ECE826 Lecture 10:

Guarantees for fitting NNs with GD

Contents

- Going deeper on PL
- No bad local minima
- It's all about the Jacobian
- Brief overview of NTK/large overparameterization

Minimizing the Empirical Risk

- The empirical cost function that we have access to

$$\min_{h \in \mathscr{H}} \left(R_S[h] = \frac{1}{n} \sum_{i=1}^{n} \ell(h(x_i); y_i) \right)$$

- Question: Can we approximate the solution to this minimization? If so how fast?

- The answer must depend on:
    1) n, the sample size
    2) $\mathscr{H}$, the hypothesis class and loss function
    3) $\mathscr{D}$, the data distribution
    4) the optimization algorithm that outputs our classifier

Last time: PL is good

SGD/GD on general non convex functions?

Theorem

Let $f(w)$ be a β-smooth function with L-bounded stoch. gradients (i.e., $\mathbb{E}_i\|\nabla f_i(w)\| \leq L$). Then, the gradients of SGD with step-size $\gamma = \frac{R}{\beta L^2 T}$ satisfy

$$\min_{k\in[T]} \mathbb{E}\|\nabla f(w_k)\|^2 \leq 2\sqrt{\frac{R\beta L^2}{T}}$$

Proof:

$$f(w_{k+1}) - f(w_k) - \langle \nabla f(w_k), w_{k+1} - w_k\rangle \leq \frac{\beta}{2}\|w_k - w_{k+1}\|^2$$

$$\implies \mathbb{E}f(w_{k+1}) - \mathbb{E}f(w_k) + \gamma\langle \nabla f(w_k), \nabla f_{s_k}(w_k)\rangle \leq \frac{\beta}{2}\|\gamma \nabla f_{s_k}(w_k)\|^2$$

$$\implies \mathbb{E}\|\nabla f(w_k)\|^2 \leq \frac{\mathbb{E}f(w_{k+1}) - \mathbb{E}f(w_k)}{\gamma} + \frac{\gamma L^2 \beta}{2}$$

$$\implies \min_k \mathbb{E}\|\nabla f(w_k)\|^2 \leq \frac{1}{T}\sum_{k=1}^{T} \mathbb{E}\|\nabla f(w_k)\|^2 \leq \frac{\mathbb{E}f(w_1) - f(w_0)}{\gamma T} + \frac{\gamma L^2 \beta}{2T}$$

This is a very slow rate, that is very conservative. Makes sense!

It also doesn't tell us anything about the quality of the solution that SGD finds

GD on Polyak-Łojasiewicz functions

Theorem

Let $f(w)$ be a β-smooth, μ-PL function (i.e., $\left\| \nabla_w L(w) \right\|^2 \geq \mu(L(w) - L^*)$.

Then, GD with with step-size $\gamma = \frac{1}{L}$ satisfies

$$f(w_k) - f^* \leq \left(1 - \frac{\mu}{\beta}\right)^k \left(f(w_0) - f^*\right)$$

Proof:

$$f(w_{k+1}) - f(w_k) \leq \langle \nabla f(w_k), w_{k+1} - w_k \rangle + \frac{\beta}{2}\|w_k - w_{k+1}\|^2$$

$$\leq -\gamma\|\nabla f(w_k)\|^2 + \frac{\beta}{2\beta^2}\|\nabla f(w_k)\|^2$$

$$\leq -\frac{1}{\beta}\|\nabla f(w_k)\|^2 \leq -\frac{\mu}{\beta}(f(w_k) - f^*)$$

much faster rate, when is PL satisfied?

$$\implies f(w_{k+1}) - f(w_k) - f^* \leq -\frac{1}{\beta}\|\nabla f(w_k)\|^2 \leq -\frac{\mu}{\beta}(f(w_k) - f^*) - f^*$$

$$f(w_{k+1}) - f^* \leq -\frac{\mu}{\beta}(f(w_k) - f^*) - (f^* - f(w_k))$$

$$f(w_{k+1}) - f^* \leq \left(1 - \frac{\mu}{\beta}\right)(f(w_k) - f^*)$$

Loss landscapes and optimization in over-parameterized non-linear systems and neural networks

Chaoyue Liu[a], Libin Zhu[b,c], and Mikhail Belkin[c]

[a]*Department of Computer Science and Engineering, The Ohio State University*
[b]*Department of Computer Science and Engineering, University of California, San Diego*
[c]*Halıcıoğlu Data Science Institute, University of California, San Diego*

May 28, 2021

Overparameterized Nonlinear Learning: Gradient Descent Takes the Shortest Path?

