A Gram-Gauss-Newton Method Learning Overparameterized Deep Neural Networks for Regression Problems

Tianle Cai

Peking University

June 21, 2019

Joint work with: Ruiqi Gao, Jikai Hou, Siyu Chen, Dong Wang, Di He, Zhihua Zhang, Liwei Wang

Outline

Modern Deep Learning

Characterized by highly **overparameterization**, i.e.,

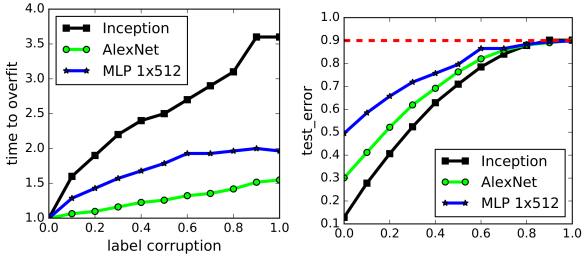
Overparameterization

The number of parameters m is much larger than the number of data n.

and highly **nonconvexity**.

Strong Expressivity under Optimization

Deep networks can be optimized to zero training error even for noisy problems [Zhang et al., 2016].



Rethinking Optimization

Why can networks find a global optima despite of nonconvexity?

Interpolation Regime

Overparameterized networks provably converge to zero training loss using gradient descent. [Du et al., 2018, Allen-Zhu et al., 2018, Zou et al., 2018]

Key Insights

- Overparameterized networks are approximately **linear** w.r.t. the parameters in a neighbor of initialization (in parameter space).
- There is a global optima inside this neighbor.

Neural Taylor Expansion

Let $f(w, x)$ denote the network with parameter $w \in \mathbb{R}^m$ and input $x \in \mathbb{R}^d$. We assume the output of $f(w, x)$ is a scalar.

Neural Taylor Expansion/ Feature Map of Neural Tangent Kernel (NTK)

Within a neighbor of the initialization w_0, $\nabla_w f(w, x) \approx \nabla_w f(w_0, x)$. Then we have the approximate neural Taylor expansion [Chizat and Bach, 2018]:

$$f(w, x) \approx \underbrace{f(w_0, x)}_{\text{bias term}} + \underbrace{(w - w_0)}_{\text{linear parameters}} \cdot \underbrace{\nabla_w f(w_0, x)}_{\text{feature of } x},$$

where $\nabla_w f(w, x)$ is the gradient of $f(w, x)$ w.r.t. w.

NTK

Neural Tangent Kernel (NTK) [Jacot et al., 2018]

Kernel induced by the feature map $\mathbb{R}^d \to \mathbb{R}^m, x \to \nabla_w f(w_0, x)$:

$$K(x, x') = \langle \nabla_w f(w_0, x), \nabla_w f(w_0, x') \rangle.$$

Remark

- Show the behavior of infinite width networks [Jacot et al., 2018].
- Exact (kernelized) linear model.
- Can be exactly solved by kernel regression. [Arora et al., 2019].

Interpolation Regime (cont.)

Key Insights

- Overparameterized networks are approximately **linear** w.r.t. the parameters in a neighbor of initialization (in parameter space).
  $\Rightarrow$ Gradient descent is applied within a "not so nonconvex" regime.
  $\Rightarrow$ Gradient descent can approximately find the optima within this regime.
- There is a global optima inside this neighbor.
  $\Rightarrow$ The optima reached by gradient descent is an (approximate) global optima :)

Question

How about other optimization methods, e.g. second-order methods?

Outline

Newton Type Methods

Let $L(w) = \sum_{i=1}^{n} \ell(f(w, x_i), y_i)$ denote the loss function on dataset $\{x_i, y_i\}_{i=1}^{n}$ with parameter w.

Newton type method

$$w_{k+1} = w_k - H(w_k)^{-1} \nabla_w L(w_k),$$

where $\nabla_w L$ is the gradient of loss function w.r.t. w and $H \in \mathbb{R}^{m \times m}$ is the (approximate) Hessian matrix.

Issues of Second-order Methods for Deep Learning

- The number of parameters m is too large. Storing Hessian matrix $H \in \mathbb{R}^{m \times m}$ and computing its inverse is intractable.
- Deep networks have complicated structure, so it is hard to get an approximation of the Hessian matrix.

Our Goal

- Utilize the properties of deep overparameterized networks.
- Make second-order methods tractable for deep learning.

Outline

Recap: Interpolate Regime

Key Insights

- Overparameterized networks are approximately **linear** in a neighbor of initialization. i.e. There is a benign neighbor.
- There is a global optima inside this neighbor. i.e. The neighbor is just enough.

