Analyzing Optimization in Deep Learning via Trajectories

Nadav Cohen

Institute for Advanced Study

Institute for Computational and Experimental Research in Mathematics (ICERM)

Workshop on Theory and Practice in Machine Learning and Computer Vision

19 February 2019

EVERY INDUSTRY WANTS DEEP LEARNING

Cloud Service Provider	Medicine	Media & Entertainment	Security & Defense	Autonomous Machines

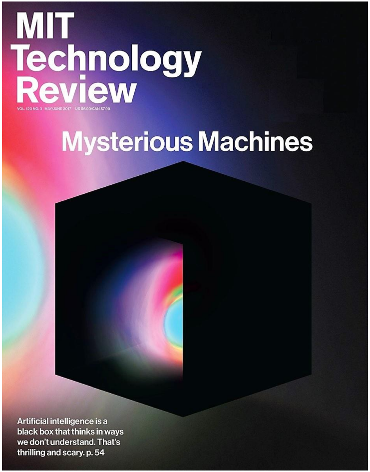
Cloud Service Provider	Medicine	Media & Entertainment	Security & Defense	Autonomous Machines
➢ Image/Video classification	➢ Cancer cell detection	➢ Video captioning	➢ Face recognition	➢ Pedestrian detection
➢ Speech recognition	➢ Diabetic grading	➢ Content based search	➢ Video surveillance	➢ Lane tracking
➢ Natural language processing	➢ Drug discovery	➢ Real time translation	➢ Cyber security	➢ Recognize traffic sign

NVIDIA

Source

NVIDIA (www.slideshare.net/openomics/the-revolution-of-deep-learning)

Limited Formal Understanding



Intelligent Machines

The Dark Secret at the Heart of AI

No one really knows how the most advanced algorithms do what they do. That could be a problem.

by Will Knight April 11, 2017

Last year, a strange self-driving car was released onto the quiet roads of Monmouth County, New Jersey. The experimental vehicle, developed by researchers at the chip maker Nvidia, didn't look different from other autonomous cars, but it was unlike anything demonstrated by Google, Tesla, or General Motors, and it showed the rising power of artificial intelligence. The car didn't follow a single instruction provided by an engineer or programmer. Instead, it relied entirely on an algorithm that had taught itself to drive by watching a human do it.

Outline

1. Deep Learning Theory: Expressiveness, Optimization and Generalization

2. Analyzing Optimization via Trajectories

3. Trajectories of Gradient Descent for Deep Linear Neural Networks
    - Convergence to Global Optimum
    - Acceleration by Depth

4. Conclusion

Statistical Learning Setup

Statistical Learning Setup

$\mathcal{X}$ — instance space (e.g. $\mathbb{R}^{100 \times 100}$ for 100-by-100 grayscale images)

Statistical Learning Setup

$\mathcal{X}$ — instance space (e.g. $\mathbb{R}^{100\times100}$ for 100-by-100 grayscale images)

$\mathcal{Y}$ — label space (e.g. $\mathbb{R}$ for regression or $\{1,\ldots,k\}$ for classification)

Statistical Learning Setup

$\mathcal{X}$ — instance space (e.g. $\mathbb{R}^{100\times100}$ for 100-by-100 grayscale images)

$\mathcal{Y}$ — label space (e.g. $\mathbb{R}$ for regression or $\{1,\ldots,k\}$ for classification)

$\mathcal{D}$ — distribution over $\mathcal{X}\times\mathcal{Y}$ (unknown)

Statistical Learning Setup

$\mathcal{X}$ — instance space (e.g. $\mathbb{R}^{100\times100}$ for 100-by-100 grayscale images)

$\mathcal{Y}$ — label space (e.g. $\mathbb{R}$ for regression or $\{1,\ldots,k\}$ for classification)

$\mathcal{D}$ — distribution over $\mathcal{X}\times\mathcal{Y}$ (unknown)

$\ell:\mathcal{Y}\times\mathcal{Y}\to\mathbb{R}_{\geq0}$ — loss func (e.g. $\ell(y,\hat{y})=(y-\hat{y})^2$ for $\mathcal{Y}=\mathbb{R}$)

Statistical Learning Setup

$\mathcal{X}$ — instance space (e.g. $\mathbb{R}^{100\times 100}$ for 100-by-100 grayscale images)

$\mathcal{Y}$ — label space (e.g. $\mathbb{R}$ for regression or $\{1, \ldots, k\}$ for classification)

$\mathcal{D}$ — distribution over $\mathcal{X} \times \mathcal{Y}$ (unknown)

$\ell : \mathcal{Y}\times\mathcal{Y} \to \mathbb{R}_{\geq 0}$ — loss func (e.g. $\ell(y, \hat{y}) = (y - \hat{y})^2$ for $\mathcal{Y} = \mathbb{R}$)

Task

Given training set $S = \{(X_i, y_i)\}_{i=1}^m$ drawn i.i.d. from $\mathcal{D}$, return hypothesis (predictor) $h : \mathcal{X} \to \mathcal{Y}$ that minimizes population loss:

$$L_{\mathcal{D}}(h) := \mathbb{E}_{(X,y)\sim\mathcal{D}}[\ell(y, h(X))]$$

Statistical Learning Setup

$\mathcal{X}$ — instance space (e.g. $\mathbb{R}^{100\times100}$ for 100-by-100 grayscale images)

$\mathcal{Y}$ — label space (e.g. $\mathbb{R}$ for regression or $\{1,\ldots,k\}$ for classification)

$\mathcal{D}$ — distribution over $\mathcal{X}\times\mathcal{Y}$ (unknown)

$\ell:\mathcal{Y}\times\mathcal{Y}\to\mathbb{R}_{\geq0}$ — loss func (e.g. $\ell(y,\hat{y})=(y-\hat{y})^2$ for $\mathcal{Y}=\mathbb{R}$)

Task

Given training set $S=\{(X_i,y_i)\}_{i=1}^m$ drawn i.i.d. from $\mathcal{D}$, return hypothesis (predictor) $h:\mathcal{X}\to\mathcal{Y}$ that minimizes population loss:

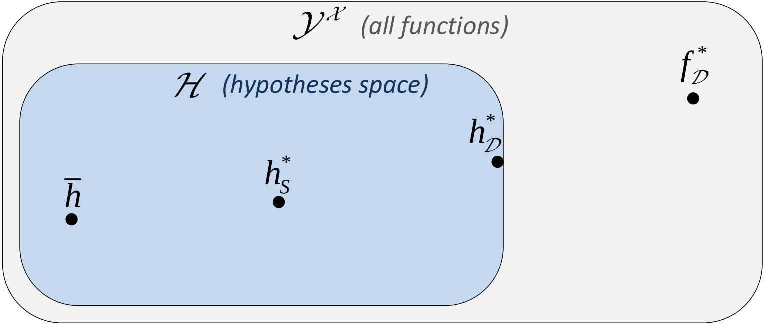
$$L_{\mathcal{D}}(h):=\mathbb{E}_{(X,y)\sim\mathcal{D}}[\ell(y,h(X))]$$

Approach

Predetermine hypotheses space $\mathcal{H}\subset\mathcal{Y}^{\mathcal{X}}$, and return hypothesis $h\in\mathcal{H}$ that minimizes empirical loss:

$$L_S(h):=\frac{1}{m}\sum\nolimits_{i=1}^m\ell(y_i,h(X_i))$$

Three Pillars of Statistical Learning Theory: Expressiveness, Generalization and Optimization



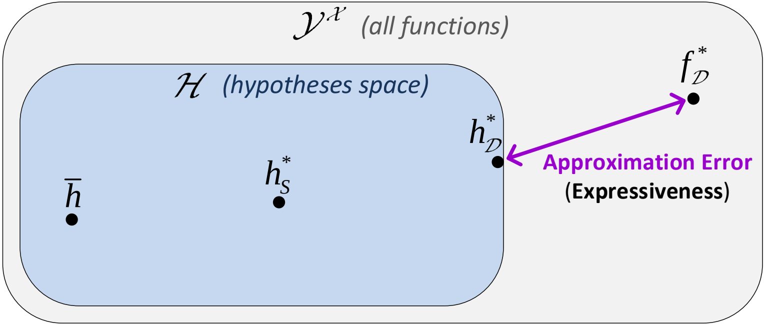
$f^*_{\mathcal{D}}$ — ground truth (minimizer of population loss over $\mathcal{Y}^{\mathcal{X}}$)

$h^*_{\mathcal{D}}$ — optimal hypothesis (minimizer of population loss over $\mathcal{H}$)

h^*_S — empirically optimal hypothesis (minimizer of empirical loss over $\mathcal{H}$)

$\bar{h}$ — returned hypothesis

Three Pillars of Statistical Learning Theory: Expressiveness, Generalization and Optimization



