Explicit and Implicit Inductive Bias in Deep Learning

Nati Srebro (TTIC)

Based on work with **Behnam Neyshabur** (TTIC→Google), **Suriya Gunasekar** (TTIC→MSR),
Ryota Tomioka (TTIC→MSR), Srinadh Bhojanapalli (TTIC→Google),
Blake Woodworth, Pedro Savarese, David McAllester (TTIC), Greg Ongie, Becca Willett (Chicago),
Daniel Soudry, Elad Hoffer, Mor Shpigel, Edward Moroshko, Itay Golan (Technion),
Shahar Azulai, Amir Globerson (Tel-Aviv), Ziwei Ji, Matus Telgarsky (UIUC),
Ashia Wilson, Becca Roelofs, Mitchel Stern, Ben Recht (Berkeley),
Russ Salakhutdinov (CMU), **Jason Lee**, Zhiyuan Li (Princeton), Yann LaCun (NYU/Facebook)

Plan

- What ~~is~~ do I mean by "Inductive Bias"?
- Inductive Bias in Deep Learning:
  The Role of Implicit Optimization Bias
- The "*complexity measure*" approach for understanding Deep Learning

(break)

- Examples of Identifying the Implicit Bias and "*complexity measure*"
  - Squared Loss vs Logistic Loss
  - Effect of initialization and other parameters
  - Explicit Regularization vs Implicit Bias
  - Can implicit bias be described in terms of a complexity measure?

- **Supervised Learning:** find $h: \mathcal{X} \to \mathcal{Y}$ with small *generalization error*

$$L(h) = \mathbb{E}_{(x,y)\sim\mathcal{D}}[loss(h(x); y)]$$

  based on samples S (hopefully $S \sim \mathcal{D}^m$) using learning rule:

$$A: S \mapsto h \quad \text{(i.e. } A: (\mathcal{X} \times \mathcal{Y})^* \to \mathcal{Y}^{\mathcal{X}} \text{)}$$

- **No Free Lunch:** For any learning rule, there exists a source $\mathcal{D}$ (i.e. reality), for which the learning rule yields expected error ½
- More formally for any A, m there exists $\mathcal{D}$ s.t. $\exists_{h^*} L(h^*) = 0$ but

$$\mathbb{E}_{S\sim\mathcal{D}^m}\left[L\big(A(S)\big)\right] \geq \frac{1}{2} - \frac{m}{2|\mathcal{X}|}$$

- **Inductive Bias:**
  - Some realities (sources $\mathcal{D}$) are less likely; design A to work well on more likely realities

    e.g., by preferring certain $y|x$ (i.e. $h(x)$) over others
  - Assumption or property of reality $\mathcal{D}$ under which A ensures good generalization error

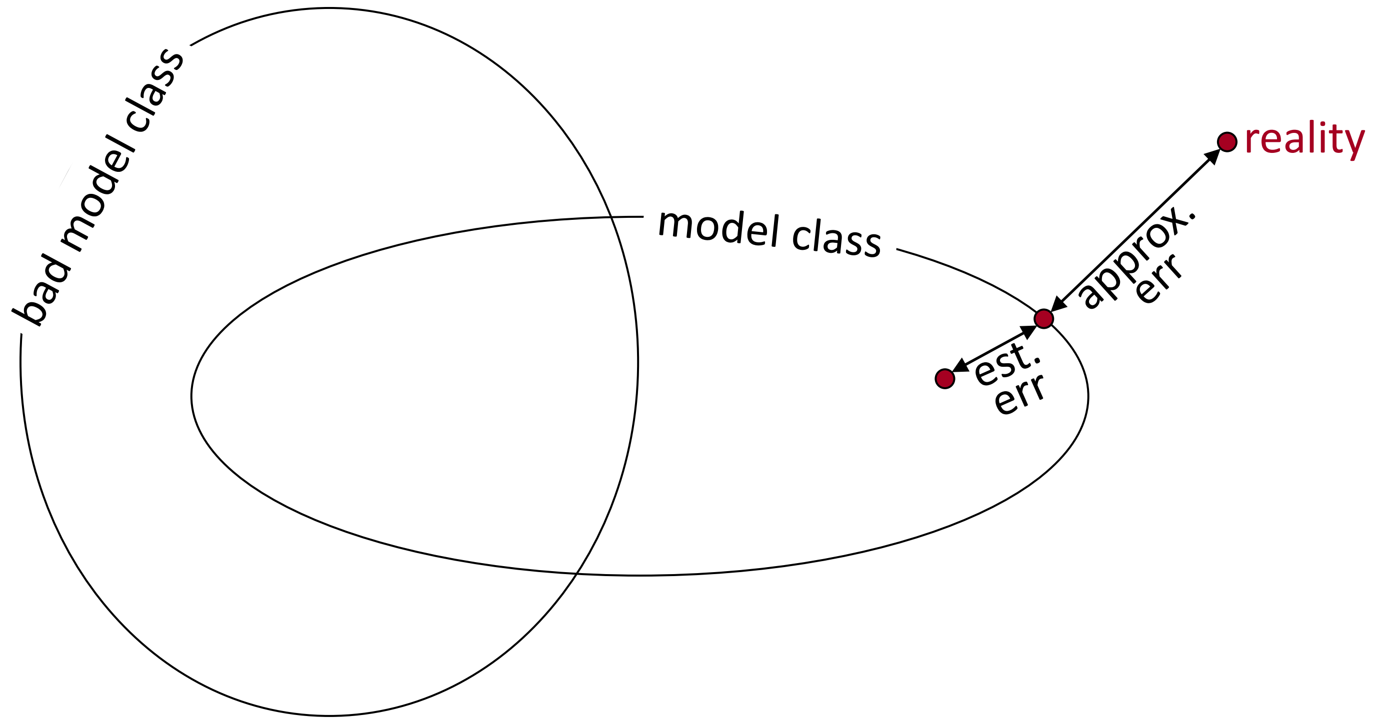
    e.g., $\exists h \in \mathcal{H}$ with low $L(h)$

    e.g., $\exists h$ with low "complexity" $c(h)$ and low $L(h)$

Flat Inductive Bias

- **"Flat" inductive bias**: $\exists h^* \in \mathcal{H}$ with low $L(h^*)$
- (Almost) optimal learning rule:
$$ERM_{\mathcal{H}}(S) = \hat{h} = \arg\min_{h \in \mathcal{H}} L_S(h)$$
- Guarantee (in expectation over $S \sim \mathcal{D}^m$):
$$L\big(ERM_{\mathcal{H}}(S)\big) \leq L(h^*) + \mathcal{R}_m(\mathcal{H}) \approx L(h^*) + \sqrt{\frac{capacity(\mathcal{H})}{m}}$$
$\Rightarrow$ can learn with $O(capacity(\mathcal{H}))$ samples
- E.g.
  - For binary loss, $capacity(\mathcal{H}) = VCdim(H)$
  - For linear predictors over d features, $capacity(\mathcal{H}) = d$
  - Usually with d parameters, $capacity(\mathcal{H}) \approx \tilde{O}(d)$
  - For linear predictors with $\|w\|_2 \leq B$, with logistic loss and normalized data: $capacity(\mathcal{H}) = B^2$

Machine Learning



- We want model classes (hypothesis classes) that:
  - Are expressive enough to capture reality well
  - Have small enough capacity to allow generalization

Complexity Measure as Inductive Bias

- **Complexity measure**: mapping $c: \mathcal{Y}^{\mathcal{X}} \to [0, \infty]$
- Associated inductive bias: $\exists h^*$ with small $c(h^*)$ and small $L(h^*)$
- Learning rule: $SRM_{\mathcal{H}}(S) = \arg\min \;\; L(h) \; , \; c(h)$

