An Investigation into Neural Net Optimization via Hessian Eigenvalue Density

Behrooz Ghorbani

Department of Electrical Engineering
Stanford University
(Joint Work with Shankar Krishnan & Ying Xiao from Google Research)

June 2019
Gradient descent and its variants are the most popular method of optimizing neural networks.
Gradient descent and its variants are the most popular method of optimizing neural networks.

The performance of these optimizers is highly dependent on the local curvature of the loss surface.
Overview

- Gradient descent and its variants are the most popular method of optimizing neural networks.

- The performance of these optimizers is highly dependent on the local curvature of the loss surface → important to study the loss curvature
Gradient descent and its variants are the most popular method of optimizing neural networks. The performance of these optimizers is highly dependent on the local curvature of the loss surface → important to study the loss curvature.

We present a scalable algorithm for computing the full eigenvalue density of the Hessian for deep neural networks.
Overview

- Gradient descent and its variants are the most popular method of optimizing neural networks.

- The performance of these optimizers is highly dependent on the local curvature of the loss surface \( \rightarrow \) important to study the loss curvature

- We present a scalable algorithm for computing the full eigenvalue density of the Hessian for deep neural networks.

- We leverage this algorithm to study the effect of architecture / hyper-parameter choices on the optimization landscape.
\( \theta \in \mathbb{R}^n \) is the model parameter. \( L(\theta) \equiv \frac{1}{N} \sum_{i=1}^{N} L(\theta, (x_i, y_i)) \).

The Hessian matrix, \( H \), is an \( n \times n \) symmetric matrix of second derivatives:

\[
H(\theta_t)_{i,j} = \frac{\partial^2 L}{\partial \theta_i \partial \theta_j} |_{\theta = \theta_t}
\]

represents the (local) loss curvature at point \( \theta \).

\( H(\theta) \) has eigenvalue-eigenvector pairs \( (\lambda_i, q_i) \)

\( i=1 \) with \( \lambda_1 \geq \lambda_2 \cdots \geq \lambda_n \).

\( \lambda_i \) is the curvature of the loss in direction of \( q_i \) in the neighborhood of \( \theta \).

We focus on estimating the empirical distribution of \( \lambda_i \) as a concrete way to study the loss curvature.
Basic Definitions

- $\theta \in \mathbb{R}^n$ is the model parameter. $L(\theta) \equiv \frac{1}{N} \sum_{i=1}^{N} L(\theta, (x_i, y_i))$.
- The Hessian matrix, $H$, is an $n \times n$ symmetric matrix of second derivatives:

$$H(\theta_t)_{i,j} = \frac{\partial^2 L}{\partial \theta_i \partial \theta_j} \bigg|_{\theta = \theta_t}$$
Basic Definitions

- $\theta \in \mathbb{R}^n$ is the model parameter. $L(\theta) \equiv \frac{1}{N} \sum_{i=1}^{N} L(\theta, (x_i, y_i))$.
- The Hessian matrix, $H$, is an $n \times n$ symmetric matrix of second derivatives:
  \[
  H(\theta_t)_{i,j} = \frac{\partial^2 L}{\partial \theta_i \partial \theta_j} \bigg|_{\theta=\theta_t}
  \]
- $H(\theta)$ represents the (local) loss curvature at point $\theta$. 

Behrooz Ghorbani  
Hessian Spectral Density  
June 2019  3 / 18
Basic Definitions

- $\theta \in \mathbb{R}^n$ is the model parameter. $L(\theta) \equiv \frac{1}{N} \sum_{i=1}^{N} L(\theta, (x_i, y_i))$.
- The Hessian matrix, $H$, is an $n \times n$ symmetric matrix of second derivatives:

$$H(\theta_t)_{i,j} = \left. \frac{\partial^2 L}{\partial \theta_i \partial \theta_j} \right|_{\theta=\theta_t}$$

- $H(\theta)$ represents the (local) loss curvature at point $\theta$.
- $H(\theta)$ has eigenvalue-eigenvector pairs $(\lambda_i, q_i)_{i=1}^{n}$ with $\lambda_1 \geq \lambda_2 \cdots \geq \lambda_n$. 
Basic Definitions

- $\theta \in \mathbb{R}^n$ is the model parameter. $L(\theta) \equiv \frac{1}{N} \sum_{i=1}^{N} L(\theta, (x_i, y_i))$.
- The Hessian matrix, $H$, is an $n \times n$ symmetric matrix of second derivatives:

$$H(\theta_t)_{i,j} = \frac{\partial^2 L}{\partial \theta_i \partial \theta_j} \bigg|_{\theta=\theta_t}$$