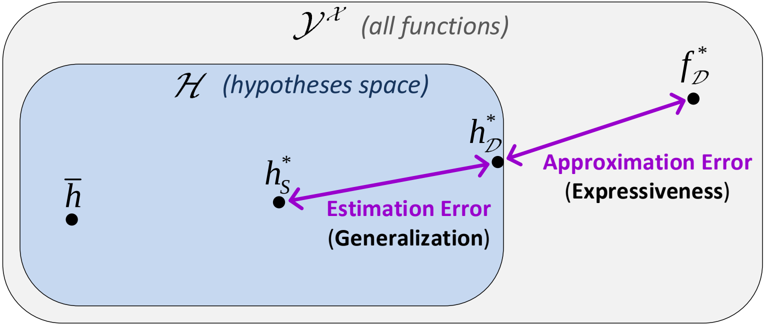
$f^*_{\mathcal{D}}$ — ground truth (minimizer of population loss over $\mathcal{Y}^{\mathcal{X}}$)

$h^*_{\mathcal{D}}$ — optimal hypothesis (minimizer of population loss over $\mathcal{H}$)

h^*_S — empirically optimal hypothesis (minimizer of empirical loss over $\mathcal{H}$)

$\bar{h}$ — returned hypothesis

Three Pillars of Statistical Learning Theory: Expressiveness, Generalization and Optimization



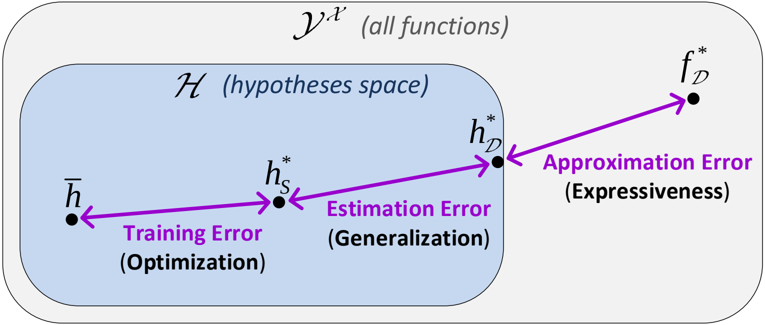
$f^*_{\mathcal{D}}$ — ground truth (minimizer of population loss over $\mathcal{Y}^{\mathcal{X}}$)

$h^*_{\mathcal{D}}$ — optimal hypothesis (minimizer of population loss over $\mathcal{H}$)

h^*_S — empirically optimal hypothesis (minimizer of empirical loss over $\mathcal{H}$)

$\bar{h}$ — returned hypothesis

Three Pillars of Statistical Learning Theory: Expressiveness, Generalization and Optimization



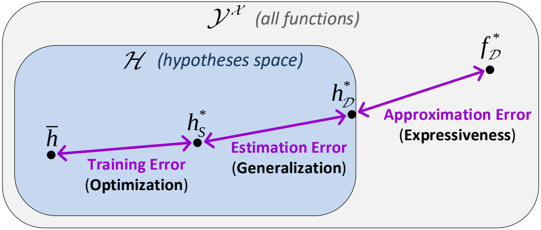
$f^*_{\mathcal{D}}$ — ground truth (minimizer of population loss over $\mathcal{Y}^{\mathcal{X}}$)

$h^*_{\mathcal{D}}$ — optimal hypothesis (minimizer of population loss over $\mathcal{H}$)

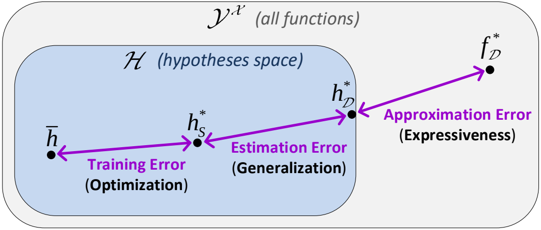
h^*_S — empirically optimal hypothesis (minimizer of empirical loss over $\mathcal{H}$)

$\bar{h}$ — returned hypothesis

Classical Machine Learning



Classical Machine Learning

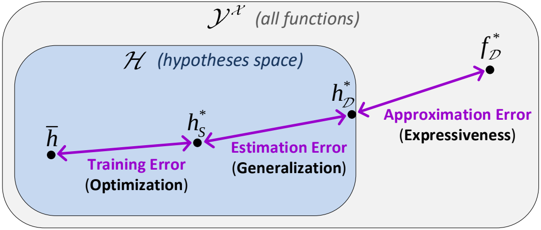


Optimization

Empirical loss minimization is a convex program:

$$\bar{h} \approx h^*_S \quad (\text{ training err} \approx 0 \text{ })$$

Classical Machine Learning



Optimization

Empirical loss minimization is a convex program:

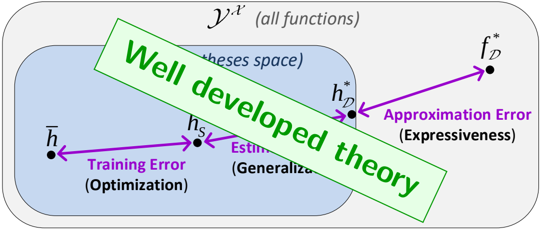
$$\bar{h} \approx h^*_S \quad (\text{ training err} \approx 0 \text{ })$$

Expressiveness & Generalization

Bias-variance trade-off:

$\mathcal{H}$	approximation err	estimation err
expands	$\searrow$	$\nearrow$
shrinks	$\nearrow$	$\searrow$

Classical Machine Learning



Optimization

Empirical loss minimization is a convex program:

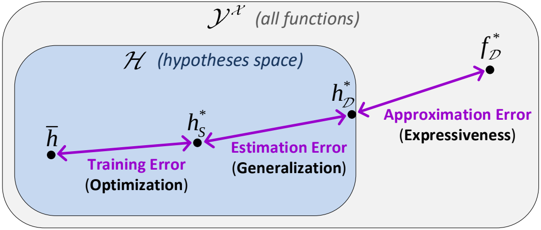
$$\bar{h} \approx h^*_S \quad (\text{ training err} \approx 0 \ )$$

Expressiveness & Generalization

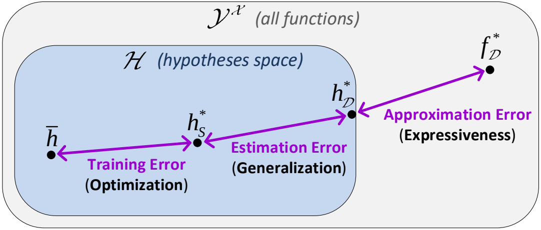
Bias-variance trade-off:

$\mathcal{H}$	approximation err	estimation err
expands	↘	↗
shrinks	↗	↘

Deep Learning



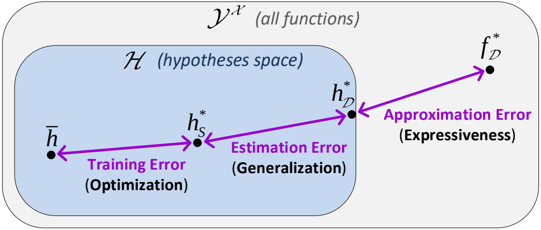
Deep Learning



Optimization

Empirical loss minimization is a non-convex program:

Deep Learning

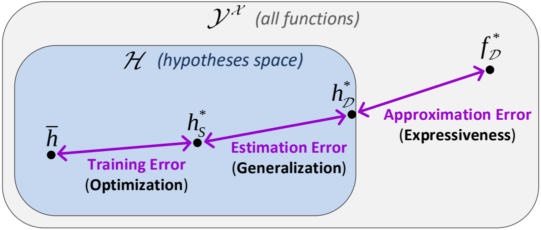


Optimization

Empirical loss minimization is a non-convex program:

- h^*_S is not unique — many hypotheses have low training err

Deep Learning

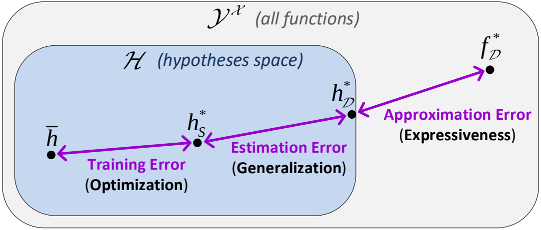


Optimization

Empirical loss minimization is a non-convex program:

- h^*_S is not unique — many hypotheses have low training err
- Gradient descent (GD) somehow reaches one of these

Deep Learning



Optimization

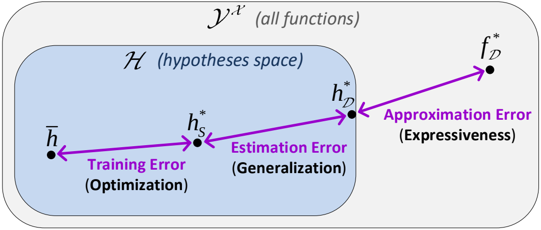
Empirical loss minimization is a non-convex program:

- h^*_S is not unique — many hypotheses have low training err
- Gradient descent (GD) somehow reaches one of these

Expressiveness & Generalization

Vast difference from classical ML:

Deep Learning



Optimization

Empirical loss minimization is a non-convex program:

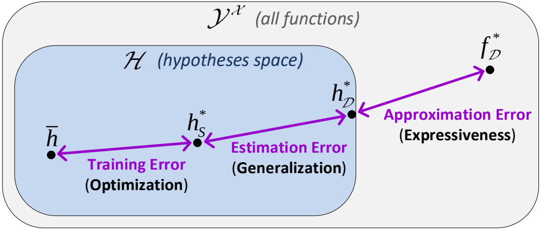
- h^*_S is not unique — many hypotheses have low training err
- Gradient descent (GD) somehow reaches one of these

Expressiveness & Generalization

Vast difference from classical ML:

- Some low training err hypotheses generalize well, others don't

Deep Learning



Optimization

Empirical loss minimization is a non-convex program:

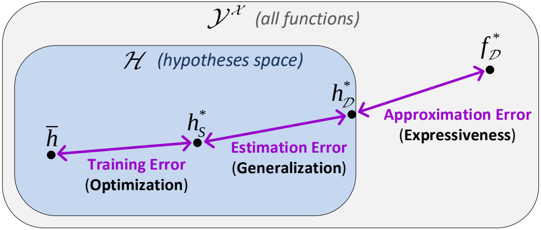
- h^*_S is not unique — many hypotheses have low training err
- Gradient descent (GD) somehow reaches one of these

Expressiveness & Generalization

Vast difference from classical ML:

- Some low training err hypotheses generalize well, others don't
- W/typical data, solution returned by GD often generalizes well

Deep Learning



Optimization

Empirical loss minimization is a non-convex program:

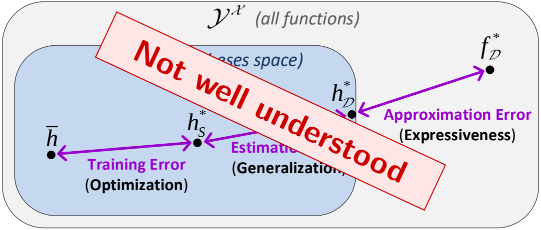
- h^*_S is not unique — many hypotheses have low training err
- Gradient descent (GD) somehow reaches one of these

Expressiveness & Generalization

Vast difference from classical ML:

- Some low training err hypotheses generalize well, others don't
- W/typical data, solution returned by GD often generalizes well
- Expanding $\mathcal{H}$ reduces approximation err, but also estimation err!

Deep Learning



Optimization

Empirical loss minimization is a non-convex program:

- h^*_S is not unique — many hypotheses have low training err
- Gradient descent (GD) somehow reaches one of these

Expressiveness & Generalization

Vast difference from classical ML:

- Some low training err hypotheses generalize well, others don't
- W/typical data, solution returned by GD often generalizes well
- Expanding $\mathcal{H}$ reduces approximation err, but also estimation err!

Outline

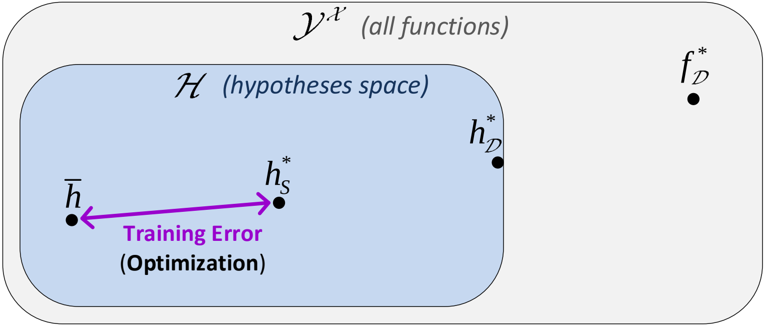
1 Deep Learning Theory: Expressiveness, Optimization and Generalization

2 Analyzing Optimization via Trajectories

3 Trajectories of Gradient Descent for Deep Linear Neural Networks
- Convergence to Global Optimum
- Acceleration by Depth

4 Conclusion

Optimization



$f^*_{\mathcal{D}}$ — ground truth

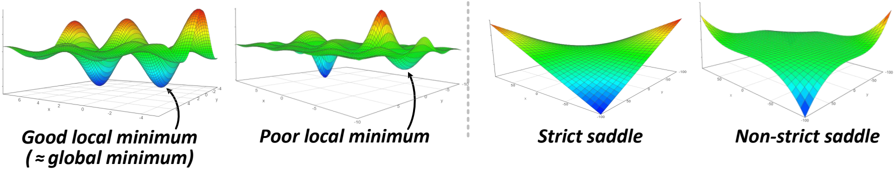
$h^*_{\mathcal{D}}$ — optimal hypothesis

h^*_S — empirically optimal hypothesis

$\bar{h}$ — returned hypothesis

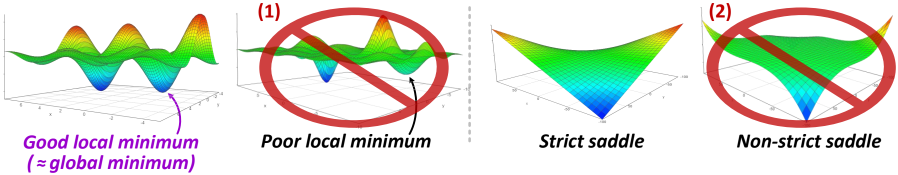
Approach: Convergence via Critical Points

Prominent approach for analyzing optimization in DL is via critical points ($\nabla = 0$) in loss landscape



Approach: Convergence via Critical Points

Prominent approach for analyzing optimization in DL is via critical points ($\nabla = 0$) in loss landscape

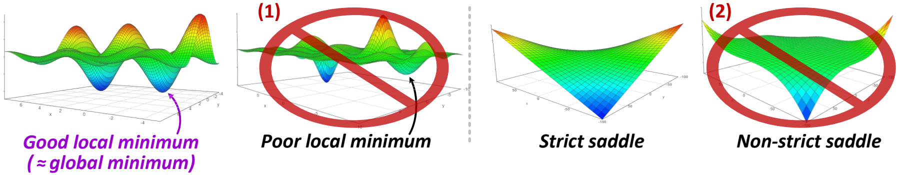


Result *(cf. Ge et al. 2015; Lee et al. 2016)*

If: **(1)** there are no poor local minima; and **(2)** all saddle points are strict, then gradient descent (GD) converges to global minimum

Approach: Convergence via Critical Points

Prominent approach for analyzing optimization in DL is via critical points ($\nabla = 0$) in loss landscape



Result *(cf. Ge et al. 2015; Lee et al. 2016)*

If: **(1)** there are no poor local minima; and **(2)** all saddle points are strict, then gradient descent (GD) converges to global minimum

Motivated by this, many[1] studied the validity of **(1)** and/or **(2)**

[1] e.g. Haeffele & Vidal 2015; Kawaguchi 2016; Soudry & Carmon 2016; Safran & Shamir 2018

Limitations

Convergence of GD to global min was proven via critical points only for problems involving shallow (2 layer) models

Limitations

Convergence of GD to global min was proven via critical points only for problems involving shallow (2 layer) models

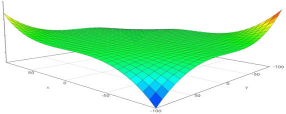
Approach is insufficient when treating deep (≥ 3 layer) models:

Limitations

Convergence of GD to global min was proven via critical points only for problems involving shallow (2 layer) models

Approach is insufficient when treating deep (≥ 3 layer) models:

- **(2)** is violated — $\exists$ non-strict saddles, e.g. when all weights $= 0$

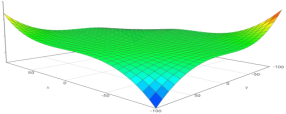


Limitations

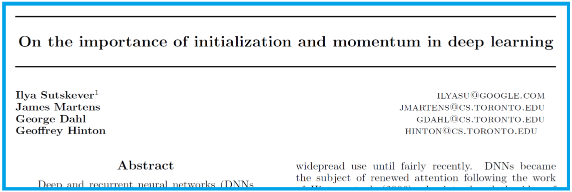
Convergence of GD to global min was proven via critical points only for problems involving shallow (2 layer) models

Approach is insufficient when treating deep (≥ 3 layer) models:

- **(2)** is violated — $\exists$ non-strict saddles, e.g. when all weights $= 0$



- Algorithmic aspects essential for convergence w/deep models, e.g. proper initialization, are ignored