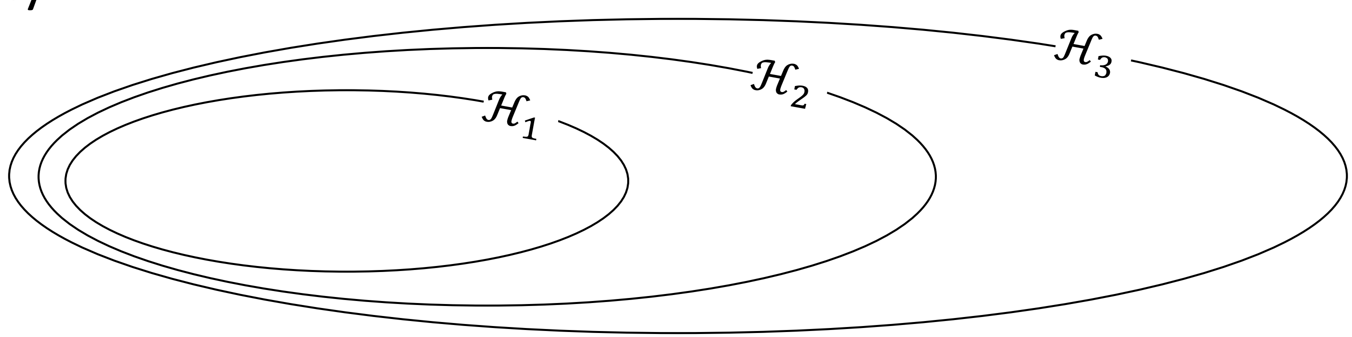
  e.g. $\arg\min L(h) + \lambda\, c(h)$ or $\arg\min L(h)$ s.t. $c(h) \le B$

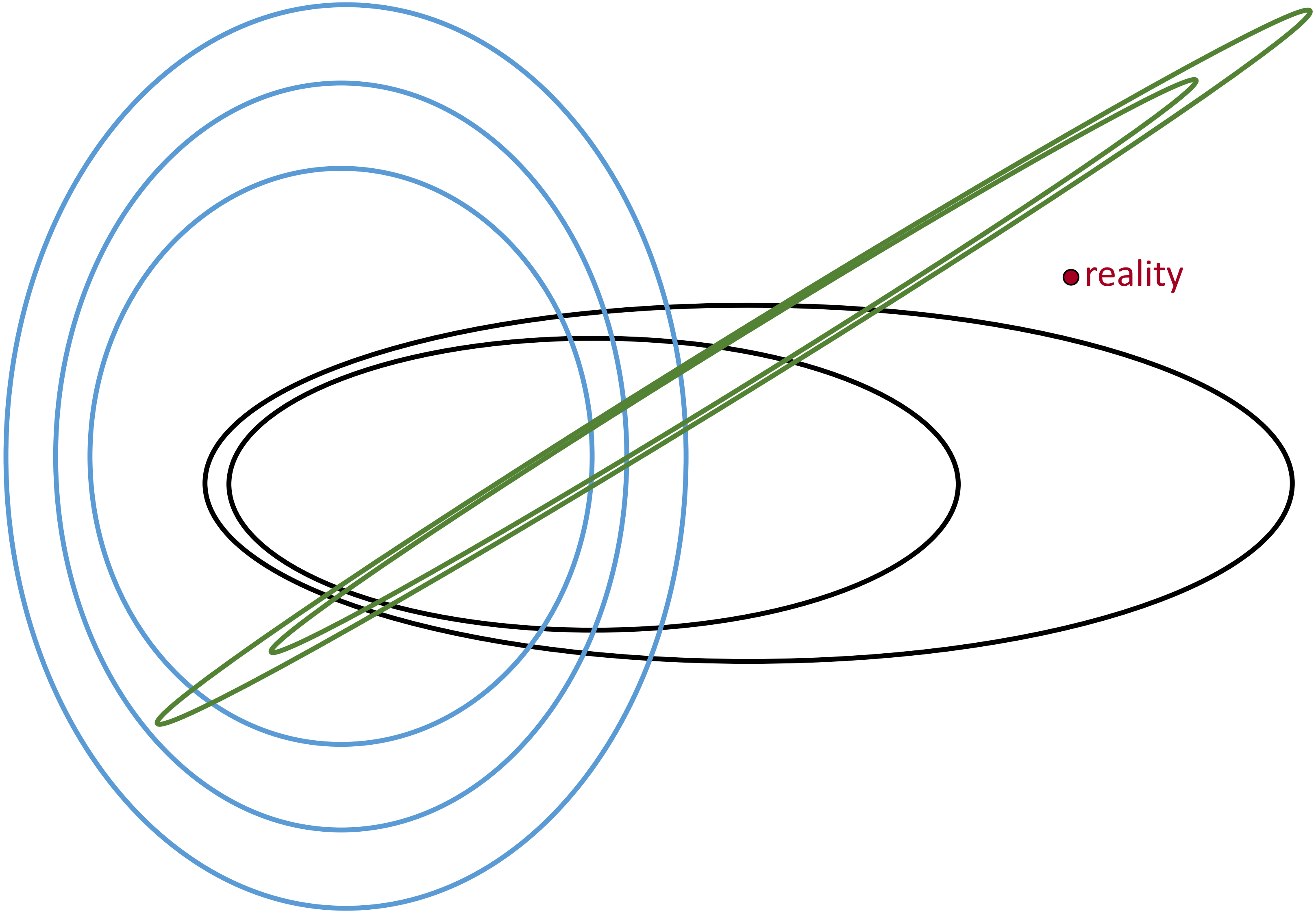
  and choose λ or B using cross-validation
- Guarantee:

$\mathcal{H}_B = \{h | c(h) \le B\}$

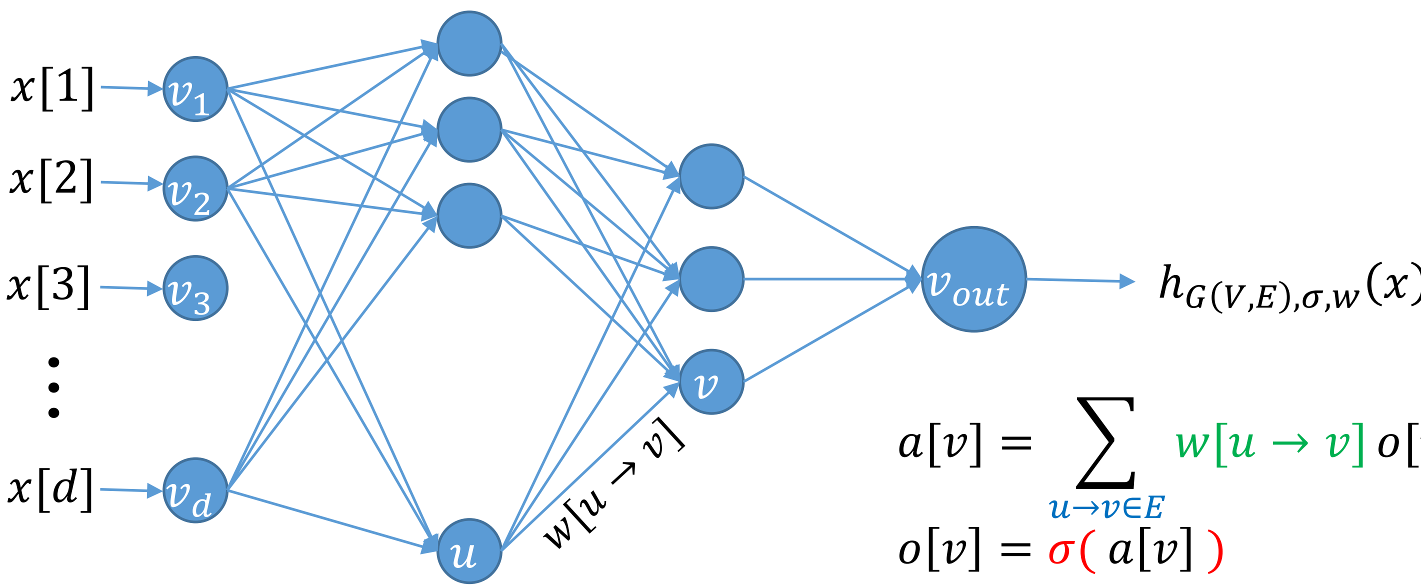
$$L\big(SRM_{\mathcal{H}}(S)\big) \leq\approx L(h^*) + \sqrt{\frac{capacity\big(\mathcal{H}_{c(h^*)}\big)}{m}}$$