Samet Oymak* and Mahdi Soltanolkotabi†

A Convergence Theory for Deep Learning via Over-Parameterization

Zeyuan Allen-Zhu
zeyuan@csail.mit.edu

Yuanzhi Li
yuanzhil@stanford.edu

Zhao Song
zhaos@utexas.e
UT-Austin
ity of Washington

On the Convergence Rate of Training Recurrent Neural Networks

Zeyuan Allen-Zhu
zeyuan@csail.mit.edu
Microsoft Research AI

Yuanzhi Li
yuanzhil@stanford.edu
Stanford University
Princeton University

Zhao Song
zhaos@utexas.edu
UT-Austin
University of Washington
Harvard University

October 28, 2018

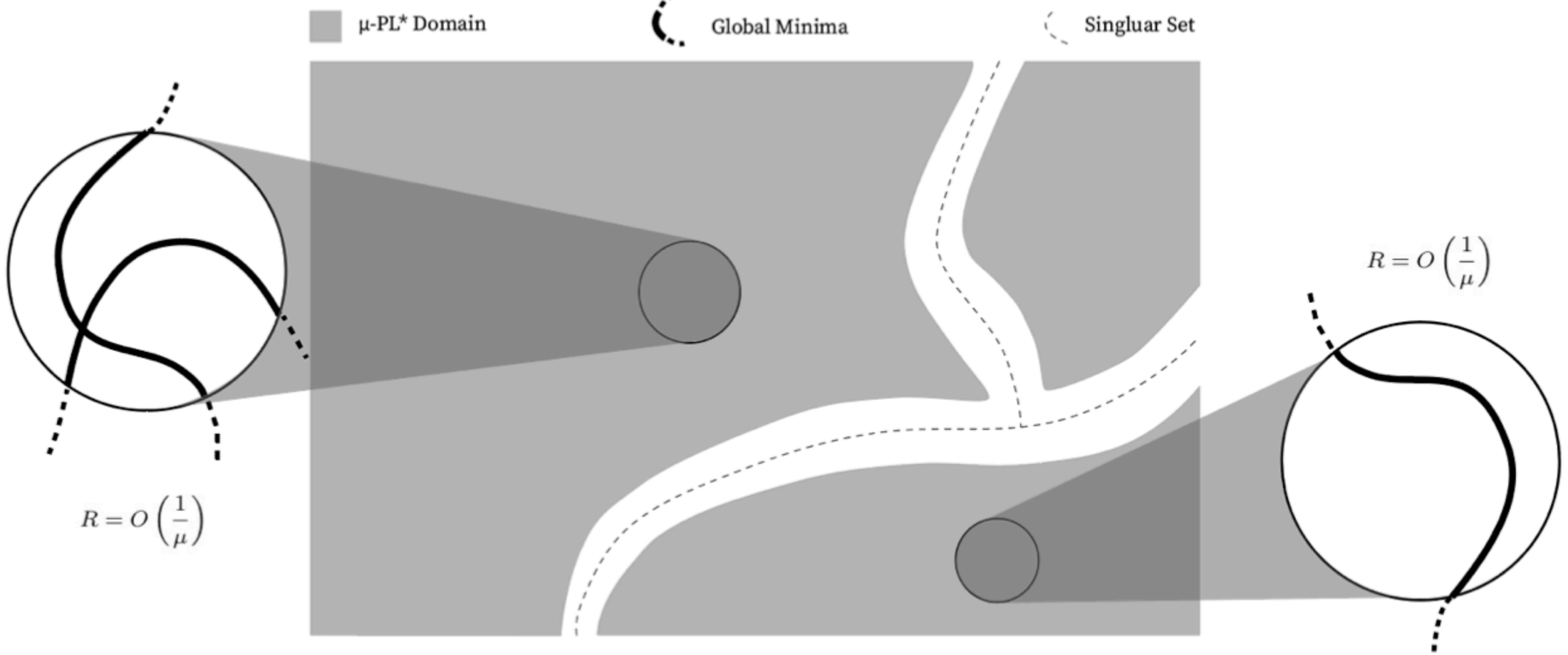
No bad local minima: Data independent training error guarantees for multilayer neural networks

Daniel Soudry
Department of Statistics
Columbia University
New York, NY 10027, USA
daniel.soudry@gmail.com

Yair Carmon
Department of Electrical Engineering
Stanford University
Stanford, CA 94305, USA
yairc@stanford.edu

Gradient Descent Finds Global Minima of Deep Neural Networks

Simon S. Du [*1] **Jason D. Lee** [*2] **Haochuan Li** [*3 4] **Liwei Wang** [*5 4] **Xiyu Zhai** [*6]



via Over-Parameterization

Zeyuan Allen-Zhu
zeyuan@csail.mit.edu
Microsoft Research AI

Yuanzhi Li
yuanzhil@stanford.edu
Stanford University
Princeton University

Zhao Song
zhaos@utexas.edu
UT Austin
University of Washington
Harvard University

PL-like conditions hold in neighborhoods around initialization/optima.

PL in Nonlinear Least Squares

Nonlinear least squares

- Let's say we are trying to solve $\min_w \|h(X;w) - y\|^2$ with GD.
- The gradient of the loss is equal to $\nabla_w \|h(X;w) - y\|^2 = [\nabla_w h(X;w)](h(X;w) - y)$

Nonlinear least squares

- Let's say we are trying to solve $\min_w \|h(X;w) - y\|^2$ with GD.
- The gradient of the loss is equal to $\nabla_w \|h(X;w) - y\|^2 = [\nabla_w h(X;w)](h(X;w) - y)$
- Let us refer to $J(w) = \nabla_w h(X;w) \in \mathbb{R}^{d \times n}$ as the Jacobian of the predictions
- Note that again

$$\left\| \nabla_w L(w) \right\|^2 = \left\| J(w)(h(X;w) - y) \right\|^2 \geq 4\lambda_{\min}(J(w)^T J(w)) \left\| h(X;w) - y \right\|^2$$