On the importance of initialization and momentum in deep learning

Ilya Sutskever[1] ILYASU@GOOGLE.COM
James Martens JMARTENS@CS.TORONTO.EDU
George Dahl GDAHL@CS.TORONTO.EDU
Geoffrey Hinton HINTON@CS.TORONTO.EDU

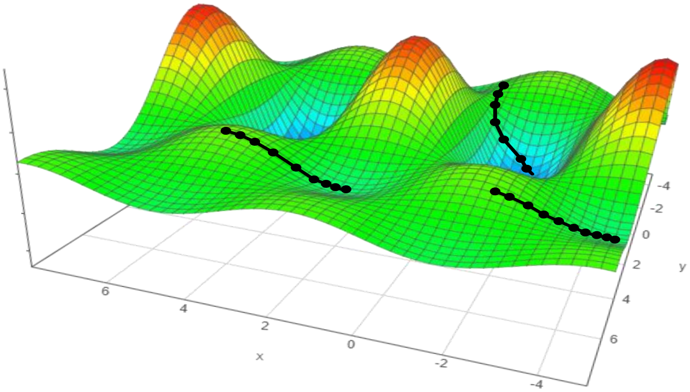
Abstract

Deep and recurrent neural networks (DNNs

widespread use until fairly recently. DNNs became the subject of renewed attention following the work

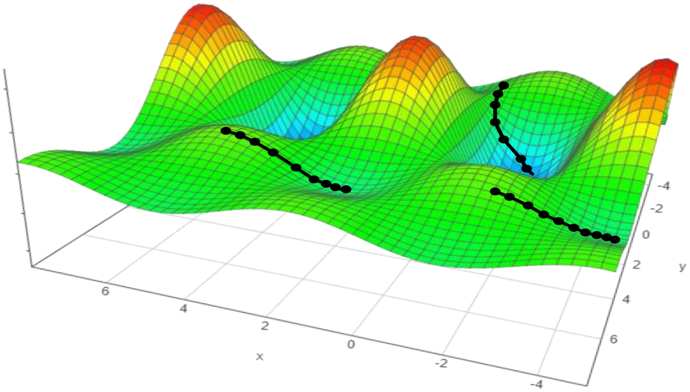
Optimizer Trajectories Matter

Different optimization trajectories may lead to qualitatively different results



Optimizer Trajectories Matter

Different optimization trajectories may lead to qualitatively different results



$\Longrightarrow$ details of algorithm and init should be taken into account!

Existing Trajectory Analyses

Existing Trajectory Analyses

Trajectory approach led to successful analyses of shallow models:

- Brutzkus & Globerson 2017
- Li & Yuan 2017
- Zhong et al. 2017
- Tian 2017
- Brutzkus et al. 2018
- Li et al. 2018
- Du et al. 2018
- Oymak & Soltanolkotabi 2018

Existing Trajectory Analyses

Trajectory approach led to successful analyses of shallow models:

- Brutzkus & Globerson 2017
- Li & Yuan 2017
- Zhong et al. 2017
- Tian 2017
- Brutzkus et al. 2018
- Li et al. 2018
- Du et al. 2018
- Oymak & Soltanolkotabi 2018

It also allowed treating prohibitively large deep models:

- Du et al. 2018
- Allen-Zhu et al. 2018
- Zou et al. 2018

Existing Trajectory Analyses

Trajectory approach led to successful analyses of shallow models:

- Brutzkus & Globerson 2017
- Li & Yuan 2017
- Zhong et al. 2017
- Tian 2017
- Brutzkus et al. 2018
- Li et al. 2018
- Du et al. 2018
- Oymak & Soltanolkotabi 2018

It also allowed treating prohibitively large deep models:

- Du et al. 2018
- Allen-Zhu et al. 2018
- Zou et al. 2018

For deep linear residual networks, trajectories were used to show efficient convergence of GD to global min (Bartlett et al. 2018)

Outline

1. Deep Learning Theory: Expressiveness, Optimization and Generalization
2. Analyzing Optimization via Trajectories
3. Trajectories of Gradient Descent for Deep Linear Neural Networks
   - Convergence to Global Optimum
   - Acceleration by Depth
4. Conclusion

Sources

On the Optimization of Deep Networks: Implicit Acceleration by Overparameterization

Arora + ***C*** + Hazan (alphabetical order)

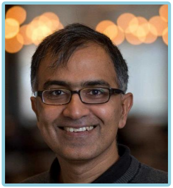
International Conference on Machine Learning (ICML) 2018

A Convergence Analysis of Gradient Descent for Deep Linear Neural Networks

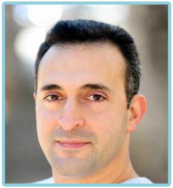
Arora + ***C*** + Golowich + Hu (alphabetical order)

To appear: International Conference on Learning Representations (ICLR) 2019

Collaborators

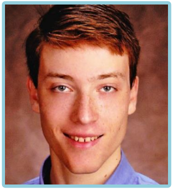


Sanjeev Arora



Elad Hazan

Wei Hu



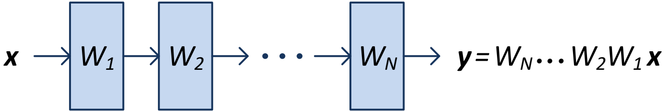
Noah Golowich

Outline

1. Deep Learning Theory: Expressiveness, Optimization and Generalization
2. Analyzing Optimization via Trajectories
3. Trajectories of Gradient Descent for Deep Linear Neural Networks
   - Convergence to Global Optimum
   - Acceleration by Depth
4. Conclusion

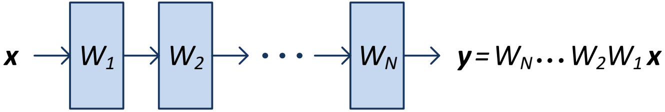
Linear Neural Networks

Linear neural networks (LNN) are fully-connected neural networks w/linear (no) activation



Linear Neural Networks

Linear neural networks (LNN) are fully-connected neural networks w/linear (no) activation

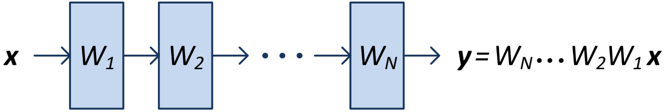


As surrogate for optimization in DL, GD over LNN (highly non-convex problem) is studied extensively [1]

[1] e.g. Saxe et al. 2014; Kawaguchi 2016; Hardt & Ma 2017; Laurent & Brecht 2018

Linear Neural Networks

Linear neural networks (LNN) are fully-connected neural networks w/linear (no) activation



As surrogate for optimization in DL, GD over LNN (highly non-convex problem) is studied extensively [1]

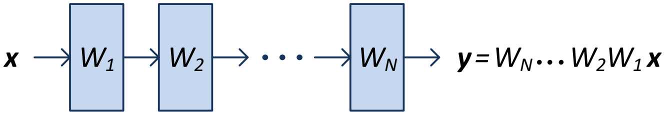
Existing Result *(Bartlett et al. 2018)*

W/linear residual networks (a special case: W_j are square and init to I_d), for ℓ_2 loss on certain data, GD efficiently converges to global min

[1] e.g. Saxe et al. 2014; Kawaguchi 2016; Hardt & Ma 2017; Laurent & Brecht 2018

Linear Neural Networks

Linear neural networks (LNN) are fully-connected neural networks w/linear (no) activation



As surrogate for optimization in DL, GD over LNN (highly non-convex problem) is studied extensively [1]

Existing Result *(Bartlett et al. 2018)*

W/linear residual networks (a special case: W_j are square and init to I_d), for ℓ_2 loss on certain data, GD efficiently converges to global min

↑

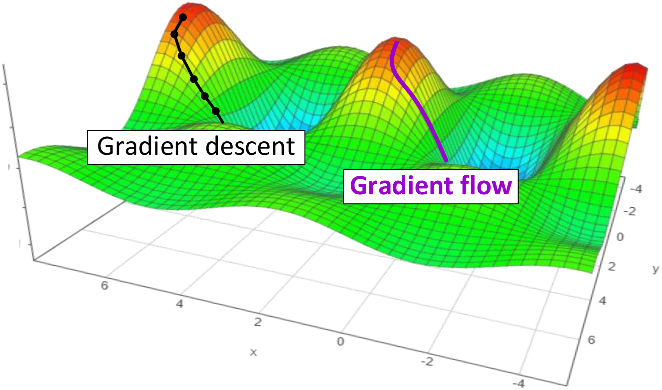
Only existing proof of efficient convergence to global min for GD training deep model

[1] e.g. Saxe et al. 2014; Kawaguchi 2016; Hardt & Ma 2017; Laurent & Brecht 2018

Gradient Flow

Gradient flow (GF) is a continuous version of GD (learning rate $\to 0$):