- E.g.:
  - Degree of poly
  - Sparsity
  - $||w||$




reality

Feed-Forward Neural Networks (The Multilayer Perceptron)

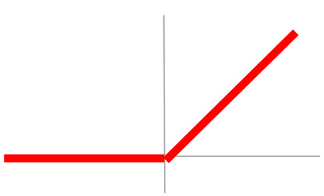


$$a[v] = \sum_{u \to v \in E} w[u \to v]\, o[u]$$

$$o[v] = \sigma(\, a[v]\,)$$

Architecture:

- Directed Acyclic Graph G(V,E). Units (neurons) indexed by vertices in V.
  - "Input Units" $v_1 \ldots v_d \in V$, with no incoming edges and $o[v_i] = x[i]$
  - "Output Unit" $v_{out} \in V$, $h_w(x) = o[v_{out}]$
- "Activation Function" $\sigma: \mathbb{R} \to \mathbb{R}$. E.g. $\sigma_{RELU}(z) = [z]_+$



Parameters:

- Weight $w[u \to v]$ for each edge $u \to v \in E$

Feed Forward Neural Networks

- Fix architecture (connection graph $G(V,E)$, transfer σ)

$$\mathcal{H}_{G(V,E),\sigma} = \{\, f_w(x) = output\ of\ net\ with\ weights\ w \,\}$$

- Capacity / Generalization ability / Sample Complexity
  - $\tilde{O}(|E|)$ (number of edges, i.e. number of weights)
    (with threshold σ, or with RELU and finite precision; RELU with inf precision: $\tilde{\Theta}(|E| \cdot depth)$)
- Expressive Power / Approximation
  - Any continuous function with huge network
  - Lots of interesting things naturally with small networks
  - **Any time T computable function with network of size $\tilde{O}(T)$**

Free Lunches

- **ML as an Engineering Paradigm**: Use data and examples, instead of expert knowledge and tedious programming, to automatically create efficient systems that perform complex tasks
- We only care about $\{h | h \text{ is an efficient system}\}$
- **Free Lunch**: $\boldsymbol{TIME_T} = \{h | h \text{ comp. in time } T\}$ has capacity $O(T)$ and hence learnable with $O(T)$ samples, e.g. using ERM
- Even better: $\boldsymbol{PROG_T} = \{\text{program of length T}\}$ has capacity $O(T)$
- Problem: ERM for above is not computable!
- Modified ERM for $\boldsymbol{TIME_T}$ (truncating exec. time) is NP-complete
- P=NP ➔ **Universal Learning is possible! (Free Lunch)**
- Crypto is possible (one-way functions exist)
  ➔ No poly-time learning algorithm for $\boldsymbol{TIME_T}$
  (that is: no poly-time A and uses $poly(T)$ samples s.t. if $\exists h^* \in TIME_T$ with $L(h^*) = 0$ then $\mathbb{E}\left[L\left(A(S)\right)\right] \leq 0.4$)

No Free (Computational) Lunch

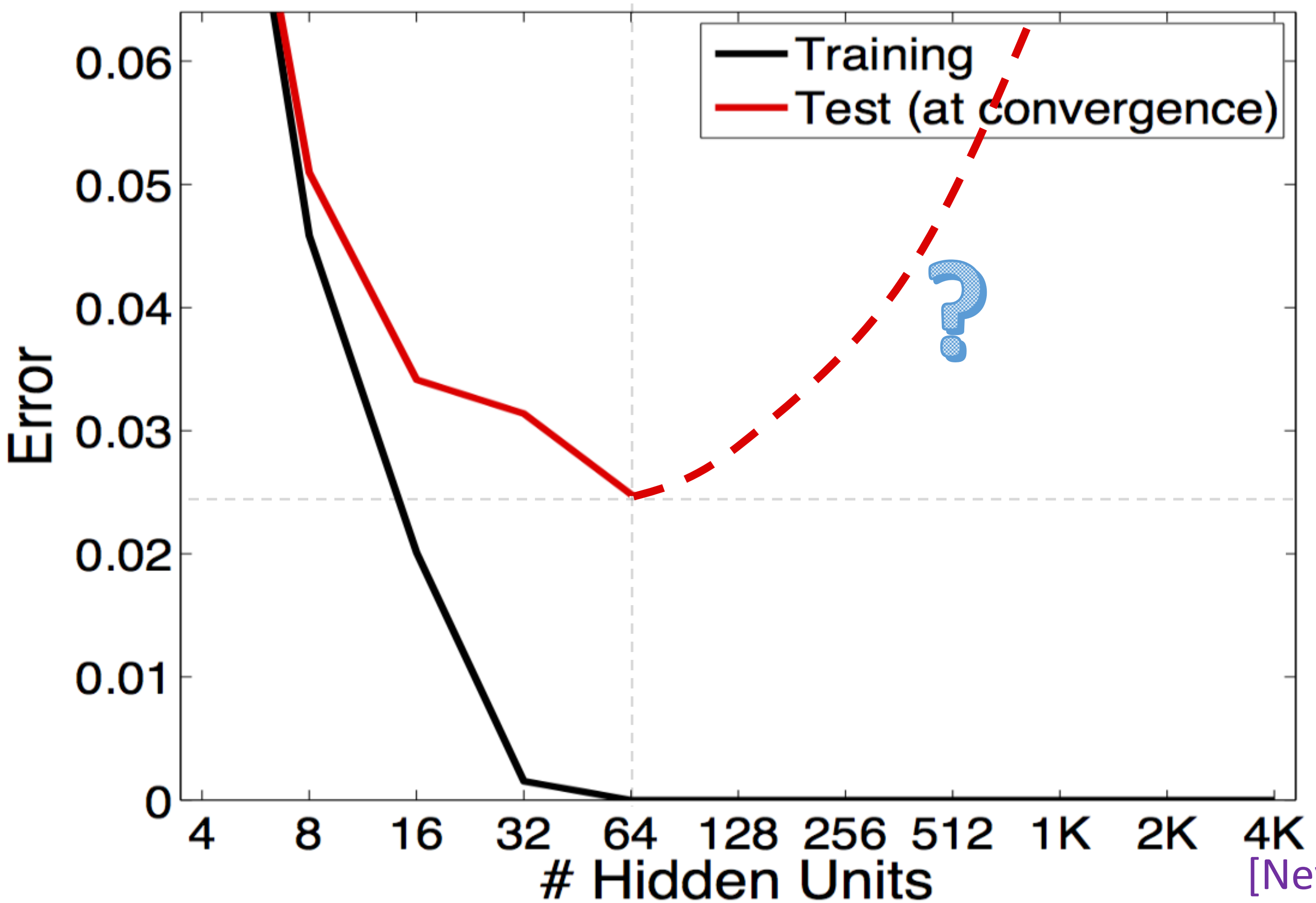
- **Statistical No-Free Lunch**: For any learning rule A, there exists a source $\mathcal{D}$ (i.e. reality), s.t. $\exists h^*$ with $L(h^*) = 0$ but $\mathbb{E}\left[L\left(A(S)\right)\right] \approx \frac{1}{2}$.
- **Cheating Free Lunch**: There exists A, s.t. for any reality $\mathcal{D}$ and any **efficiently computable $\boldsymbol{h}^*$**, A learns a predictor almost as good as h^* (with #samples=O(runtime of h^*), but *a lot* of time).
- **Computational No-Free Lunch**: For every **computationally efficient** learning ***algorithm*** $\boldsymbol{A}$, there is a reality $\mathcal{D}$ s.t. there is some comp. efficient (poly-time) h^* with $L(h^*) = 0$ but $\mathbb{E}\left[L\left(A(S)\right)\right] \approx \frac{1}{2}$.
- **Inductive Bias**: Assumption or property of reality $\mathcal{D}$ under which a learning ***algorithm*** A **runs efficiently** and ensures **good generalization error**.
- $\mathcal{H}$ or $c(h)$ are *not* sufficient inductive bias if ERM/SRM not efficiently implementable, or implementation doesn't always work (runs quickly and returns actual ERM/SRM).

