvalues emerge. The negative eigenvalues or the index of critical points of the hessian matrix are important because they describe the number of descent directions.

2.2 : The index of critical points typically grows rapidly with energy, so that critical points with many descent directions have large loss values. Using the spectral density from above - calculating this is as simple as integration over (-infinity,0) . Using the we can derive the critical energy ϵ_c. Below this - all critical points are minimizers
ϵ_c is the energy below which all critical points are minimizers. The number of descent directions depends upon the negative eigenvalues of the hessian matrix .

$$\epsilon_c = \frac{1}{16}(1 - 20\phi - 8\phi^2 + (1 + 8\phi)^{3/2})$$

A simpler version can be calculated by finding the real roots to the cubic equation above Z=0

There ϵ_c vanishes cubically as ϕ approaches 1

$$\epsilon_c = \frac{2}{27}(1-\phi)^3 + O(1-\phi)^4$$

1. Code ***eigenvalues_hessian.jl*** ->You can observe the eigenvalues of hessian change based on the energy parameter
2. Code ***index_critical_point.jl*** -> you can observe the difference between linear and non linear approximators
3. Code ***experiments_with_cifar.jl*** -> Shows you the free independence relationship between H0 and H1

Section 3: Dynamical isometry in neural networks [3]

Motivation : Dynamical isometry preserves error norms to be preserved while backpropagating. This can help with the problem of exploding/vanishing gradients. What kind of activation functions can achieve this property ?

Key contribution : Shows that ReLU networks can never achieve isometry whereas sigmoidal networks can under orthogonal weight initialization. Also shows why distribution of singular values of jacobian is so important.

Discussion:

If $W^l R\epsilon N x N$, bias vectors b^l , pre-activations h^l and post-activations x^l, then the input output jacobian is given by

$$\mathbf{J} = \frac{\partial \mathbf{x}^L}{\partial \mathbf{h}^0} = \prod_{l=1}^{L} \mathbf{D}^l \mathbf{W}^l.$$

where D^l is a diagonal matrix with entries $D^l_{ij} = \psi 0(h^l_i)\delta_{ij}$. The goal is to compute the singular value spectrum of JJ^T. The paper arrives at a result for this as a function of D^2 and WW^T by using principles of free probability as

$$S_{JJ^T} = \prod_{l=1}^{L} S_{W_l W_l^T} S_{D_l^2} = S^L_{WW^T} S^L_{D^2}$$

Where S is the S-transform. We can then calculate the density from the S-transform for the jacobian. We first calculate the spectral distributions of D^2 and WW^T and calculate their S-transforms. Multiplying those S-transforms will give us the S-transform of the jacobian which we can invert to compute the limiting spectral density. Refer to [3] to observe how this S-transform determines the dynamical isometry in case of linear networks, relu networks and for orthogonal weights during initialization.

1. Code **jacobian_singular_values.jl** for experimental investigation of above results. Note: work in progress as of writing this report

All code: https://github.com/ManvithaPonnapati/RandomMatrixTheoryAndMachineLearning.git

References:

1. Pennington, J., & Worah, P. (2017). Nonlinear random matrix theory for deep learning. In Advances in Neural Information Processing Systems (pp. 2634-2643).
2. Pennington, J., & Bahri, Y. (2017, July). Geometry of neural network loss surfaces via random matrix theory. In International Conference on Machine Learning (pp. 2798-2806).
3. Pennington, J., Schoenholz, S., & Ganguli, S. (2017). Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice. In Advances in neural information processing systems (pp. 4788-4798).
4. Goh, "Why Momentum Really Works", Distill, 2017. http://doi.org/10.23915/distill.00006