- $H(\theta)$ represents the (local) loss curvature at point $\theta$.
- $H(\theta)$ has eigenvalue-eigenvector pairs $(\lambda_i, q_i)_{i=1}^{n}$ with $\lambda_1 \geq \lambda_2 \cdots \geq \lambda_n$.
- $\lambda_i$ is the curvature of the loss in direction of $q_i$ in the neighborhood of $\theta$. 
Basic Definitions

- $\theta \in \mathbb{R}^n$ is the model parameter. $L(\theta) \equiv \frac{1}{N} \sum_{i=1}^{N} L(\theta, (x_i, y_i))$.
- The Hessian matrix, $H$, is an $n \times n$ symmetric matrix of second derivatives:
  \[ H(\theta_t)_{i,j} = \frac{\partial^2 L}{\partial \theta_i \partial \theta_j} |_{\theta_t} \]
- $H(\theta)$ represents the (local) loss curvature at point $\theta$.
- $H(\theta)$ has eigenvalue-eigenvector pairs $(\lambda_i, q_i)_{i=1}^{n}$ with $\lambda_1 \geq \lambda_2 \cdots \geq \lambda_n$.
- $\lambda_i$ is the curvature of the loss in direction of $q_i$ in the neighborhood of $\theta$.
- We focus on estimating the empirical distribution of $\lambda_i$ as a concrete way to study the loss curvature.
The eigenvalue distribution function of $H$ is defined as

$$
\phi(t) = \frac{1}{n} \sum_{i=1}^{n} \delta(t - \lambda_i)
$$
The eigenvalue distribution function of $H$ is defined as

$$\phi(t) = \frac{1}{n} \sum_{i=1}^{n} \delta(t - \lambda_i)$$

Let $f_\sigma(x) = \frac{1}{\sigma \sqrt{2\pi}} \exp\left(-\frac{x^2}{2\sigma^2}\right)$ be the Gaussian density.
Hessian Computation in Deep Networks

- The eigenvalue distribution function of $H$ is defined as

$$\phi(t) = \frac{1}{n} \sum_{i=1}^{n} \delta(t - \lambda_i)$$

- Let $f_\sigma(x) = \frac{1}{\sigma \sqrt{2\pi}} \exp\left(-\frac{x^2}{2\sigma^2}\right)$ be the Gaussian density.

$$\phi(t) = \frac{1}{n} \sum_{i=1}^{n} \delta(t - \lambda_i)$$
Hessian Computation in Deep Networks

The eigenvalue distribution function of $H$ is defined as

$$
\phi(t) = \frac{1}{n} \sum_{i=1}^{n} \delta(t - \lambda_i)
$$

Let $f_{\sigma}(x) = \frac{1}{\sigma \sqrt{2\pi}} \exp\left(-\frac{x^2}{2\sigma^2}\right)$ be the Gaussian density.

$$
\phi(t) = \frac{1}{n} \sum_{i=1}^{n} \delta(t - \lambda_i)
$$
The eigenvalue distribution function of $H$ is defined as

$$\phi(t) = \frac{1}{n} \sum_{i=1}^{n} \delta(t - \lambda_i)$$

Let $f_\sigma(x) = \frac{1}{\sigma \sqrt{2\pi}} \exp\left(-\frac{x^2}{2\sigma^2}\right)$ be the Gaussian density.

$$\phi(t) = \frac{1}{n} \sum_{i=1}^{n} \delta(t - \lambda_i) \quad \xrightarrow{\text{Convolution with Gaussian}} \quad \phi*f(t)$$
The eigenvalue distribution function of $H$ is defined as

$$
\phi(t) = \frac{1}{n} \sum_{i=1}^{n} \delta(t - \lambda_i)
$$

Let $f_\sigma(x) = \frac{1}{\sigma \sqrt{2\pi}} \exp(-\frac{x^2}{2\sigma^2})$ be the Gaussian density.