Feed Forward Neural Networks

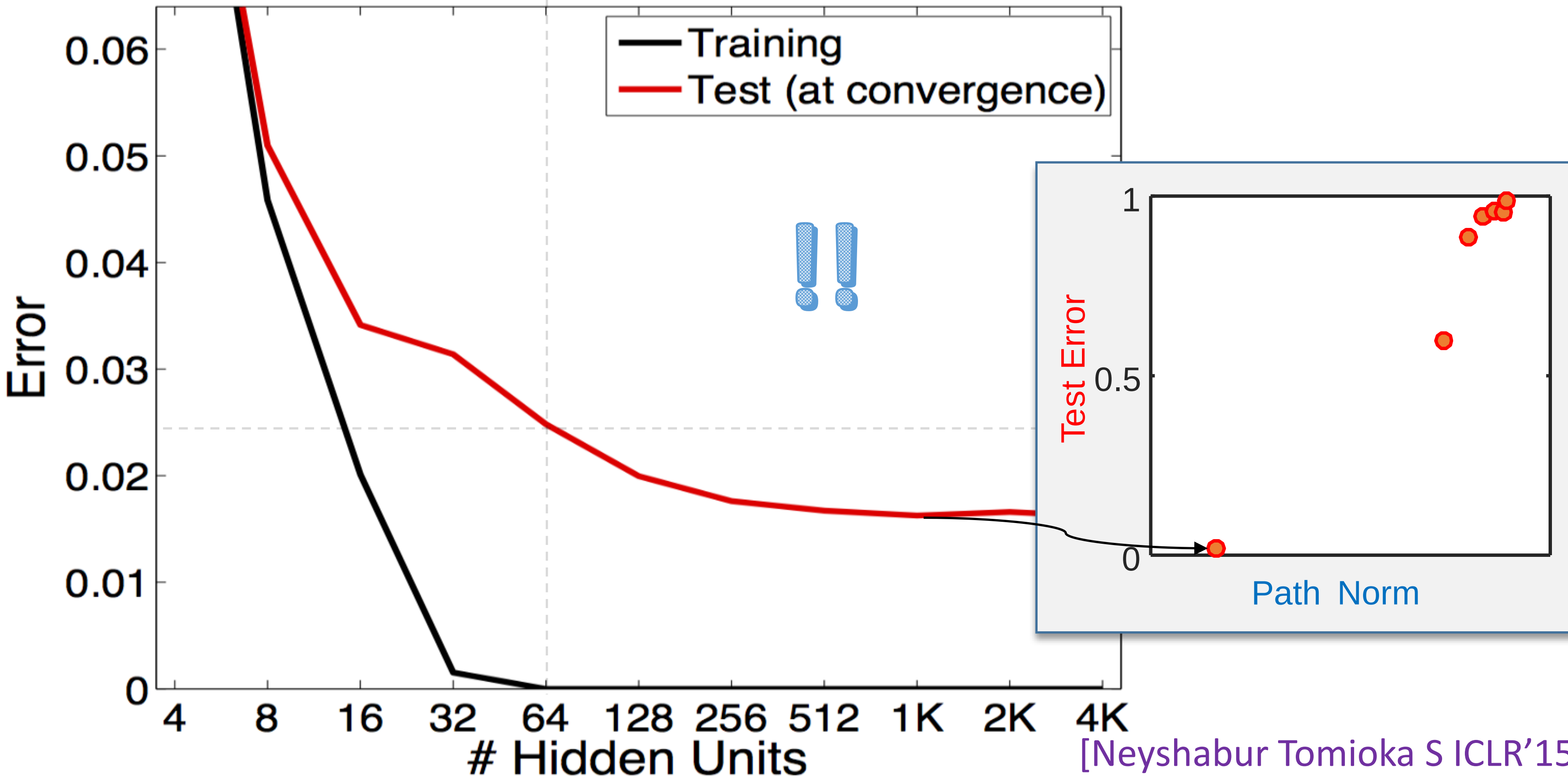
- Fix architecture (connection graph $G(V,E)$, transfer σ)

$$\mathcal{H}_{G(V,E),\sigma} = \{\, f_w(x) = output\ of\ net\ with\ weights\ w \,\}$$

- Capacity / Generalization ability / Sample Complexity
  - $\tilde{\boldsymbol{O}}(|\boldsymbol{E}|)$ (number of edges, i.e. number of weights)
    (with threshold σ, or with RELU and finite precision; RELU with inf precision: $\tilde{\Theta}(|E| \cdot depth)$)
- Expressive Power / Approximation
  - Any continuous function with huge network
  - Lots of interesting things naturally with small networks
  - **Any time T computable function with network of size $\tilde{\boldsymbol{O}}(\boldsymbol{T})$**
- Computation / Optimization
  - Even if function exactly representable with single hidden layer with $\Theta(\log d)$ units, even with no noise, and even if we allow a much larger network when learning: no poly-time algorithm always works
    [Kearns Valiant 94; Klivans Sherstov 06; Daniely Linial Shalev-Shwartz '14]
  - Magic property of reality that makes local search "work"



[Neyshabur Tomioka S ICLR'15]