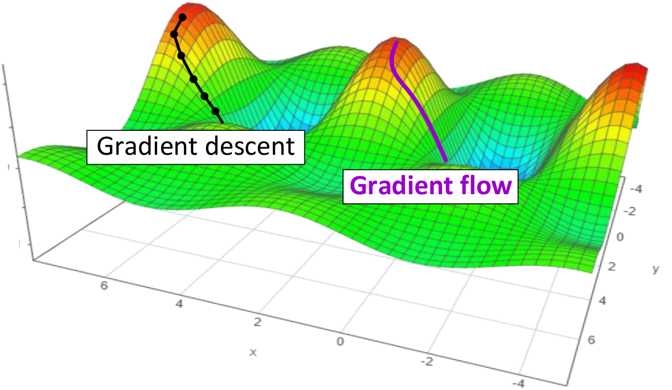
$$\frac{d}{dt}\alpha(t) = -\nabla f(\alpha(t)) \quad , \; t \in \mathbb{R}_{>0}$$



Gradient Flow

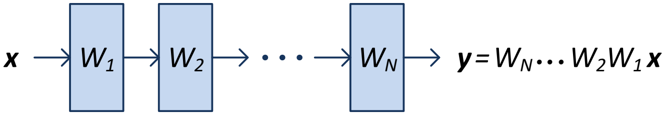
Gradient flow (GF) is a continuous version of GD (learning rate $\to 0$):

$$\frac{d}{dt}\alpha(t) = -\nabla f(\alpha(t)) \quad , \, t \in \mathbb{R}_{>0}$$

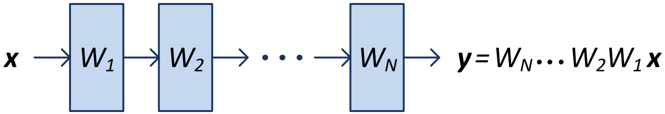


Admits use of theoretical tools from differential geometry/equations

Trajectories of Gradient Flow



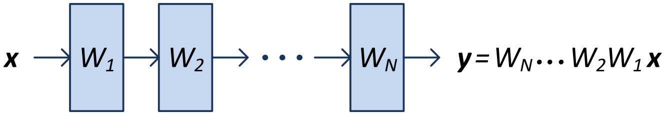
Trajectories of Gradient Flow



Loss $\ell(\cdot)$ for linear model induces **overparameterized objective** for LNN:

$$\phi(W_1, \ldots, W_N) := \ell(W_N \cdots W_2 W_1)$$

Trajectories of Gradient Flow



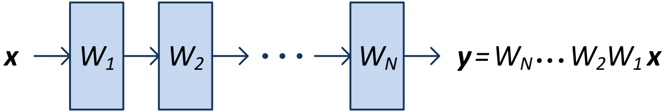
Loss $\ell(\cdot)$ for linear model induces **overparameterized objective** for LNN:

$$\phi(W_1, \ldots, W_N) := \ell(W_N \cdots W_2 W_1)$$

Definition

Weights $W_1 \ldots W_N$ are **balanced** if $W_{j+1}^\top W_{j+1} = W_j W_j^\top \ , \forall j$.

Trajectories of Gradient Flow



Loss $\ell(\cdot)$ for linear model induces **overparameterized objective** for LNN:

$$\phi(W_1, \ldots, W_N) := \ell(W_N \cdots W_2 W_1)$$

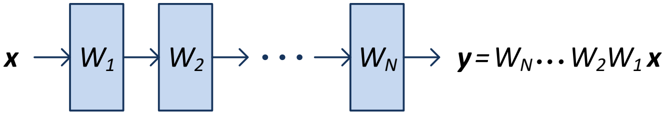
Definition

Weights $W_1 \ldots W_N$ are **balanced** if $W_{j+1}^\top W_{j+1} = W_j W_j^\top$, $\forall j$.

↑

Holds approximately under ≈ 0 init, exactly under residual (I_d) init

Trajectories of Gradient Flow



Loss $\ell(\cdot)$ for linear model induces **overparameterized objective** for LNN:

$$\phi(W_1, \ldots, W_N) := \ell(W_N \cdots W_2 W_1)$$

Definition

Weights $W_1 \ldots W_N$ are **balanced** if $W_{j+1}^\top W_{j+1} = W_j W_j^\top \ , \forall j$.

↑

Holds approximately under ≈ 0 init, exactly under residual (I_d) init

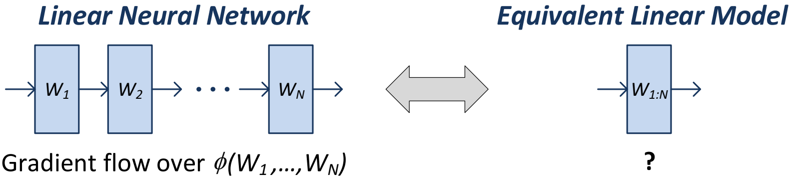
Claim

Trajectories of GF over LNN preserve balancedness: if $W_1 \ldots W_N$ are balanced at init, they remain that way throughout GF optimization

Implicit Preconditioning

Question

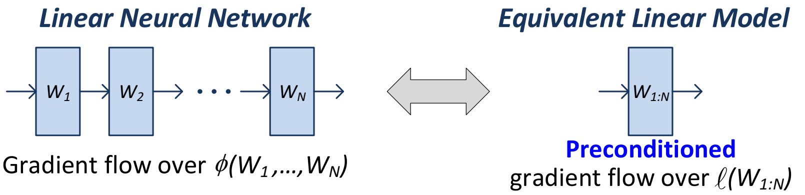
How does **end-to-end matrix** $W_{1:N} := W_N \cdots W_1$ move on GF trajectories?



Implicit Preconditioning

Question

How does **end-to-end matrix** $W_{1:N} := W_N \cdots W_1$ move on GF trajectories?



Theorem

*If $W_1 \ldots W_N$ are balanced at init, $W_{1:N}$ follows **end-to-end dynamics**:*

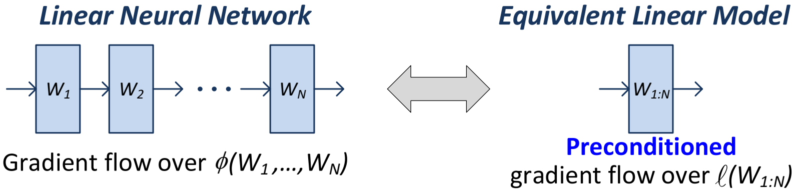
$$\frac{d}{dt} vec\left[W_{1:N}(t)\right] = -P_{W_{1:N}(t)} \cdot vec\left[\nabla \ell(W_{1:N}(t))\right]$$

where $P_{W_{1:N}(t)}$ is a preconditioner (PSD matrix) that "reinforces" $W_{1:N}(t)$

Implicit Preconditioning

Question

How does **end-to-end matrix** $W_{1:N} := W_N \cdots W_1$ move on GF trajectories?



Theorem

*If $W_1 \ldots W_N$ are balanced at init, $W_{1:N}$ follows **end-to-end dynamics**:*

$$\tfrac{d}{dt} vec\left[W_{1:N}(t)\right] = -P_{W_{1:N}(t)} \cdot vec\left[\nabla \ell(W_{1:N}(t))\right]$$

where $P_{W_{1:N}(t)}$ is a preconditioner (PSD matrix) that "reinforces" $W_{1:N}(t)$

Adding (redundant) linear layers to classic linear model induces preconditioner promoting movement in directions already taken!

Convergence to Global Optimum

$$\frac{d}{dt} vec\left[W_{1:N}(t)\right] = -P_{W_{1:N}(t)} \cdot vec\left[\nabla \ell(W_{1:N}(t))\right]$$

Convergence to Global Optimum

$$\frac{d}{dt} vec\left[W_{1:N}(t)\right] = -P_{W_{1:N}(t)} \cdot vec\left[\nabla \ell(W_{1:N}(t))\right]$$

$P_{W_{1:N}(t)} \succ 0$ when $W_{1:N}(t)$ has full rank

Convergence to Global Optimum

$$\frac{d}{dt} vec\left[W_{1:N}(t)\right] = -P_{W_{1:N}(t)} \cdot vec\left[\nabla \ell(W_{1:N}(t))\right]$$

$P_{W_{1:N}(t)} \succ 0$ when $W_{1:N}(t)$ has full rank $\implies$ loss decreases until:

(1) $\nabla \ell(W_{1:N}(t)) = 0$ **or** **(2)** $W_{1:N}(t)$ is singular

Convergence to Global Optimum

$$\frac{d}{dt} vec\left[W_{1:N}(t)\right] = -P_{W_{1:N}(t)} \cdot vec\left[\nabla \ell(W_{1:N}(t))\right]$$

$P_{W_{1:N}(t)} \succ 0$ when $W_{1:N}(t)$ has full rank $\implies$ loss decreases until:

(1) $\nabla \ell(W_{1:N}(t)) = 0$ **or** **(2)** $W_{1:N}(t)$ is singular

$\ell(\cdot)$ is typically convex $\implies$ **(1)** means global min was reached

Convergence to Global Optimum

$$\frac{d}{dt} vec\left[W_{1:N}(t)\right] = -P_{W_{1:N}(t)} \cdot vec\left[\nabla \ell(W_{1:N}(t))\right]$$

$P_{W_{1:N}(t)} \succ 0$ when $W_{1:N}(t)$ has full rank $\implies$ loss decreases until:

(1) $\nabla \ell(W_{1:N}(t)) = 0$ **or** **(2)** $W_{1:N}(t)$ is singular

$\ell(\cdot)$ is typically convex $\implies$ **(1)** means global min was reached

Corollary

Assume $\ell(\cdot)$ is convex and LNN is init such that:

- $W_1 \ldots W_N$ *are balanced*
- $\ell(W_{1:N}) < \ell(W)$ *for any singular* W

Then, GF converges to global min

From Gradient Flow to Gradient Descent

From Gradient Flow to Gradient Descent

Our convergence result for GF made two assumptions on init:

From Gradient Flow to Gradient Descent

Our convergence result for GF made two assumptions on init:

1. Weights are balanced:

$$W_{j+1}^{\top} W_{j+1} = W_j W_j^{\top} \quad , \forall j$$

From Gradient Flow to Gradient Descent

Our convergence result for GF made two assumptions on init:

1. Weights are balanced:

$$\|W_{j+1}^\top W_{j+1} - W_j W_j^\top\|_F = 0 \quad ,\forall j$$

From Gradient Flow to Gradient Descent

Our convergence result for GF made two assumptions on init:

1. Weights are balanced:

$$\|W_{j+1}^\top W_{j+1} - W_j W_j^\top\|_F = 0 \quad ,\forall j$$

2. Loss is smaller than that of any singular solution:

$$\ell(W_{1:N}) < \ell(W) \quad ,\forall W \ s.t. \ \sigma_{min}(W) = 0$$

From Gradient Flow to Gradient Descent

Our convergence result for GF made two assumptions on init:

1. Weights are balanced:

$$\|W_{j+1}^\top W_{j+1} - W_j W_j^\top\|_F = 0 \quad ,\forall j$$

2. Loss is smaller than that of any singular solution:

$$\ell(W_{1:N}) < \ell(W) \quad ,\forall W \; s.t. \; \sigma_{min}(W) = 0$$

For translating to GD, we define discrete forms of these conditions:

From Gradient Flow to Gradient Descent

Our convergence result for GF made two assumptions on init:

1. Weights are balanced:
$$\|W_{j+1}^\top W_{j+1} - W_j W_j^\top\|_F = 0 \quad ,\forall j$$
2. Loss is smaller than that of any singular solution:
$$\ell(W_{1:N}) < \ell(W) \quad ,\forall W \text{ s.t. } \sigma_{min}(W) = 0$$

For translating to GD, we define discrete forms of these conditions:

Definition

For $\delta \geq 0$, weights $W_1 \ldots W_N$ are **δ-balanced** if:
$$\|W_{j+1}^\top W_{j+1} - W_j W_j^\top\|_F \leq \boldsymbol{\delta} \quad ,\forall j$$

From Gradient Flow to Gradient Descent

Our convergence result for GF made two assumptions on init:

1. Weights are balanced:

$$\|W_{j+1}^\top W_{j+1} - W_j W_j^\top\|_F = 0 \quad ,\forall j$$

2. Loss is smaller than that of any singular solution:

$$\ell(W_{1:N}) < \ell(W) \quad ,\forall W \text{ } s.t. \text{ } \sigma_{min}(W) = 0$$

For translating to GD, we define discrete forms of these conditions:

Definition

For $\delta \geq 0$, weights $W_1 \ldots W_N$ are **δ-balanced** if:

$$\|W_{j+1}^\top W_{j+1} - W_j W_j^\top\|_F \leq \boldsymbol{\delta} \quad ,\forall j$$

Definition

For $c > 0$, weights $W_1 \ldots W_N$ have **deficiency margin c** if:

$$\ell(W_{1:N}) \leq \ell(W) \quad ,\forall W \text{ } s.t. \text{ } \sigma_{min}(W) \leq \mathbf{c}$$

Convergence to Global Optimum for Gradient Descent

Suppose $\ell(\cdot) = \ell_2$ loss $\left(\text{i.e. } \ell(W) = \frac{1}{m}\sum_{i=1}^{m} \|W\mathbf{x}_i - \mathbf{y}_i\|_2^2\right)$

Convergence to Global Optimum for Gradient Descent

Suppose $\ell(\cdot) = \ell_2$ loss $\left(\text{i.e. } \ell(W) = \frac{1}{m}\sum_{i=1}^{m}\|W\mathbf{x}_i - \mathbf{y}_i\|_2^2\right)$

Theorem

Assume GD over LNN is init s.t. $W_1 \ldots W_N$ have deficiency margin $c > 0$ and are δ-balanced w/ $\delta \leq \mathcal{O}(c^2)$. Then, for any learning rate $\eta \leq \mathcal{O}(c^4)$:

$$\text{loss(iteration } t) \;\leq\; e^{-\Omega(c^2\eta t)}$$

Convergence to Global Optimum for Gradient Descent

Suppose $\ell(\cdot) = \ell_2$ loss $\left(\text{i.e. } \ell(W) = \frac{1}{m}\sum_{i=1}^{m} \|W\mathbf{x}_i - \mathbf{y}_i\|_2^2\right)$

Theorem

Assume GD over LNN is init s.t. $W_1 \ldots W_N$ have deficiency margin $c > 0$ and are δ-balanced w/ $\delta \leq \mathcal{O}(c^2)$. Then, for any learning rate $\eta \leq \mathcal{O}(c^4)$:

$$loss(iteration\ t) \ \leq \ e^{-\Omega(c^2 \eta t)}$$

Claim

Assumptions on init — deficiency margin and δ-balancedness:

Convergence to Global Optimum for Gradient Descent

Suppose $\ell(\cdot) = \ell_2$ loss $\left(\text{i.e. } \ell(W) = \frac{1}{m}\sum_{i=1}^{m} \|W\mathbf{x}_i - \mathbf{y}_i\|_2^2\right)$

Theorem

Assume GD over LNN is init s.t. $W_1 \ldots W_N$ have deficiency margin $c > 0$ and are δ-balanced w/ $\delta \leq \mathcal{O}(c^2)$. Then, for any learning rate $\eta \leq \mathcal{O}(c^4)$:

$$\textit{loss(iteration } t\textit{)} \;\leq\; e^{-\Omega(c^2 \eta t)}$$

Claim

Assumptions on init — deficiency margin and δ-balancedness:

- *Are necessary (violating any of them can lead to divergence)*

Convergence to Global Optimum for Gradient Descent

Suppose $\ell(\cdot) = \ell_2$ loss $\left(\text{i.e. } \ell(W) = \frac{1}{m}\sum_{i=1}^{m} \|W\mathbf{x}_i - \mathbf{y}_i\|_2^2\right)$

Theorem

Assume GD over LNN is init s.t. $W_1 \ldots W_N$ have deficiency margin $c > 0$ and are δ-balanced w/ $\delta \leq \mathcal{O}(c^2)$. Then, for any learning rate $\eta \leq \mathcal{O}(c^4)$:

$$\textit{loss(iteration } t\textit{)} \;\leq\; e^{-\Omega(c^2 \eta t)}$$

Claim

Assumptions on init — deficiency margin and δ-balancedness:

- *Are necessary (violating any of them can lead to divergence)*
- *For output dim 1, hold w/const prob under random "balanced" init*

Convergence to Global Optimum for Gradient Descent

Suppose $\ell(\cdot) = \ell_2$ loss $\left(\text{i.e. } \ell(W) = \frac{1}{m}\sum_{i=1}^{m} \|W\mathbf{x}_i - \mathbf{y}_i\|_2^2\right)$

Theorem

Assume GD over LNN is init s.t. $W_1 \ldots W_N$ have deficiency margin $c > 0$ and are δ-balanced w/$\delta \leq \mathcal{O}(c^2)$. Then, for any learning rate $\eta \leq \mathcal{O}(c^4)$:

$$loss(iteration\ t) \ \leq \ e^{-\Omega(c^2\eta t)}$$

Claim

Assumptions on init — deficiency margin and δ-balancedness:

- *Are necessary (violating any of them can lead to divergence)*
- *For output dim 1, hold w/const prob under random "balanced" init*

Guarantee of efficient (linear rate) convergence to global min!
Most general guarantee to date for GD efficiently training deep net.