$$
\phi(t) = \frac{1}{n} \sum_{i=1}^{n} \delta(t - \lambda_i) \quad \xrightarrow{\text{Convolution with Gaussian}} \quad \phi_\sigma(t) = \frac{1}{n} \sum_{i=1}^{n} f_\sigma(t - \lambda_i)
$$
The eigenvalue distribution function of $H$ is defined as

$$
\phi(t) = \frac{1}{n} \sum_{i=1}^{n} \delta(t - \lambda_i)
$$

Let $f_\sigma(x) = \frac{1}{\sigma \sqrt{2\pi}} \exp(-\frac{x^2}{2\sigma^2})$ be the Gaussian density.

$$
\begin{align*}
\phi(t) &= \frac{1}{n} \sum_{i=1}^{n} \delta(t - \lambda_i) \\
\phi \ast f(t) &\rightarrow \phi_\sigma(t) = \frac{1}{n} \sum_{i=1}^{n} f_\sigma(t - \lambda_i)
\end{align*}
$$
Estimating the Smoothed Density

- Gene Golub and his students [Golub and Welsch (1969); Bai et al. (1996)]

\[
\begin{align*}
\text{Constructs } (\omega_i, \ell_i) & \quad m_i = 1 \\
\text{such that for all "nice" functions } g, \\
\sum_{i=1}^{n} g(\lambda_i) & \approx \sum_{i=1}^{m} w_i g(\ell_i) \\
\text{Use } g(x) = f(\sigma(t-x)) \\
\phi(\sigma(t)) & = \frac{1}{n} \sum_{i=1}^{n} f(\sigma(t-\lambda_i)) \approx \hat{\phi}(t) = \frac{1}{m} \sum_{i=1}^{m} w_i f(\sigma(t-\ell_i))
\end{align*}
\]
Estimating the Smoothed Density

- Gene Golub and his students [Golub and Welsch (1969); Bai et al. (1996)]
- Constructs \((w_i, \ell_i)_{i=1}^m\) such that for all "nice" functions \(g\),

\[
\frac{1}{n} \sum_{i=1}^n g(\lambda_i) \approx \sum_{i=1}^m w_i g(\ell_i)
\]
Estimating the Smoothed Density

- Gene Golub and his students [Golub and Welsch (1969); Bai et al. (1996)]
- Constructs \((w_i, \ell_i)_{i=1}^m\) such that for all "nice" functions \(g\),

\[
\frac{1}{n} \sum_{i=1}^{n} g(\lambda_i) \approx \sum_{i=1}^{m} w_i g(\ell_i)
\]

- Use \(g(x) = f_\sigma(t - x)\):

\[
\phi_\sigma(t) = \frac{1}{n} \sum_{i=1}^{n} f_\sigma(t - \lambda_i) \approx \hat{\phi}(t) = \frac{1}{m} \sum_{i=1}^{m} w_i f_\sigma(t - \ell_i)
\]
Algorithm Sketch

**Stochastic**

1. Draw \( v \sim \mathcal{N}(0, \frac{1}{n} I_n) \)

**Lanczos**

1. Compute a basis for \( \{v, Hv, \ldots, H^{m-1} v\} \). Call this basis \( V \).
2. Let \( T = V^T HV \)

**Quadrature**

1. Diagonalize \( T = UDU^T \).
2. Estimate \( \phi_\sigma(t) = \frac{1}{n} \sum_{i=1}^{n} f(t - \lambda_i) \) with \( \hat{\phi}_v(t) = \sum_{i=1}^{m} U^2_{1,i} f(t - D_{i,i}) \)

**Computational Complexity**

Calculating \( (w_i, \ell_i) \) \( m_i = 1 \) takes \( O(m \times \text{model size} \times \text{dataset size}) \). In practice, \( m \approx 100 \) is more than enough. Explicitly calculating the eigenvalues takes \( O(\text{model size}^2 \times \text{dataset size}) \).
Algorithm Sketch

Stochastic  Draw $v \sim \mathcal{N}(0, \frac{1}{n}I_n)$

Lanczos  1. Compute a basis for $\{v, Hv, \ldots, H^{m-1}v\}$. Call this basis $V$.
2. Let $T = V^T HV$

Quadrature  1. Diagonalize $T = UDU^T$.
2. Estimate $\phi_\sigma(t) = \frac{1}{n} \sum_{i=1}^{n} f(t - \lambda_i)$ with $\hat{\phi}_v(t) = \sum_{i=1}^{m} U_{1,i}^2 f(t - D_{i,i})$

Computational Complexity

Calculating $(w_i, \ell_i)$ for $m = 1$ takes $O(m \times \text{model size} \times \text{dataset size})$. In practice, $m \approx 100$ is more than enough.