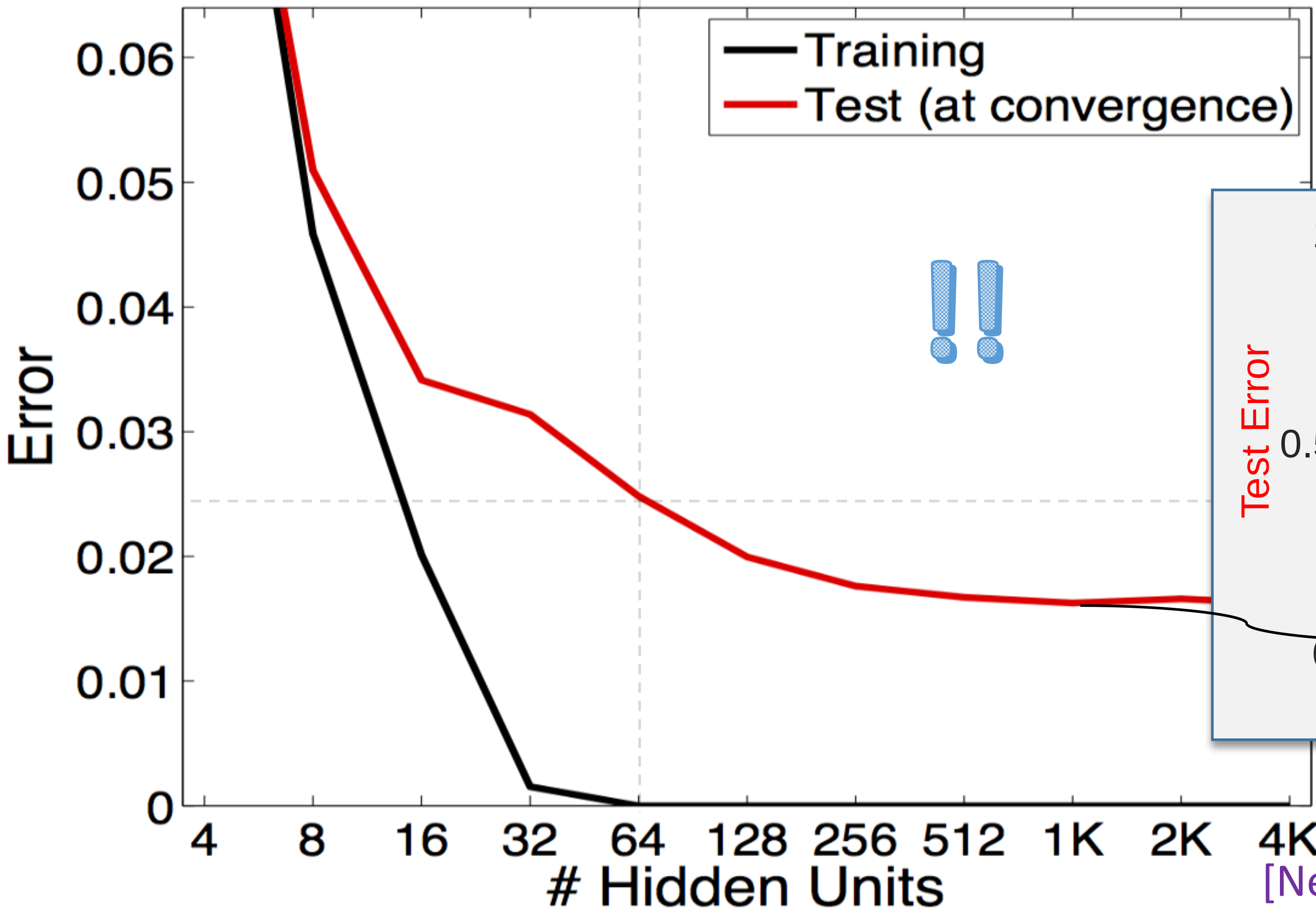


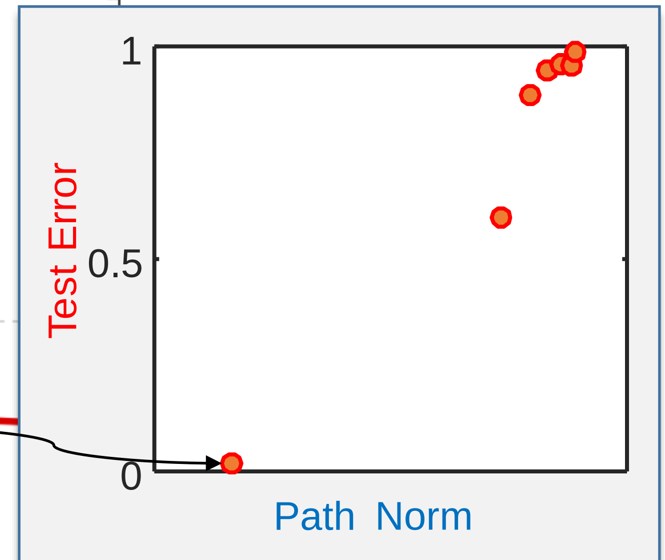
[Neyshabur Tomioka S ICLR'15]

For valid generalization, the size of the weights is more important than the size of the network