Outline

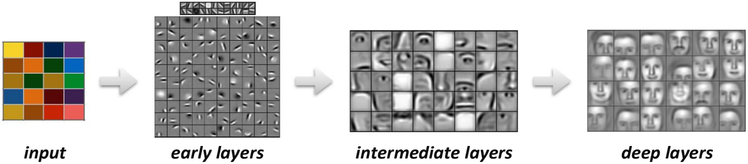
1. Deep Learning Theory: Expressiveness, Optimization and Generalization
2. Analyzing Optimization via Trajectories
3. Trajectories of Gradient Descent for Deep Linear Neural Networks
   - Convergence to Global Optimum
   - Acceleration by Depth
4. Conclusion

The Effect of Depth

The Effect of Depth

Conventional wisdom:

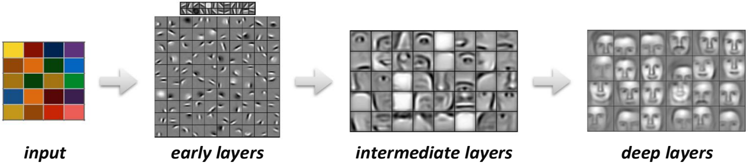
- Depth boosts expressiveness



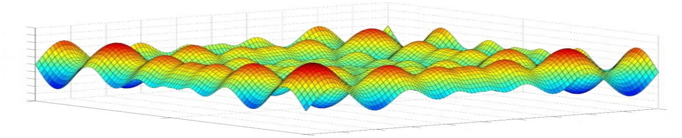
The Effect of Depth

Conventional wisdom:

- Depth boosts expressiveness



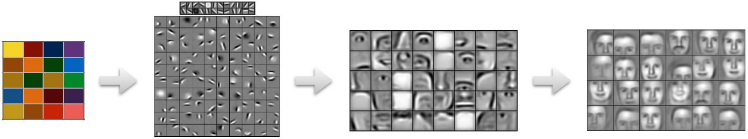
- But complicates optimization



The Effect of Depth

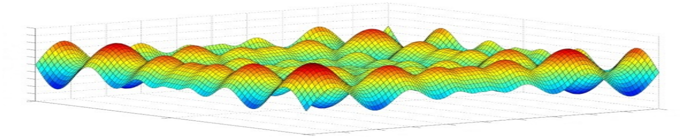
Conventional wisdom:

- Depth boosts expressiveness



input ***early layers*** ***intermediate layers*** ***deep layers***

- But complicates optimization



We will see: not always true...

Effect of Depth for Linear Neural Networks

Effect of Depth for Linear Neural Networks

For LNN, we derived end-to-end dynamics:

$$\frac{d}{dt} vec\left[W_{1:N}(t)\right] = -P_{W_{1:N}(t)} \cdot vec\left[\nabla \ell(W_{1:N}(t))\right]$$

Effect of Depth for Linear Neural Networks

For LNN, we derived end-to-end dynamics:

$$\frac{d}{dt} vec\left[W_{1:N}(t)\right] = -P_{W_{1:N}(t)} \cdot vec\left[\nabla\ell(W_{1:N}(t))\right]$$

Consider a discrete version:

$$vec\Big[W_{1:N}(t+1)\Big] \leftarrow vec\Big[W_{1:N}(t)\Big] - \eta \cdot P_{W_{1:N}(t)} \cdot vec\Big[\nabla\ell(W_{1:N}(t))\Big]$$

Effect of Depth for Linear Neural Networks

For LNN, we derived end-to-end dynamics:

$$\frac{d}{dt} vec\left[W_{1:N}(t)\right] = -P_{W_{1:N}(t)} \cdot vec\left[\nabla\ell(W_{1:N}(t))\right]$$

Consider a discrete version:

$$vec\left[W_{1:N}(t+1)\right] \leftarrow vec\left[W_{1:N}(t)\right] - \eta \cdot P_{W_{1:N}(t)} \cdot vec\left[\nabla\ell(W_{1:N}(t))\right]$$

Claim

For any $p > 2$, there exist settings where $\ell(\cdot) = \ell_p$ loss:

$$\ell(W) = \frac{1}{m}\sum_{i=1}^{m} \|W\mathbf{x}_i - \mathbf{y}_i\|_p^p$$

Effect of Depth for Linear Neural Networks

For LNN, we derived end-to-end dynamics:

$$\tfrac{d}{dt} vec\left[W_{1:N}(t)\right] = -P_{W_{1:N}(t)} \cdot vec\left[\nabla \ell(W_{1:N}(t))\right]$$

Consider a discrete version:

$$vec\Big[W_{1:N}(t+1)\Big] \leftarrow vec\Big[W_{1:N}(t)\Big] - \eta \cdot P_{W_{1:N}(t)} \cdot vec\Big[\nabla \ell(W_{1:N}(t))\Big]$$

Claim

For any $p > 2$, there exist settings where $\ell(\cdot) = \ell_p$ loss:

$$\ell(W) = \tfrac{1}{m}\textstyle\sum_{i=1}^{m} \|W\mathbf{x}_i - \mathbf{y}_i\|_p^p \quad \leftarrow \text{convex}$$

Effect of Depth for Linear Neural Networks

For LNN, we derived end-to-end dynamics:

$$\tfrac{d}{dt} vec\left[W_{1:N}(t)\right] = -P_{W_{1:N}(t)} \cdot vec\left[\nabla \ell(W_{1:N}(t))\right]$$

Consider a discrete version:

$$vec\Big[W_{1:N}(t+1)\Big] \leftarrow vec\Big[W_{1:N}(t)\Big] - \eta \cdot P_{W_{1:N}(t)} \cdot vec\Big[\nabla \ell(W_{1:N}(t))\Big]$$

Claim

For any $p > 2$, there exist settings where $\ell(\cdot) = \ell_p$ loss:

$$\ell(W) = \tfrac{1}{m}\textstyle\sum_{i=1}^{m} \|W\mathbf{x}_i - \mathbf{y}_i\|_p^p \quad \leftarrow \textit{convex}$$

and disc end-to-end dynamics reach global min arbitrarily faster than GD

Effect of Depth for Linear Neural Networks

For LNN, we derived end-to-end dynamics:

$$\frac{d}{dt} vec\left[W_{1:N}(t)\right] = -P_{W_{1:N}(t)} \cdot vec\left[\nabla \ell(W_{1:N}(t))\right]$$

Consider a discrete version:

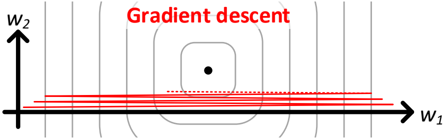
$$vec\left[W_{1:N}(t+1)\right] \hookleftarrow vec\left[W_{1:N}(t)\right] - \eta \cdot P_{W_{1:N}(t)} \cdot vec\left[\nabla \ell(W_{1:N}(t))\right]$$

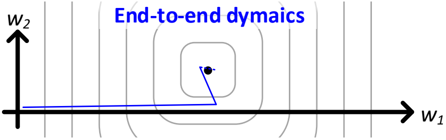
Claim

For any $p > 2$, there exist settings where $\ell(\cdot) = \ell_p$ loss:

$$\ell(W) = \frac{1}{m}\sum_{i=1}^{m} \|W\mathbf{x}_i - \mathbf{y}_i\|_p^p \quad \leftarrow \text{convex}$$

and disc end-to-end dynamics reach global min arbitrarily faster than GD





Experiments

Experiments

Linear neural networks:

Experiments

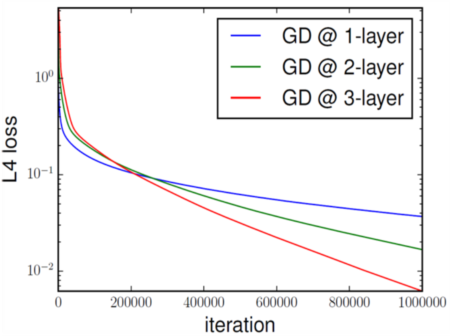
Linear neural networks:

Regression problem from UCI ML Repository ; ℓ_4 loss

Experiments

Linear neural networks:

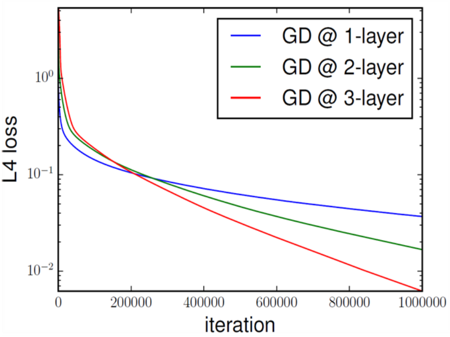
Regression problem from UCI ML Repository ; ℓ_4 loss



Experiments

Linear neural networks:

Regression problem from UCI ML Repository ; ℓ_4 loss

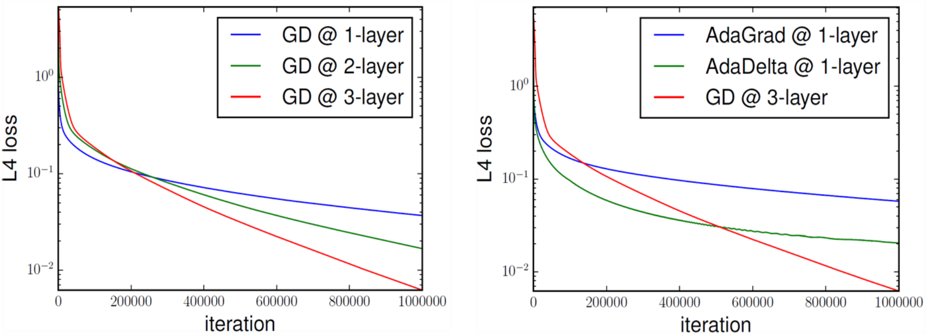


Depth can speed-up GD, even w/o any gain in expressiveness, and despite introducing non-convexity!