Nonlinear least squares

- Let's say we are trying to solve $\min_w \|h(X;w) - y\|^2$ with GD.
- The gradient of the loss is equal to $\nabla_w \|h(X;w) - y\|^2 = [\nabla_w h(X;w)](h(X;w) - y)$
- Let us refer to $J(w) = \nabla_w h(X;w) \in \mathbb{R}^{d \times n}$ as the Jacobian of the predictions
- Note that again

$$\left\| \nabla_w L(w) \right\|^2 = \left\| J(w)(h(X;w) - y) \right\|^2 \geq 4\lambda_{\min}(J(w)^T J(w)) \left\| h(X;w) - y \right\|^2$$

- Ha! that is the again PL condition (assuming $L^* = 0$) with $\mu = 4\lambda_{\min}(X^T X)$

Nonlinear least squares

- Let's say we are trying to solve $\min_w \|h(X;w) - y\|^2$ with GD.
- The gradient of the loss is equal to $\nabla_w \|h(X;w) - y\|^2 = [\nabla_w h(X;w)](h(X;w) - y)$
- Let us refer to $J(w) = \nabla_w h(X;w) \in \mathbb{R}^{d \times n}$ as the Jacobian of the predictions
- Note that again

$$\left\| \nabla_w L(w) \right\|^2 = \left\| J(w)(h(X;w) - y) \right\|^2 \geq 4\lambda_{\min}(J(w)^T J(w)) \left\| h(X;w) - y \right\|^2$$

- Ha! that is the again PL condition (assuming $L^* = 0$) with $\mu = 4\lambda_{\min}(X^T X)$

Lemma:

Non-linear least squares where $\min_{w \in \mathscr{W}} \text{rank}(J(w(w)) = n$ are PL in $\mathscr{W}$

Nonlinear least squares

- Let's say we are trying to solve $\min_w \|h(X;w) - y\|^2$ with GD.
- The gradient of the loss is equal to $\nabla_w \|h(X;w) - y\|^2 = [\nabla_w h(X;w)](h(X;w) - y)$
- Let us refer to $J(w) = \nabla_w h(X;w) \in \mathbb{R}^{d \times n}$ as the Jacobian of the predictions
- Note that again

$$\left\| \nabla_w L(w) \right\|^2 = \left\| J(w)(h(X;w) - y) \right\|^2 \geq 4\lambda_{\min}(J(w)^T J(w)) \left\| h(X;w) - y \right\|^2$$

- Ha! that is the again PL condition (assuming $L^* = 0$) with $\mu = 4\lambda_{\min}(X^T X)$

full-rank Jacobian ⇒ all local minima are global

Lemma:

Non-linear least squares where $\min_{w \in \mathcal{W}} \text{rank}(J(w(w)) = n$ are PL in $\mathcal{W}$

Nonlinear least squares

- Let's say we are trying to solve $\min_w \|h(X;w) - y\|^2$ with GD.
- The gradient of the loss is equal to $\nabla_w \|h(X;w) - y\|^2 = [\nabla_w h(X;w)](h(X;w) - y)$
- Let us refer to $J(w) = \nabla_w h(X;w) \in \mathbb{R}^{d \times n}$ as the Jacobian of the predictions
- Note that again

$$\left\| \nabla_w L(w) \right\|^2 = \left\| J(w)(h(X;w) - y) \right\|^2 \geq 4\lambda_{\min}(J(w)^T J(w)) \left\| h(X;w) - y \right\|^2$$

- Ha! that is the again PL condition (assuming $L^* = 0$) with $\mu = 4\lambda_{\min}(X^T X)$

full-rank Jacobian ⇒ all local minima are global

Lemma:

Non-linear least squares where $\min_{w \in \mathcal{W}} \text{rank}(J(w(w)) = n$ are PL in $\mathcal{W}$

Most modern DL theory work tries to show that the above is indeed the case

PL in Neural Networks?

No bad local minima almost always?

No bad local minima: Data independent training error guarantees for multilayer neural networks

Daniel Soudry
Department of Statistics
Columbia University
New York, NY 10027, USA
daniel.soudry@gmail.com

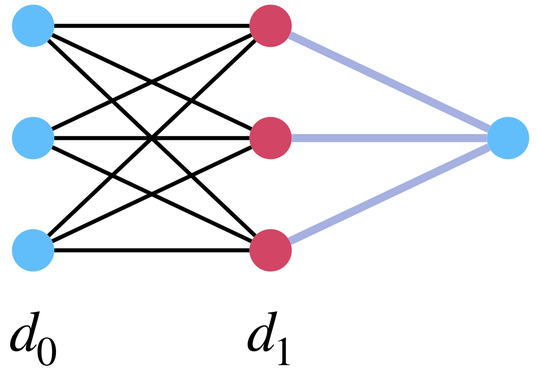
Yair Carmon
Department of Electrical Engineering
Stanford University
Stanford, CA 94305, USA
yairc@stanford.edu

No bad local minima almost always?