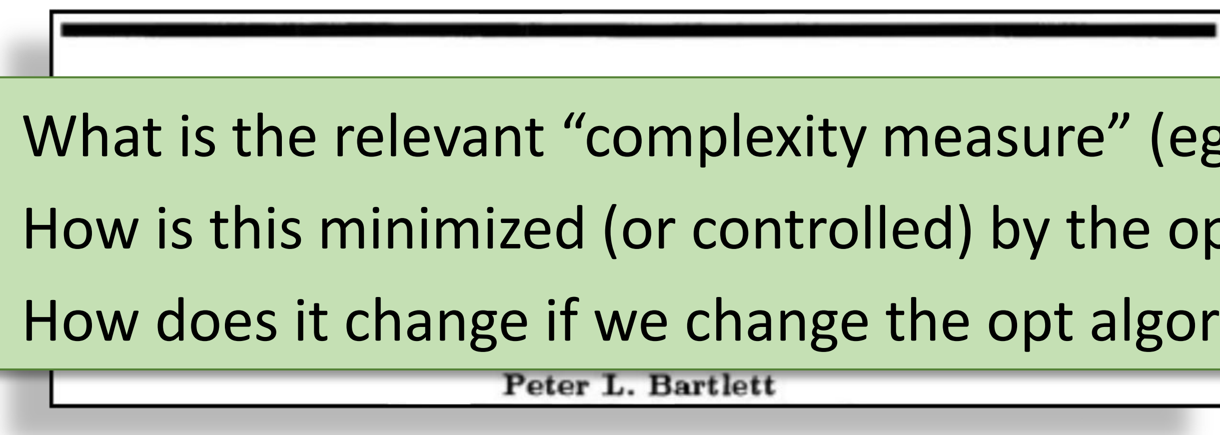
1997

Peter L. Bartlett

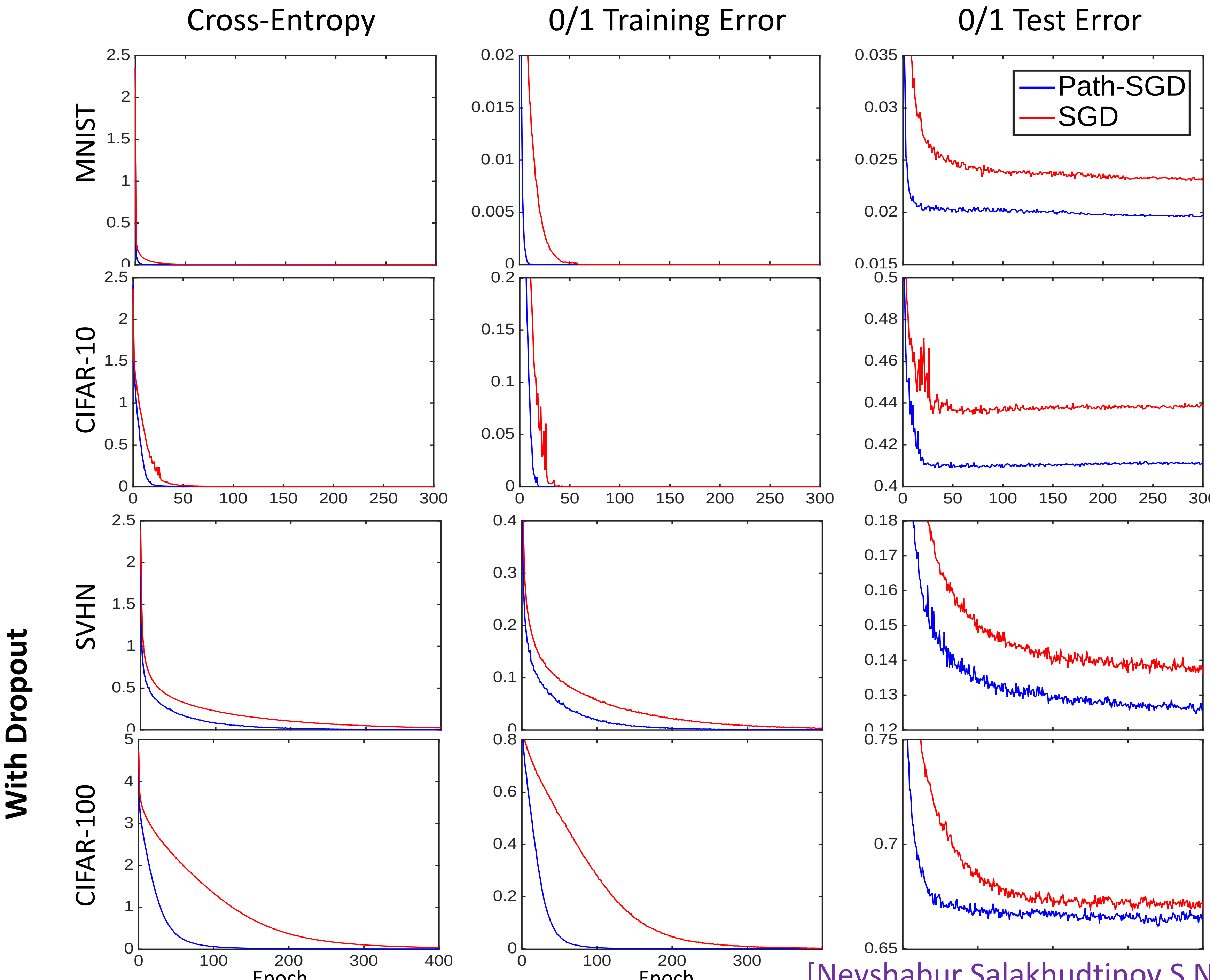




[Neyshabur Tomioka S ICLR'15]

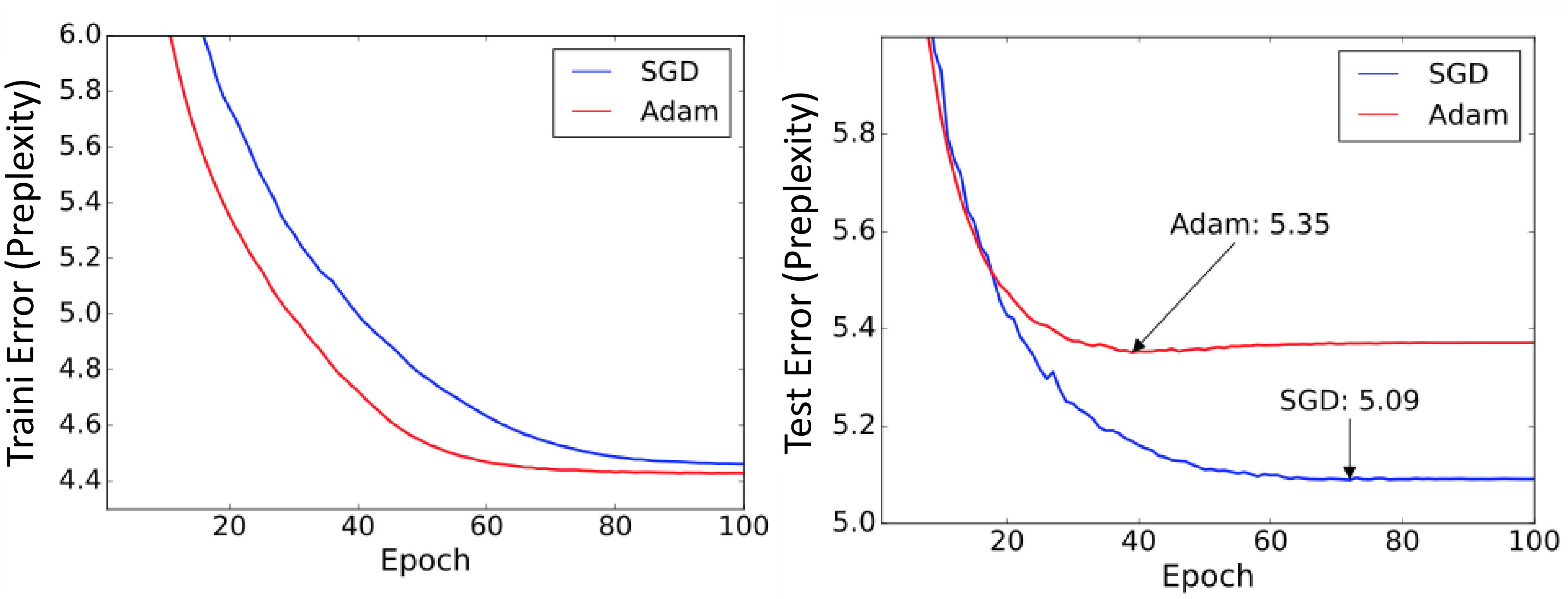


- What is the relevant "complexity measure" (eg norm)?
- How is this minimized (or controlled) by the opt algorithm?
- How does it change if we change the opt algorithm?



[Neyshabur Salakhudtinov S NIPS'15]

SGD vs ADAM



Results on Penn Treebank using 3-layer LSTM

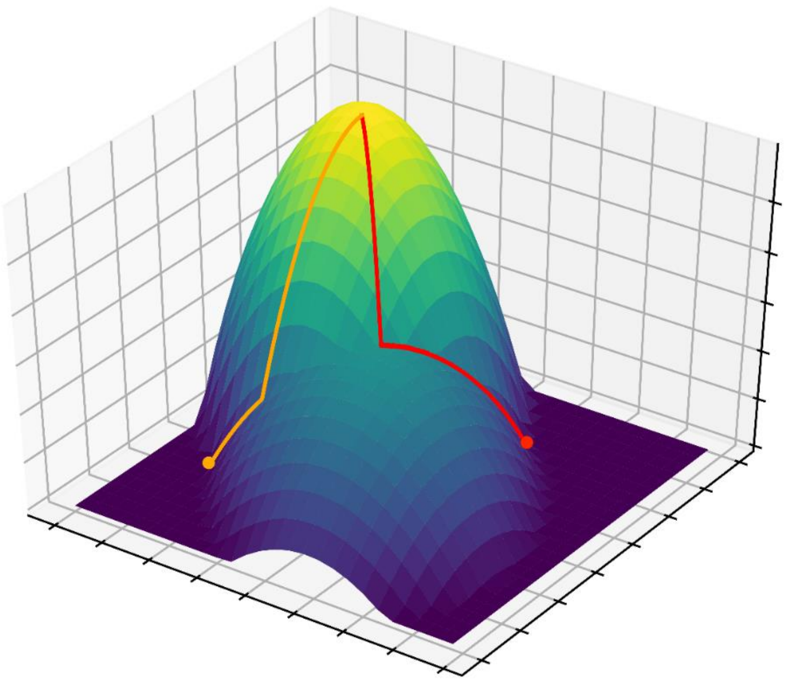
[Wilson Roelofs Stern S Recht, “The Marginal Value of Adaptive Gradient Methods in Machine Learning”, NIPS’17]