Explicitly calculating the eigenvalues takes $O(\text{model size}^2 \times \text{dataset size})$. 

Behrooz Ghorbani
Hessian Spectral Density
June 2019 6 / 18
Algorithm Sketch

Stochastic  Draw $\mathbf{v} \sim \mathcal{N}(0, \frac{1}{\ell} \mathbf{I}_n)$

Lanczos  
1. Compute a basis for $\{\mathbf{v}, H\mathbf{v}, \ldots, H^{m-1}\mathbf{v}\}$. Call this basis $\mathbf{V}$.
2. Let $T = \mathbf{V}^T H \mathbf{V}$

Quadrature  
1. Diagonalize $T = \mathbf{U} \mathbf{D} \mathbf{U}^T$.
2. Estimate $\phi_{\sigma}(t) = \frac{1}{n} \sum_{i=1}^{n} f(t - \lambda_i)$ with $\hat{\phi}_\mathbf{v}(t) = \sum_{i=1}^{m} U_{1,i}^2 f(t - D_{i,i})$

Computational Complexity

- Calculating $(w_i, \ell_i)_{i=1}^{m}$ takes $O(m \times \text{model size} \times \text{dataset size})$. In practice, $m \approx 100$ is more than enough.
Algorithm Sketch

**Stochastic**  
Draw $\mathbf{v} \sim \mathcal{N}(0, \frac{1}{n} I_n)$

**Lanczos**  
1. Compute a basis for $\{\mathbf{v}, H\mathbf{v}, \ldots, H^{m-1}\mathbf{v}\}$. Call this basis $V$.
2. Let $T = V^T H V$

**Quadrature**  
1. Diagonalize $T = U D U^T$.
2. Estimate $\phi_\sigma(t) = \frac{1}{n} \sum_{i=1}^{n} f(t - \lambda_i)$ with $\hat{\phi}_V(t) = \sum_{i=1}^{m} U_{1,i}^2 f(t - D_{i,i})$

**Computational Complexity**

- Calculating $(w_i, \ell_i)_{i=1}^{m}$ takes $O(m \times \text{model size} \times \text{dataset size})$. In practice, $m \approx 100$ is more than enough.
- Explicitly calculating the eigenvalues takes $O(\text{model size}^2 \times \text{dataset size})$. 
Accuracy

- The algorithm enjoys strong theoretical guarantees.
- We present some such guarantees in our paper. Ubaru et al. (2017) provide additional details.
Accuracy

- The algorithm enjoys strong theoretical guarantees.
- We present some such guarantees in our paper. Ubaru et al. (2017) provide additional details.

**Figure:** Comparison of a degree 90 quadrature approximation with the actual Hessian density. The Hessian is calculated from a 2-layer network with 15910 parameters trained on MNIST.
Let’s Train a ResNet-32

- 460K parameters.
- Trained on CIFAR-10.
- The network has Batch-Normalization (Ioffe and Szegedy (2015)).
Experiments: Initialization

- At initialization time, the Hessian has significant negative eigenvalues.
At initialization time, the Hessian has significant negative eigenvalues. This points to a significant local non-convexity of the network at time 0.
Experiments: Initialization

- At initialization time, the Hessian has significant negative eigenvalues.
- This points to a significant local non-convexity of the network at time 0.
- There is a significant difference between the initialization landscape and the training landscape.
Experiments: Initialization

- At initialization time, the Hessian has significant negative eigenvalues.
- This points to a significant local non-convexity of the network at time 0.
- There is a significant difference between the initialization landscape and the training landscape.
Experiments: Initialization

- At initialization time, the Hessian has significant negative eigenvalues.
- This points to a significant local non-convexity of the network at time 0.
- There is a significant difference between the initialization landscape and the training landscape.
- For small datasets such as CIFAR-10 / MNIST, negative directions disappear extremely fast.
Experiments: Further Training

- After the first epoch, the Hessian spectrum stabilizes.

Figure: Spectrum of the network stabilizes after the first epoch.
Experiments: Further Training

- After the first epoch, the Hessian spectrum stabilizes.

**Figure:** Spectrum of the network stabilizes after the first epoch.
Experiments: Further Training

- After the first epoch, the Hessian spectrum stabilizes.
- The Hessian contains information about non-local geometry of the loss.