No bad local minima: Data independent training error guarantees for multilayer neural networks

Daniel Soudry
Department of Statistics
Columbia University
New York, NY 10027, USA
daniel.soudry@gmail.com

Yair Carmon
Department of Electrical Engineering
Stanford University
Stanford, CA 94305, USA
yairc@stanford.edu



Theorem (informal):

For 1-hidden layer nets $\text{rank}(J(w)) = n$ almost surely for leaky-ReLU networks, if $d_0 \cdot d_1 \geq n$

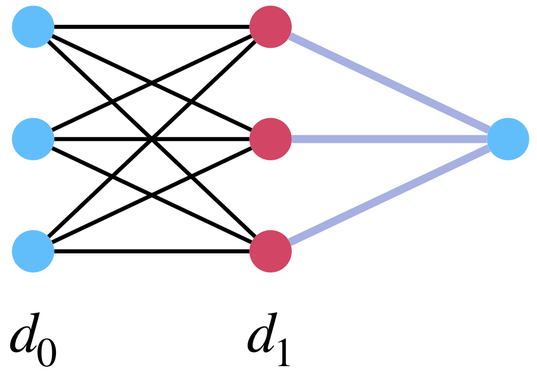
For L-hidden layer nets $\text{rank}(J(w)) = n$ a.s., if params. of last layer $\geq n$

No bad local minima almost always?

No bad local minima: Data independent training error guarantees for multilayer neural networks

Daniel Soudry
Department of Statistics
Columbia University
New York, NY 10027, USA
daniel.soudry@gmail.com

Yair Carmon
Department of Electrical Engineering
Stanford University
Stanford, CA 94305, USA
yairc@stanford.edu



Theorem (informal):
For 1-hidden layer nets $\text{rank}(J(w)) = n$ almost surely for leaky-ReLU networks, if $d_0 \cdot d_1 \geq n$
For L-hidden layer nets $\text{rank}(J(w)) = n$ a.s., if params. of last layer $\geq n$

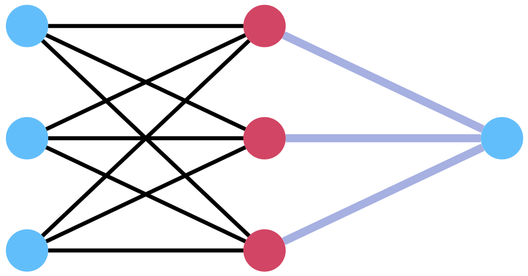
Most modern DL theory work tries to show that the above is indeed the case

More examples

1-layer linear Neural Networks

- Let us assume we have a 1-layer linear network.
- The prediction of this network is given as $h(W; x) = \langle v, Wx \rangle$

$\sigma(x) = x$



assume output edges are all fixed

reminder:

$$\| \nabla_w L(w) \|^2 \geq 4\lambda_{\min}(J(w)^T J(w)) \| h(X; w) - y \|$$

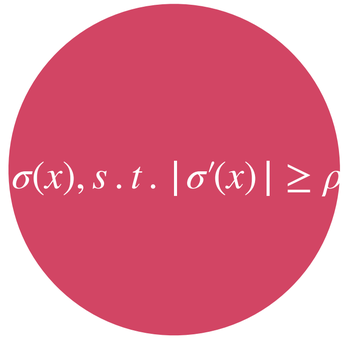
- The Jacobian is equal to $J(w) = \nabla_w h(W, x) = \begin{bmatrix} v_1 x_1 & v_1 x_2 & \dots & v_1 x_n \\ \vdots & \vdots & \dots & \vdots \\ v_k x_1 & v_k x_2 & \dots & v_k x_n \end{bmatrix} \in \mathbb{R}^{kd \times n}$

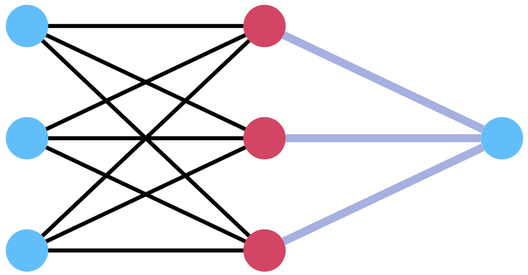
- Note that $J(w) = v \otimes X$ and we know that $\text{rank}(J(w)) = \text{rank}(v) \cdot \text{rank}(X) = \text{rank}(X)$

- Hence, if the matrix of data points is full rank n, then the cost function is PL.

1-layer Neural Networks, general activation

- Let us assume we have a 1-layer linear network
- The prediction of this network is given as $h(W; x) = \langle v, \sigma(Wx) \rangle$



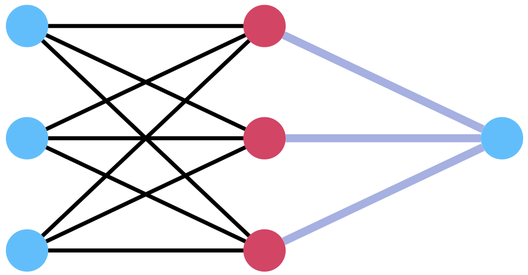


assume output edges are all fixed

1-layer Neural Networks, general activation

- Let us assume we have a 1-layer linear network
- The prediction of this network is given as $h(W;x) = \langle v, \sigma(Wx) \rangle$