Different optimization algorithm

➔ Different bias in optimum reached

➔ Different Inductive bias

➔ Different generalization properties



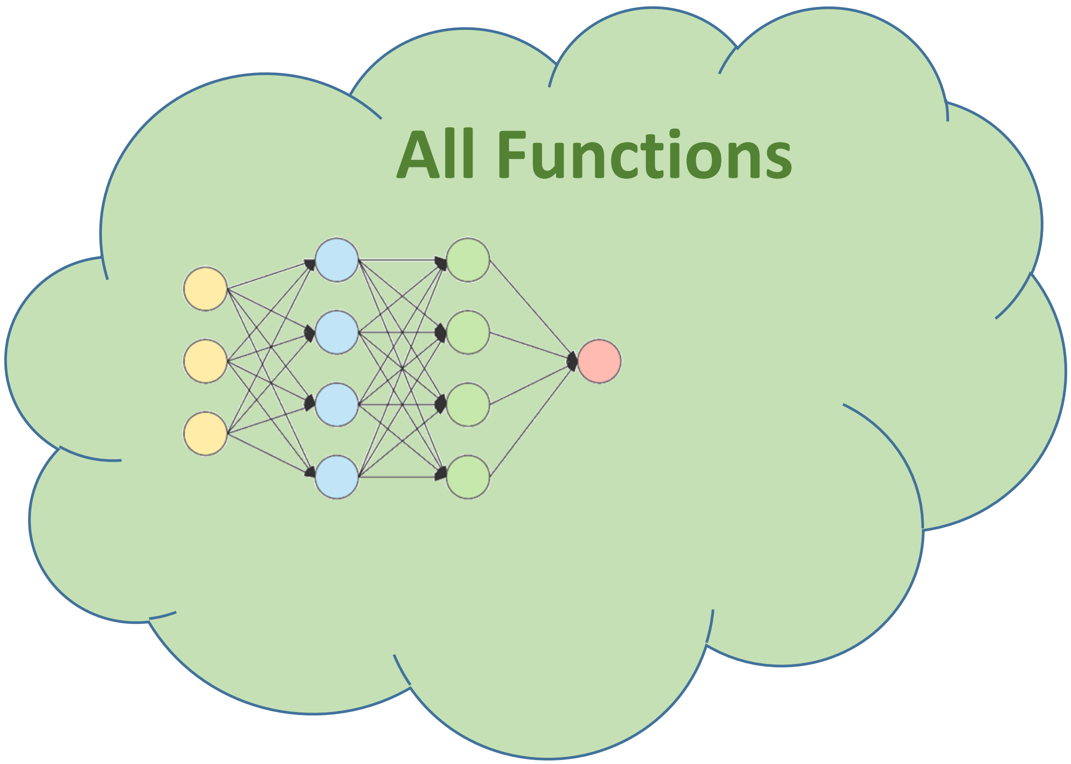
Need to understand optimization alg. not just as reaching ***some*** (global) optimum, but as reaching a ***specific*** optimum

Different optimization algorithm

➔ Different bias in optimum reached

➔ Different Inductive bias

➔ Different generalization properties



Need to understand optimization alg. not just as reaching ***some*** (global) optimum, but as reaching a ***specific*** optimum

The Deep Recurrent Residual Boosting Machine

Joe Flow, DeepFace Labs

Section 1: Introduction

We suggest a new amazing architecture and loss function that is great for learning. All you have to do to learn is fit the model on your training data

Section 2: Learning Contribution: our model

The model class h_w is amazing. **Our learning method is:**

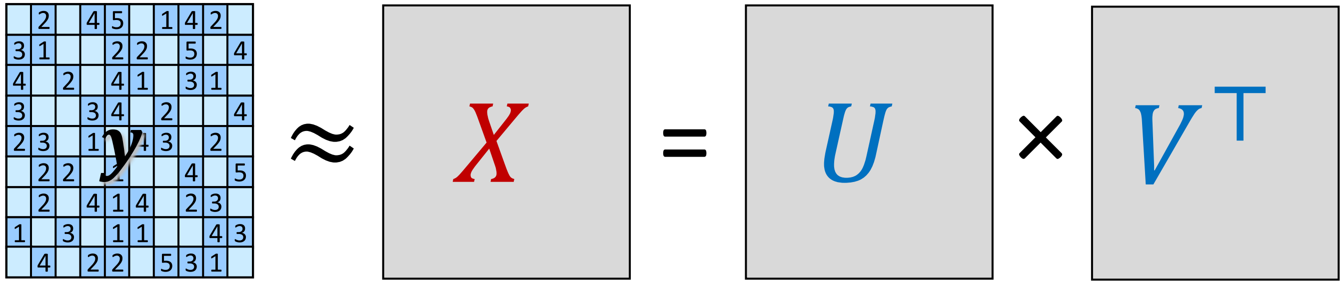
$$\boldsymbol{\arg\min_{w} \frac{1}{m} \sum_{i=1}^{m} loss(h_w(x); y)} \qquad (*)$$

Section 3: Optimization

This is how we solve the optimization problem (*): […]

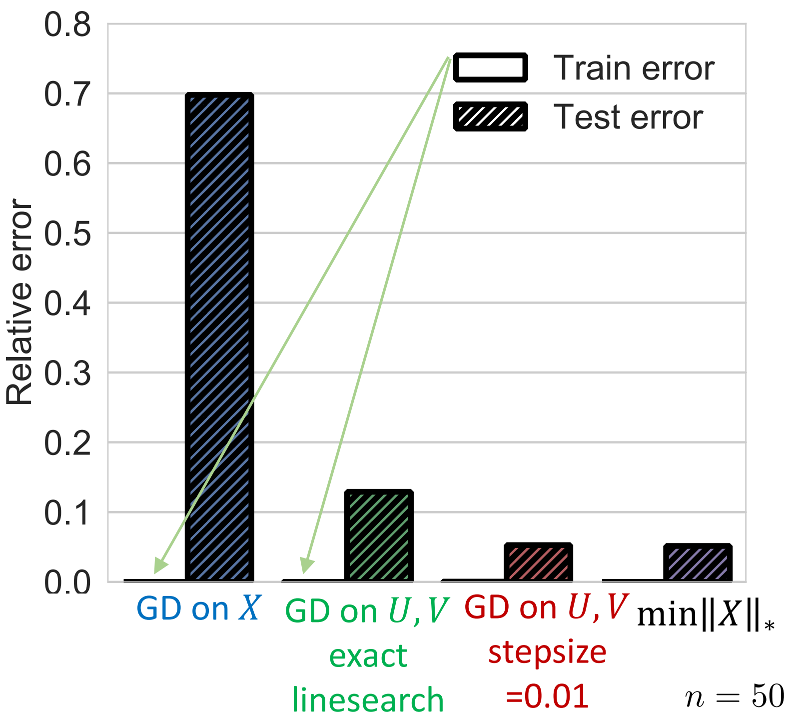
Section 4: Experiments

It works!



$$\min_{X\in\mathbb{R}^{n\times n}} \|observed(X) - y\|_2^2 \equiv \min_{U,V\in\mathbb{R}^{n\times n}} \|observed(UV^\top) - y\|_2^2$$

- Underdetermined non-sensical problem, lots of useless global min
- Since U, V full dim, no constraint on X, all the same non-sense global min



Grad Descent on $U, V \overset{???}{\rightarrow}$ **min $\|X\|_*$ solution**

(with inf. small stepsize and initialization)

→ good generalization if Y (aprox) low rank

[Gunasekar Woodworth Bhojanapalli Neyshabur S 2017]

When $y = \langle \boldsymbol{A}_i, W^* \rangle$, W^* low rank, $\boldsymbol{A}_i$ RIP

[Yuanzhi Li, Hongyang Zhang and Tengyu Ma 2018]

Not always **min $\|X\|_*$** !

[Zhiyuan Li, Yuping Luo, Kaifeng Lyu ICLR 2021]

$n = 50,\ m = 300,\ A_i$ iid Gaussian, X^* rank-2 ground truth

$y = \mathcal{A}(X^*) + \mathcal{N}(0, 10^{-3}),\ y_{\text{test}} = \mathcal{A}_{\text{test}}(X^*) + \mathcal{N}(0, 10^{-3})$

Different optimization algorithm

➔ Different bias in optimum reached

➔ Different Inductive bias

➔ Different generalization properties

All Matrices

(all functions $i \times j \to y_{ij}$)

Need to understand optimization alg. not just as reaching ***some*** (global) optimum, but as reaching a ***specific*** optimum

Deep Learning

- Expressive Power
  - We are searching over the space of all functions…
    … but with what bias? What (implicit) assumptions?
  - How does this bias look? Is it reasonable/sensible?
- Capacity / Generalization ability / Sample Complexity
  - What's the true complexity measure (inductive bias)?
  - How does it control generalization?
- Computation / Optimization
  - How and where does optimization bias us?
    Under what conditions?