Idea: Exploit the Nearly Linear Behavior within the Interpolation Regime

Regression with Square Loss

$$\min_w L(w) = \frac{1}{2}\sum_{i=1}^{n}(f(w, x_i) - y_i)^2.$$

The Hessian $H(w)$ can be calculated as:

$$\nabla_w^2 L(w) = J(w)^\top J(w) + \sum_{i=1}^{n}(f(w, x_i) - y_i)\nabla_w^2 f(w, x_i),$$

where $J(w) = (\nabla f(w, x_1), \cdots, \nabla f(w, x_n))^\top \in \mathbb{R}^{n\times m}$ denotes the Jacobian matrix.

Idea: Exploit the Nearly Linear Behavior within the Interpolation Regime (cont.)

When $f(w, x)$ is nearly linear w.r.t. w, i.e. $\nabla_w^2 f(w, x) \approx 0$, we have

$$\begin{aligned} H(w) =& J(w)^\top J(w) + \sum_{i=1}^{n} (f(w, x_i) - y_i) \nabla_w^2 f(w, x_i) \\ \approx& J(w)^\top J(w), \end{aligned}$$

which suggests $J(w)^\top J(w) \in \mathbb{R}^{m \times m}$ is a good approximation of Hessian.

Remark

- Same idea as Gauss-Newton method.
- Good convergence rate for mildly nonlinear f. [Golub, 1965]

Idea: Exploit the Nearly Linear Behavior within the Interpolation Regime (cont.)

Update Rule of Classic Gauss-Newton Method

$$
\begin{aligned}
w \to & w - \Big(\underbrace{J(w)^\top J(w)}_{\text{approximate Hessian}} \Big)^{-1} \nabla_w L(w) \\
= & w - \Big(\underbrace{J(w)^\top J(w)}_{\text{approximate Hessian}} \Big)^{-1} J(w)^\top (f(w) - y),
\end{aligned}
$$

where we denote $f(w) = (f(w, x_1, \cdots, f(w, x_n))^\top, y = (y_1, \cdots, y_n)^\top$ for convenience. The equation comes from $\nabla_w L(w) = J(w)^\top (f(w) - y)$.

Problems of Approximate Hessian

- The approximate Hessian $J(w)^\top J(w) \in \mathbb{R}^{m\times m}$ is still too large to be stored and inverted.
- For overparameterized model, $J(w) \in \mathbb{R}^{n\times m}$ has rank no more than the number of data $n(\ll m)$. So $J(w)^\top J(w)$ is not invertible.

Recap: Neural Taylor Expansian and NTK

Neural Taylor Expansion and NTK

$$f(w,x) \approx \underbrace{f(w_0,x)}_{\text{bias term}} + \underbrace{(w-w_0)}_{\text{linear parameters}} \cdot \underbrace{\nabla_w f(w_0,x)}_{\text{feature of } x},$$

$$K(x,x') = \langle \nabla_w f(w_0,x), \nabla_w f(w_0,x') \rangle.$$

Remark

- The linear expansion approximation used in NTK is same as Gauss-Newton method.
- The solution of kernel regression is given by

$$w = w_0 - J(w_0)^\top G(w_0)^{-1}(f(w_0) - y),$$

where $G(w_0)_{ij} = \langle \nabla_w f(w_0,x_i), \nabla_w f(w_0,x_j) \rangle$ is the Gram matrix of NTK.

Idea: From Hessian to Gram Matrix

Gram Matrix

$$G(w) = J(w)J(w)^\top \in \mathbb{R}^{n\times n},$$

where n is the number of data.

Relation between Gram and Approximate Hessian

$$J(w)^\top (\underbrace{J(w)J(w)^\top}_{\text{Gram}})^{-1} = (\ \underbrace{J(w)^\top J(w)}_{\text{approximate Hessian}}\ )^\dagger J(w)^\top,$$

where $(\cdot)^\dagger$ is the pseudo-inverse.

Gram-Gauss-Newton Method (GGN)

Update Rule of Gram-Gauss-Newton Method

$$w \to w - J(w)^\top (\underbrace{J(w)J(w)^\top}_{\text{Gram}})^{-1}(f(w) - y).$$

Remark

- Can be viewed as solving NTK kernel regression based on current parameters.
- Gram matrix is $n \times n$ where $n \ll m$.
- Combine with the **mini-batch scheme**, we can only compute the Gram matrix of each batch.