$\sigma(x), s.t. \; |\sigma'(x)| \geq \rho$



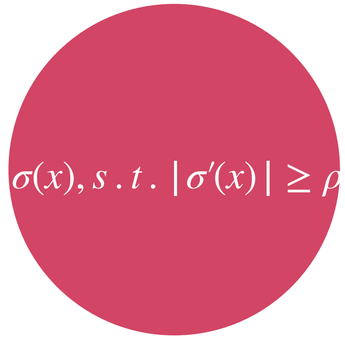
assume output edges are all fixed

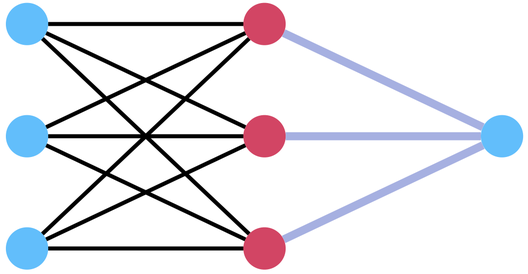
reminder:

$$\| \nabla_w L(w) \|^2 \geq 4\lambda_{\min}(J(w)^T J(w)) \| h(X;w) - y \|$$

1-layer Neural Networks, general activation

- Let us assume we have a 1-layer linear network
- The prediction of this network is given as $h(W;x) = \langle v, \sigma(Wx) \rangle$





assume output edges are all fixed

reminder:

$$\| \nabla_w L(w) \|^2 \geq 4\lambda_{\min}(J(w)^T J(w)) \| h(X;w) - y \|$$

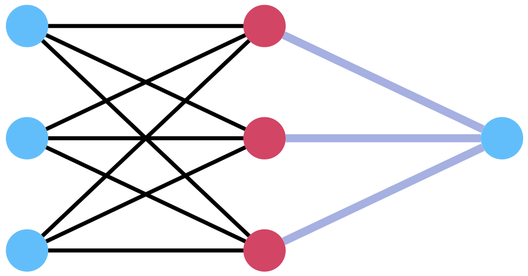
The Jacobian is equal to $J(w) =$

$$\begin{bmatrix} v_1\sigma'(\langle w_1, x_1 \rangle) \cdot x_1 & v_2\sigma'(\langle w_1, x_2 \rangle) \cdot x_2 & \dots & v_k\sigma'(\langle w_1, x_n \rangle) \cdot x_n \\ \vdots & \vdots & \dots & \vdots \\ v_1\sigma'(\langle w_k, x_1 \rangle) \cdot x_1 & v_2\sigma'(\langle w_k, x_2 \rangle) \cdot x_2 & \dots & v_k\sigma'(\langle w_k, x_n \rangle) \cdot x_n \end{bmatrix}$$

1-layer Neural Networks, general activation

- Let us assume we have a 1-layer linear network
- The prediction of this network is given as $h(W;x) = \langle v, \sigma(Wx) \rangle$

$\sigma(x), s.t. \, |\sigma'(x)| \geq \rho$



assume output edges are all fixed

reminder:

$$\| \nabla_w L(w) \|^2 \geq 4\lambda_{\min}(J(w)^T J(w)) \| h(X;w) - y \|$$

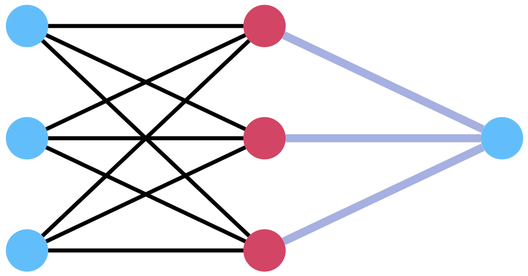
The Jacobian is equal to $J(w) = \begin{bmatrix} v_1\sigma'(\langle w_1, x_1 \rangle) \cdot x_1 & v_2\sigma'(\langle w_1, x_2 \rangle) \cdot x_2 & \dots & v_k\sigma'(\langle w_1, x_n \rangle) \cdot x_n \\ \vdots & \vdots & \dots & \vdots \\ v_1\sigma'(\langle w_k, x_1 \rangle) \cdot x_1 & v_2\sigma'(\langle w_k, x_2 \rangle) \cdot x_2 & \dots & v_k\sigma'(\langle w_k, x_n \rangle) \cdot x_n \end{bmatrix}$

Note $\text{rank}(J(w)) = n$ if the rank of the data matrix is n.

1-layer Neural Networks, general activation

- Let us assume we have a 1-layer linear network
- The prediction of this network is given as $h(W;x) = \langle v, \sigma(Wx) \rangle$

$\sigma(x), s.t. \, |\sigma'(x)| \geq \rho$



assume output edges are all fixed

reminder:

$$\| \nabla_w L(w) \|^2 \geq 4\lambda_{\min}(J(w)^T J(w)) \| h(X;w) - y \|$$

The Jacobian is equal to $J(w) = \begin{bmatrix} v_1\sigma'(\langle w_1, x_1 \rangle) \cdot x_1 & v_2\sigma'(\langle w_1, x_2 \rangle) \cdot x_2 & \dots & v_k\sigma'(\langle w_1, x_n \rangle) \cdot x_n \\ \vdots & \vdots & \dots & \vdots \\ v_1\sigma'(\langle w_k, x_1 \rangle) \cdot x_1 & v_2\sigma'(\langle w_k, x_2 \rangle) \cdot x_2 & \dots & v_k\sigma'(\langle w_k, x_n \rangle) \cdot x_n \end{bmatrix}$

Note $\text{rank}(J(w)) = n$ if the rank of the data matrix is n.