Experiments

Linear neural networks:

Regression problem from UCI ML Repository ; ℓ_4 loss

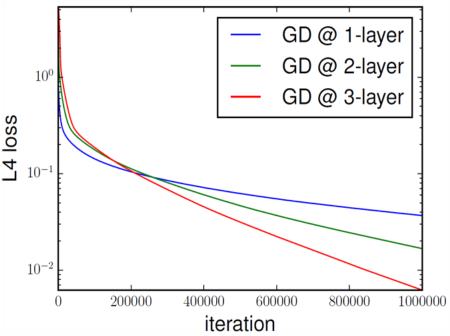


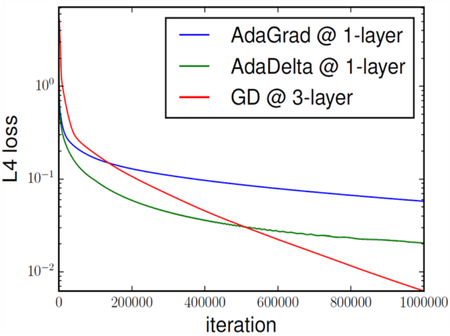
Depth can speed-up GD, even w/o any gain in expressiveness, and despite introducing non-convexity!

Experiments

Linear neural networks:

Regression problem from UCI ML Repository ; ℓ_4 loss





Depth can speed-up GD, even w/o any gain in expressiveness, and despite introducing non-convexity!

This speed-up can outperform popular acceleration methods designed for convex problems!

Experiments (cont')

Non-linear neural networks:

Experiments (cont')

Non-linear neural networks:

TensorFlow convolutional network tutorial for MNIST: [1]

- Arch: (conv $\rightarrow$ ReLU $\rightarrow$ max pool) x 2 $\rightarrow$ dense $\rightarrow$ ReLU $\rightarrow$ dense
- Training: stochastic GD w/momentum, dropout

[1] `https://github.com/tensorflow/models/tree/master/tutorials/image/mnist`

Experiments (cont')

Non-linear neural networks:

TensorFlow convolutional network tutorial for MNIST:[1]

- Arch: (conv $\rightarrow$ ReLU $\rightarrow$ max pool) x 2 $\rightarrow$ dense $\rightarrow$ ReLU $\rightarrow$ dense
- Training: stochastic GD w/momentum, dropout

We overparameterized by adding linear layer after each dense layer

[1] https://github.com/tensorflow/models/tree/master/tutorials/image/mnist

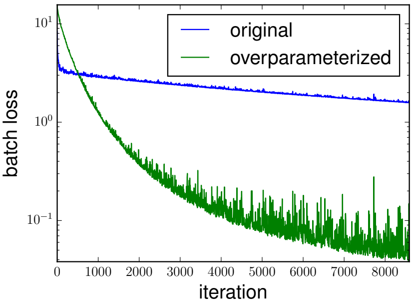
Experiments (cont')

Non-linear neural networks:

TensorFlow convolutional network tutorial for MNIST:[1]

- Arch: (conv $\rightarrow$ ReLU $\rightarrow$ max pool) x 2 $\rightarrow$ dense $\rightarrow$ ReLU $\rightarrow$ dense
- Training: stochastic GD w/momentum, dropout

We overparameterized by adding linear layer after each dense layer



[1] https://github.com/tensorflow/models/tree/master/tutorials/image/mnist

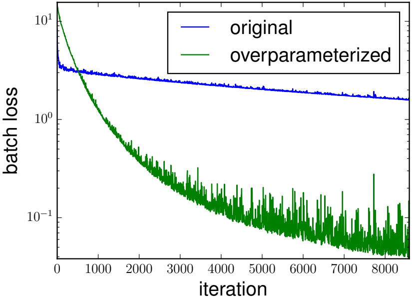
Experiments (cont')

Non-linear neural networks:

TensorFlow convolutional network tutorial for MNIST: [1]

- Arch: (conv → ReLU → max pool) x 2 → dense → ReLU → dense
- Training: stochastic GD w/momentum, dropout

We overparameterized by adding linear layer after each dense layer



Adding depth, w/o any gain in expressiveness, and only $+15\%$ in params, accelerated non-linear net by orders-of-magnitude!

[1] `https://github.com/tensorflow/models/tree/master/tutorials/image/mnist`

Outline

1. Deep Learning Theory: Expressiveness, Optimization and Generalization

2. Analyzing Optimization via Trajectories

3. Trajectories of Gradient Descent for Deep Linear Neural Networks
   - Convergence to Global Optimum
   - Acceleration by Depth

4. Conclusion

Recap

Recap

Understanding DL calls for addressing three fundamental Qs:

Expressiveness Optimization Generalization

Recap

Understanding DL calls for addressing three fundamental Qs:

Expressiveness   Optimization   Generalization

Optimization

Recap

Understanding DL calls for addressing three fundamental Qs:

Expressiveness   Optimization   Generalization

Optimization

Deep (≥ 3 layer) models can't be treated via geometry alone

$\implies$ specific optimizer trajectories should be taken into account

Recap

Understanding DL calls for addressing three fundamental Qs:

Expressiveness Optimization Generalization

Optimization

Deep (≥ 3 layer) models can't be treated via geometry alone

$\Longrightarrow$ specific optimizer trajectories should be taken into account

We **analyzed trajectories of GD over deep linear neural nets**:

Recap

Understanding DL calls for addressing three fundamental Qs:

Expressiveness Optimization Generalization

Optimization

Deep (≥ 3 layer) models can't be treated via geometry alone

$\implies$ specific optimizer trajectories should be taken into account

We **analyzed trajectories of GD over deep linear neural nets**:

- Derived **guarantee for convergence to global min at linear rate** (most general guarantee to date for GD efficiently training deep model)

Recap

Understanding DL calls for addressing three fundamental Qs:

Expressiveness   Optimization   Generalization

Optimization

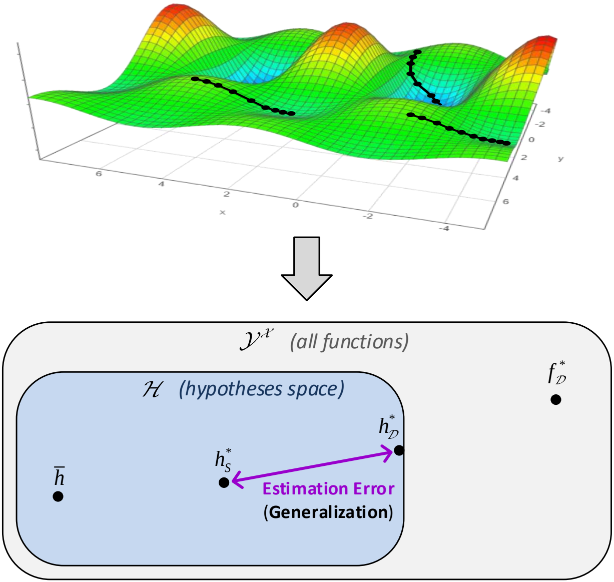
Deep (≥ 3 layer) models can't be treated via geometry alone

$\Longrightarrow$ specific optimizer trajectories should be taken into account

We **analyzed trajectories of GD over deep linear neural nets**:

- Derived **guarantee for convergence to global min at linear rate** (most general guarantee to date for GD efficiently training deep model)
- **Depth induces preconditioner that can accelerate convergence**, w/o any gain in expressiveness, and despite introducing non-convexity

Next Step: Analyzing Generalization via Trajectories



Outline

1. Deep Learning Theory: Expressiveness, Optimization and Generalization
2. Analyzing Optimization via Trajectories
3. Trajectories of Gradient Descent for Deep Linear Neural Networks
   - Convergence to Global Optimum
   - Acceleration by Depth
4. Conclusion

Thank You