Second-order Convergence for Overparameterized Networks

Theorem

For two-layer ReLU network, if width $M = \Omega\left(\text{poly}\left(\frac{n}{\lambda_0}\right)\right)$, then with high probability over the random initialization, the full-batch GGN satisfies:

1. The Gram matrix G at each iteration is invertible;
2. The loss converges to zero in a way that

$$\|f(w_{t+1}) - y\|_2 \leq \frac{C}{\sqrt{M}}\|f(w_t) - y\|_2^2$$

for some constant C that is independent of M.

Computational Cost

Scale of Key Items

Let m denote the number of parameters, B denote the batch size (e.g. 128).

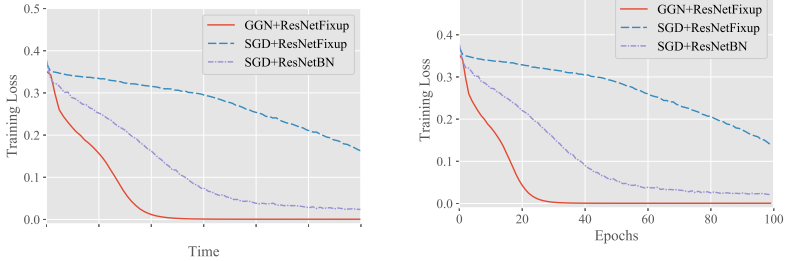
- $J(w) \in \mathbb{R}^{B \times m}$.
- $G(w) = J(w)J(w)^\top \in \mathbb{R}^{B \times B}$.

Main Overhead

- Computation of Gram matrix.
- Computation of the inverse of Gram matrix.

Experimental Results I

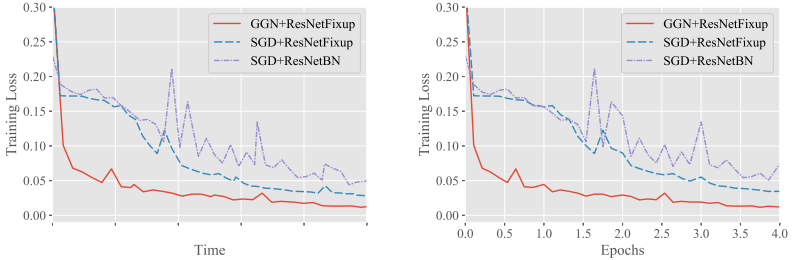
We conduct experiments on two regression datasets, AFAD-LITE task (human age prediction by image) and RSNA Bone Age regression.



(a) Loss-time curve on AFAD-LITE (b) Loss-epoch curve on AFAD-LITE

Figure: Training curves of GGN and SGD on two regression tasks.

Experimental Results II



(a) Loss-time curve on RSNA Bone Age

(b) Loss-epoch curve on RSNA Bone Age

Figure: Training curves of GGN and SGD on two regression tasks.

Take away

- Get along with overparameterization: Data-related term (e.g. Gram) is cheaper than parameter-related term (e.g. Hessian).
- Enjoy overparameterization: Utilize the local property of overparameterized networks and achieve higher convergence rate.

Q&A

Thank you!

See our paper on `https://arxiv.org/abs/1905.11675`.

And slides on `http://suo.im/5fsOXj`.

References I

Allen-Zhu, Z., Li, Y., and Song, Z. (2018).
On the convergence rate of training recurrent neural networks.
arXiv preprint arXiv:1810.12065.

Arora, S., Du, S. S., Hu, W., Li, Z., Salakhutdinov, R., and Wang, R. (2019).
On exact computation with an infinitely wide neural net.
arXiv preprint arXiv:1904.11955.

Chizat, L. and Bach, F. (2018).
A note on lazy training in supervised differentiable programming.
arXiv preprint arXiv:1812.07956.

Du, S. S., Zhai, X., Poczos, B., and Singh, A. (2018).
Gradient descent provably optimizes over-parameterized neural networks.
arXiv preprint arXiv:1810.02054.

References II


Golub, G. (1965).
Numerical methods for solving linear least squares problems.
Numerische Mathematik, 7(3):206–216.


Jacot, A., Gabriel, F., and Hongler, C. (2018).
Neural tangent kernel: Convergence and generalization in neural networks.
In *Advances in neural information processing systems*, pages 8571–8580.


Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. (2016).
Understanding deep learning requires rethinking generalization.
arXiv preprint arXiv:1611.03530.


Zou, D., Cao, Y., Zhou, D., and Gu, Q. (2018).
Stochastic gradient descent optimizes over-parameterized deep relu networks.
arXiv preprint arXiv:1811.08888.