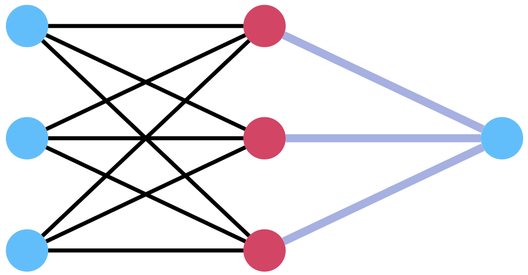
this would again imply that all local minima = global, but speed of convergence open

What if we only trained the last layer?

1-layer Neural Networks

- Let us assume we have a 1-layer linear network

$$\sigma(x) = \mathbf{1}_{x \geq 0}$$

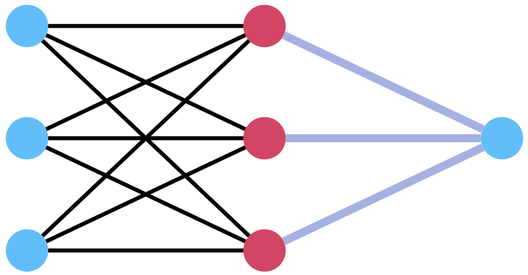


Assume input layer is random

1-layer Neural Networks

- Let us assume we have a 1-layer linear network

$\sigma(x) = \mathbf{1}_{x \geq 0}$



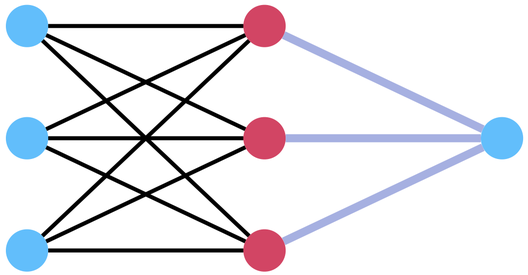
Assume input layer is random

- Note that $\min_v \sum_i \left(\langle v, \sigma(Wx_i)\rangle - y_i\right)^2$ is equivalent to $\min_v \|Qv - y\|^2$ where $Q = \left[\sigma(Wx_1) \ \ldots \ \sigma(Wx_n)\right]$

1-layer Neural Networks

- Let us assume we have a 1-layer linear network

$\sigma(x) = \mathbf{1}_{x \geq 0}$



Assume input layer is random

- Note that $\min_v \sum_i \left(\langle v, \sigma(Wx_i)\rangle - y_i\right)^2$ is equivalent to $\min_v \|Qv - y\|^2$ where $Q = \left[\sigma(Wx_1) \ \ldots \ \sigma(Wx_n)\right]$
- Training the last layer is a CONVEX problem! Can always fit perfectly as long as Q is full rank

Is Q full rank?

- Reminder: $Q = \left[\sigma(Wx_1) \ \ldots \ \sigma(Wx_n)\right]$. Note that $\text{rank}(Q) = \text{rank}(QQ^T)$ and that

$$[QQ^T]_{i,j} = \langle \sigma(Wx_i), \sigma(Wx_j) \rangle = \sum_{l=1}^{k} \sigma(w_l x_i) \cdot \sigma(w_l x_j)$$

- What would happen if we took the number of parameters to infinity?

$$\frac{1}{k} \sum_{l=1}^{k} \sigma(w_l x_i) \cdot \sigma(w_l x_j) \longrightarrow E_w[\sigma(wx_i) \cdot \sigma(wx_j)] = K^{\infty}_{i,j}$$

Theorem (Du et al):

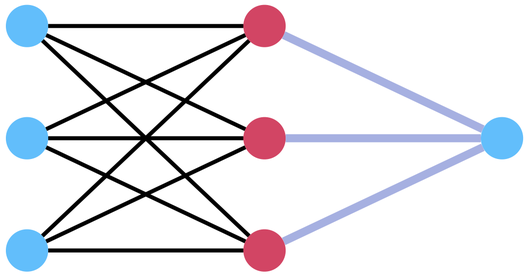
For ReLu networks K^{∞} is full-rank as long as two input points are not parallel.

Moreover, K^{p} is very close to K^{∞} (hence similar spectral properties) for polynomial overparameterization.

1-layer Neural Networks

- Let us assume we have a 1-layer linear network

$\sigma(x) = \mathbf{1}_{x \geq 0}$



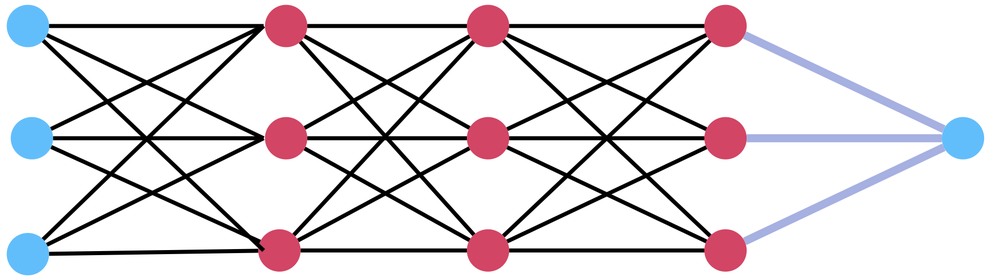
Assume input layer is random

- Note that $\min_v \sum_i \left(\langle v, \sigma(Wx_i)\rangle - y_i\right)^2$ is equivalent to $\min_v \|Qv - y\|^2$ where $Q = \left[\sigma(Wx_1) \ \ldots \ \sigma(Wx_n)\right]$
- Training the last layer is a CONVEX problem! Can always fit perfectly as long as Q is full rank (true for large enough overparameterization and full rank inputs)

Training the last layer, not very interesting?
Seems that it doesn't capture much of the non-convex nature of DL?