Figure: Spectrum of the network stabilizes after the first epoch.
Experiments: Further Training

- After the first epoch, the Hessian spectrum stabilizes.
- The Hessian contains information about non-local geometry of the loss.
- The eigenvalues of the Hessian at this stage determine if the network can be trained effectively.

**Figure:** Spectrum of the network stabilizes after the first epoch.
Experiments: Reducing Learning Rate

- Prevalent view: smaller learning rates allow you to converge to sharper local minima.
Experiments: Reducing Learning Rate

- Prevalent view: smaller learning rates allow you to converge to sharper local minima.
Experiments: Reducing Learning Rate

- Prevalent view: smaller learning rates allow you to converge to sharper local minima.
- Reducing the learning rate should bring about an increase in the top eigenvalue.
Experiments: Reducing Learning Rate

- Prevalent view: smaller learning rates allow you to converge to sharper local minima.
- Reducing the learning rate should bring about an increase in the top eigenvalue.

**Figure:** Learning rate is reduced by a factor of 10 at step 40k. Surprisingly, the top eigenvalue also decreases.
The Hessian spectrum at the end of the training:
Experiments: End of the Training

- The Hessian spectrum at the end of the training:

Figure: Spectrum of the Hessian after 100k steps of training. The smallest eigenvalue is $\approx -0.0006$. 
Let’s remove Batch-Normalization from the network and reexamine the spectrum!
Let’s remove Batch-Normalization from the network and reexamine the spectrum!

**Figure:** Spectrum of the Hessian after 7k steps of training. Outlier eigenvalues appear when BN is removed from the network.
Experiments: Batch-Normalization

- This observation is consistent over different architectures / datasets:
Experiments: Batch-Normalization

- This observation is consistent over different architectures / datasets:

Figure: The eigenvalue comparison of the Hessian of Resnet-18 trained on ImageNet dataset. Model with BN is shown in blue and the model without BN in red. The Hessians are computed at the end of training.
Our intuition from convex optimization suggests that first-order methods slow down significantly when $\lambda_i$ are highly spread out. [see Bottou and Bousquet (2008) for explicit bounds]
Our intuition from convex optimization suggests that first-order methods slow down significantly when $\lambda_i$ are highly spread out. [see Bottou and Bousquet (2008) for explicit bounds]

Often quantities such as Condition number, $\kappa \equiv \frac{\lambda_1}{\lambda_n}$, are used to measure this spread.
Experiments: Batch-Normalization

- Our intuition from convex optimization suggests that first-order methods slow down significantly when $\lambda_i$ are highly spread out. [see Bottou and Bousquet (2008) for explicit bounds]

- Often quantities such as Condition number, $\kappa \equiv \frac{\lambda_1}{\lambda_n}$, are used to measure this spread.

Conjecture

Batch-Normalization helps optimization by removing large outlier eigenvalues.
Our intuition from convex optimization suggests that first-order methods slow down significantly when $\lambda_i$ are highly spread out. [see Bottou and Bousquet (2008) for explicit bounds]

Often quantities such as Condition number, $\kappa \equiv \frac{\lambda_1}{\lambda_n}$, are used to measure this spread.

Let’s test this assertion!

**Conjecture**

Batch-Normalization helps optimization by removing large outlier eigenvalues.
Our observations suggest that BN is effective because it removes the outlier eigenvalues of the Hessian.
Our observations suggest that BN is effective because it removes the outlier eigenvalues of the Hessian.

We predict that in scenarios where BN is not effective, outlier eigenvalues are still present.
Our observations suggest that BN is effective because it removes the outlier eigenvalues of the Hessian.

We predict that in scenarios where BN is not effective, outlier eigenvalues are still present.

Example: When statistics of the BN layer are computed from the full-dataset.
Our observations suggest that BN is effective because it removes the outlier eigenvalues of the Hessian.

We predict that in scenarios where BN is not effective, outlier eigenvalues are still present.

Example: When statistics of the BN layer are computed from the full-dataset.

Figure: Optimization progress (in terms of loss) of batch normalization with mini-batch statistics and population statistics.
BN with Population Statistics

Figure: The Hessian spectrum for a Resnet-32 after 15k steps. On the left BN is using mini-batch statistics. The network on the right is using population statistics.
Figure: The Hessian spectrum for a Resnet-32 after 15k steps. On the left BN is using mini-batch statistics. The network on the right is using population statistics.
Hope to see you at our poster session today (06:30 to 09:00 at Pacific Ballroom #51)