L-layer Neural Networks

- Let us assume we have a 1-layer linear network



$\sigma(x) = \mathbf{1}_{x \geq 0}$

Assume all but last layers are random

- Note that $\min_v \sum_i \left(\langle v, h(W;x)\rangle - y_i\right)^2$ is equivalent to $\min_v \|Qv - y\|^2$ where $Q = \left[h(w;x_1) \ \dots \ h(w;x_n)\right]$
- Training the last layer is a CONVEX problem no matter how deep the net is!)

Training the last layer, not very interesting?
Seems that it doesn't capture much of the non-convex nature of DL?

Back to training all the layers

Lazy training for large overparameterization

- Many recent works show that if the overparameterization is VERY large, then

$$h(w; x) \approx h(w_0; x) + \langle w - w_0, \nabla_w h(w_t; x) \rangle$$

Lazy training for large overparameterization

- Many recent works show that if the overparameterization is VERY large, then

$$h(w; x) \approx h(w_0; x) + \langle w - w_0, \nabla_w h(w_t; x) \rangle$$

- Specifically, the network behaves like a linear classifier and the residual error follows the following dynamics

$$y - h(w_k; X) = (I_n - \gamma G(k))(y - h(w_k; X))$$

Lazy training for large overparameterization

- Many recent works show that if the overparameterization is VERY large, then

$$h(w; x) \approx h(w_0; x) + \langle w - w_0, \nabla_w h(w_t; x) \rangle$$

- Specifically, the network behaves like a linear classifier and the residual error follows the following dynamics

$$y - h(w_k; X) = (I_n - \gamma G(k))(y - h(w_k; X))$$

- where $G(k) = J(w_k)^T J(w_k)$

Lazy training for large overparameterization

- Many recent works show that if the overparameterization is VERY large, then

$$h(w; x) \approx h(w_0; x) + \langle w - w_0, \nabla_w h(w_t; x) \rangle$$

- Specifically, the network behaves like a linear classifier and the residual error follows the following dynamics

$$y - h(w_k; X) = (I_n - \gamma G(k))(y - h(w_k; X))$$

- where $G(k) = J(w_k)^T J(w_k)$

- When the overparameterization is large then $\|G(k) - G(0)\| < \epsilon$.

Lazy training for large overparameterization

- Many recent works show that if the overparameterization is VERY large, then

$$h(w; x) \approx h(w_0; x) + \langle w - w_0, \nabla_w h(w_t; x) \rangle$$

- Specifically, the network behaves like a linear classifier and the residual error follows the following dynamics

$$y - h(w_k; X) = (I_n - \gamma G(k))(y - h(w_k; X))$$

- where $G(k) = J(w_k)^T J(w_k)$

- When the overparameterization is large then $\|G(k) - G(0)\| < \epsilon$.

- Moreover $G(0)$ is shown to have a non-zero minimum eigenvalue, so that

$$\|y - h(w_k; X)\| \lessapprox (1 - \gamma\lambda_n)^k \|y - h(w_k; X)\|$$

Lazy training for large overparameterization

Theorem (Du et al):

1) For width = O(poly(n)) then the loss function becomes PL around the initial set of weights.
2) GD doesn't move far from init even if run for ever.
3) OPT is close to int
4) GD converges in poly-time to OPT.

For width = O(poly(n)) then the NNs+squared loss = PL close to init

Current theoretical SOTA

Subquadratic Overparameterization for Shallow Neural Networks

Chaehwan Song[1*] **Ali Ramezani-Kebrya**[1*]

Thomas Pethick[1] **Armin Eftekhari**[2†] **Volkan Cevher**[1]

something odd..

Table 1: Scaling with the number of training data in the overparameterization regime. QL=quadratic loss, CLL=convex and Lipschitz loss, SD=separable data.

Depth	Algorithm	Setting	Activation	Scaling	Reference
2	GD on layer 1	QL	ReLU	$\tilde{\Omega}(n^2)$	Oymak and Soltanolkotabi [38]
L	GD on layer L	CLL	ReLU	$\tilde{\Omega}(n)$	Kawaguchi and Huang [21]
2	GD	SD	ReLU	$\tilde{\Omega}(n^2)$	Song and Yang [39]
2	GD	SD and QL	ReLU	$\tilde{\Omega}(n^6)$	Du et al. [12]
L	GD	SD and QL	ReLU	$\Omega(n^8 L^{12})$	Zou and Gu [44]
2	GD	QL	Smooth	$\tilde{\Omega}(n^{\frac{3}{2}})$	**This paper**

A curious observation on fitting the data

Small ReLU networks are powerful memorizers: a tight analysis of memorization capacity

Chulhee Yun
MIT
Cambridge, MA 02139
chulheey@mit.edu

Suvrit Sra
MIT
Cambridge, MA 02139
suvrit@mit.edu

Ali Jadbabaie
MIT
Cambridge, MA 02139
jadbabai@mit.edu

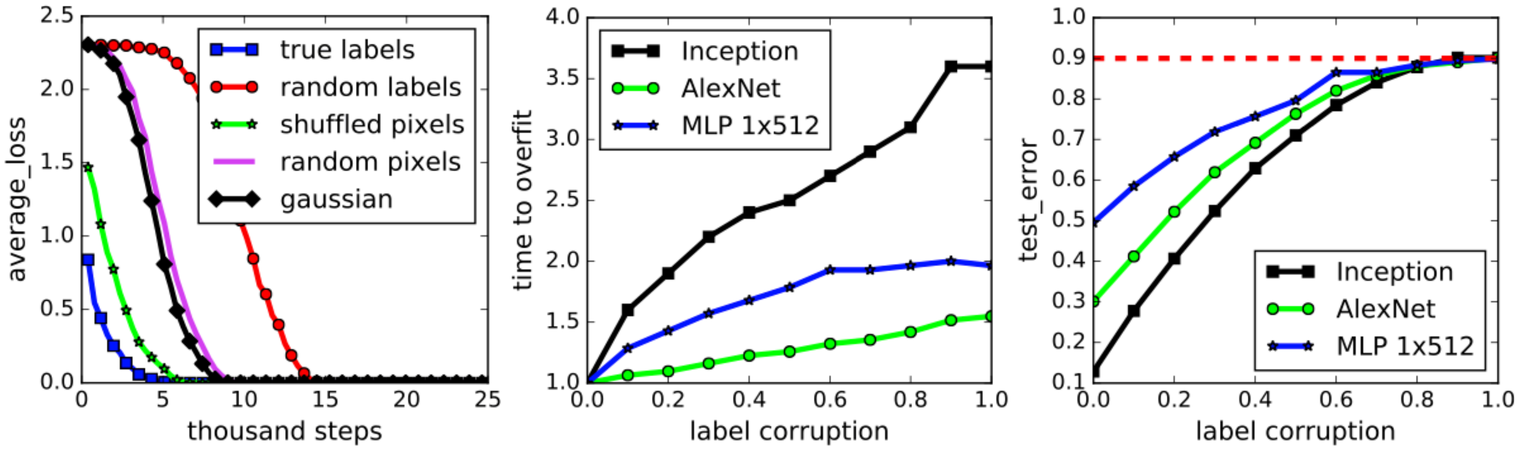
Theorem:

Any data set of size n can be memorized by a 3-layer ReLU neural network with $O(n)$ weights.

These constructions can be made in linear time. Yet SGD on the same arch needs so much more larger overarm. Why??

But somehow SGD does more than just that..

Rethinking Generalization [Zhang et al. ICLR17]



(a) learning curves (b) convergence slowdown (c) generalization error growth

Figure 1: Fitting random labels and random pixels on CIFAR10. (a) shows the training loss of various experiment settings decaying with the training steps. (b) shows the relative convergence time with different label corruption ratio. (c) shows the test error (also the generalization error since training error is 0) under different label corruptions.

- Overparameterized, SGD-trained models :
  1. Can fit even completely random labels (i.e., huge capacity)
  2. Yet, generalize well

Open Question: How can this be?

Possible Explanations

- Maybe every model that fits the training data generalizes (no bad global minima)
- Maybe SGD "can avoid" bad global minima (implicit regularization)?

More on this next time.

Open Problem for projects

- Understanding the effects of overparameterization
- Deep vs wide networks (width helps in theory, depth in practice).
- Understanding when/if memorization hurts

reading list

Soudry, D. and Carmon, Y., 2016. No bad local minima: Data independent training error guarantees for multilayer neural networks. arXiv preprint arXiv:1605.08361.
https://arxiv.org/pdf/1605.08361

Du, S.S., Zhai, X., Poczos, B. and Singh, A., 2018. Gradient descent provably optimizes over-parameterized neural networks. ICLR 2019
https://arxiv.org/pdf/1810.02054

Allen-Zhu, Z., Li, Y. and Song, Z., 2019, May. A convergence theory for deep learning via over-parameterization. In International Conference on Machine Learning (pp. 242-252). PMLR.
http://proceedings.mlr.press/v97/allen-zhu19a/allen-zhu19a.pdf

Du, S., Lee, J., Li, H., Wang, L. and Zhai, X., 2019, May. Gradient descent finds global minima of deep neural networks. In International conference on machine learning (pp. 1675-1685). PMLR.
http://proceedings.mlr.press/v97/du19c/du19c.pdf

Liu, C., Zhu, L. and Belkin, M., 2022. Loss landscapes and optimization in over-parameterized non-linear systems and neural networks. Applied and Computational Harmonic Analysis.
Vancouver
https://arxiv.org/abs/2003.00307

Oymak, S. and Soltanolkotabi, M., 2019, May. Overparameterized nonlinear learning: Gradient descent takes the shortest path?. In International Conference on Machine Learning (pp. 4951-4960). PMLR.
http://proceedings.mlr.press/v97/oymak19a/oymak19a.pdf