Ultimate Question: What is the true Inductive Bias? What makes reality *efficiently* learnable by fitting a (huge) neural net with a specific algorithm?

The "complexity measure" approach

Identify $c(h)$ s.t.

- Optimization algorithm biases towards low $c(h)$
- $\mathcal{H}_{c(reality)} = \{h | c(h) \leq c(reality)\}$ has low capacity
- Reality is well explained by low $c(h)$

- Mathematical questions:
  - **What is the bias of optimization algorithms?**
  - What is the capacity (≡sample complexity) of the sublevel sets $\mathcal{H}_c$?
- Question about reality (scientific Q?): does it have low $c(h)$?

Simple Example: Least Squares

- Consider an under-constraint least-squares problem ($n < m$):
$$\min_{w \in \mathbb{R}^n} \| Aw - b \|^2$$
$A \in \mathbb{R}^{m \times n}$

- Claim: Gradient Descent (or SGD, or conjugate gradient descent, or BFGS) converges to the least norm solution
$$\min_{Aw=b} \|w\|_2$$

➢ Proof: iterates always spanned by rows of A (more details soon)

Implicit Bias in Least Squared

$$\min \| Aw - b \|^2$$

- Gradient Descent (+Momentum) on w

  ➔ $\min_{Aw=b} \|w\|_2$

- Gradient Descent on factorization $W = UV$

  ➔ $\min_{A(W)=b} \|W\|_*$ with stepsize↘ 0 and init ↘ 0, only in special cases (commutative measurements; or incoherent problems)

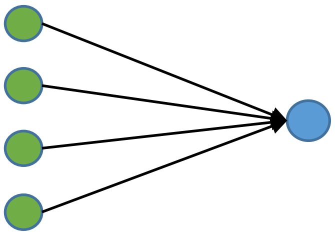
- AdaGrad on w

  ➔ in some special cases $\min_{Aw=b} \|w\|_\infty$, but not always, and it depends on stepsize, adaptation parameters, momentum

- Coordinate Descent (steepest descent w.r.t. $\|w\|_1$)

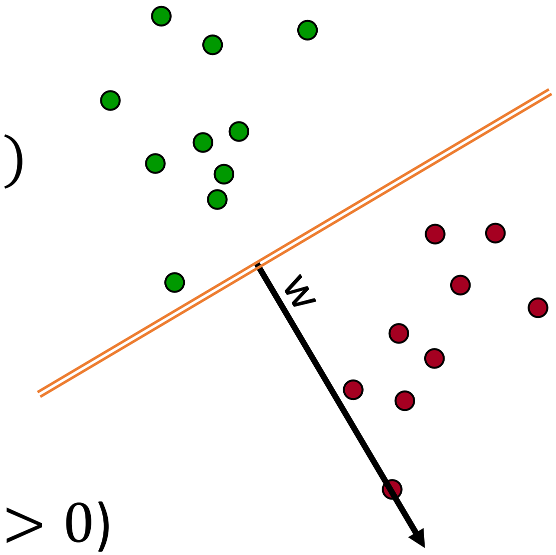
  ➔ Related to, but not quite $\min_{Aw=b} \|w\|_1$ (Lasso) (with stepsize↘ 0 and particular tie-breaking ≈ LARS)

Implicit Bias in Logistic Regression



$$\arg\min_{w\in\mathbb{R}^n} \mathcal{L}(w) = \sum_{i=1}^{m} \ell(y_i\langle w, x_i\rangle)$$

$$\ell(z) = \log(1+e^{-z})$$



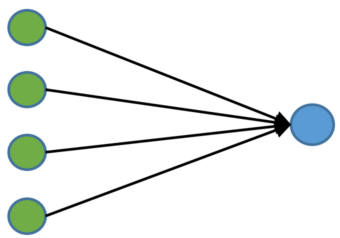
- Data $\{(x_i, y_i)\}_{i=1}^{m}$ linearly separable ($\exists_w \forall_i y_i\langle w, x_i\rangle > 0$)
- Where does gradient descent converge?

$$w(t) = w(t) - \eta\nabla\mathcal{L}(w(t))$$

  - $\inf \mathcal{L}(w) = 0$, but minima unattainable
  - GD diverges to infinity: $w(t) \to \infty$, $\mathcal{L}\big(w(t)\big) \to 0$
- **In what direction?** What does $\frac{w(t)}{\|w(t)\|}$ converge to?

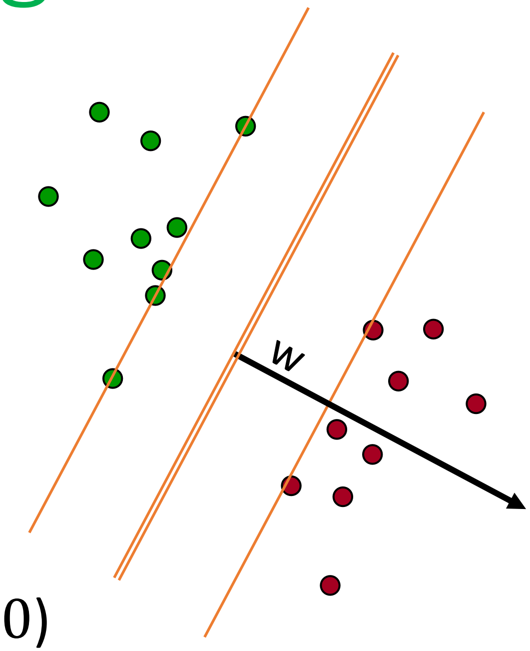
[Soudry Hoffer S 2017] based on [Telgarsky 2013 "Margins, shrinkage, and boosting"]

Implicit Bias in Logistic Regression



$$\arg\min_{w\in\mathbb{R}^n} \mathcal{L}(w) = \sum_{i=1}^{m} \ell(y_i\langle w, x_i\rangle)$$

$$\ell(z) = \log(1+e^{-z})$$



- Data $\{(x_i, y_i)\}_{i=1}^{m}$ linearly separable ($\exists_w \forall_i y_i\langle w, x_i\rangle > 0$)
- Where does gradient descent converge?

$$w(t) = w(t) - \eta\nabla\mathcal{L}(w(t))$$

  - $\inf \mathcal{L}(w) = 0$, but minima unattainable
  - GD diverges to infinity: $w(t) \to \infty$, $\mathcal{L}\big(w(t)\big) \to 0$
- **In what direction?** What does $\frac{w(t)}{\|w(t)\|}$ converge to?
- **<u>Theorem</u>**: $\frac{w(t)}{\|w(t)\|_2} \to \frac{\hat{w}}{\|\hat{w}\|_2}$ $\quad \hat{w} = \arg\min \|w\|_2 \;\; s.t. \forall_i y_i\langle w, x_i\rangle \geq 1$

[Soudry Hoffer S 2017] based on [Telgarsky 2013 "Margins, shrinkage, and boosting"]

How Fast is the Margin Maximized?

Convergence to the max margin $\hat{w}$: *

$$\left\| \frac{w(t)}{\|w(t)\|} - \frac{\hat{w}}{\|\hat{w}\|} \right\| = O\left(\frac{1}{\log t}\right)$$

Convergence of the margin itself:

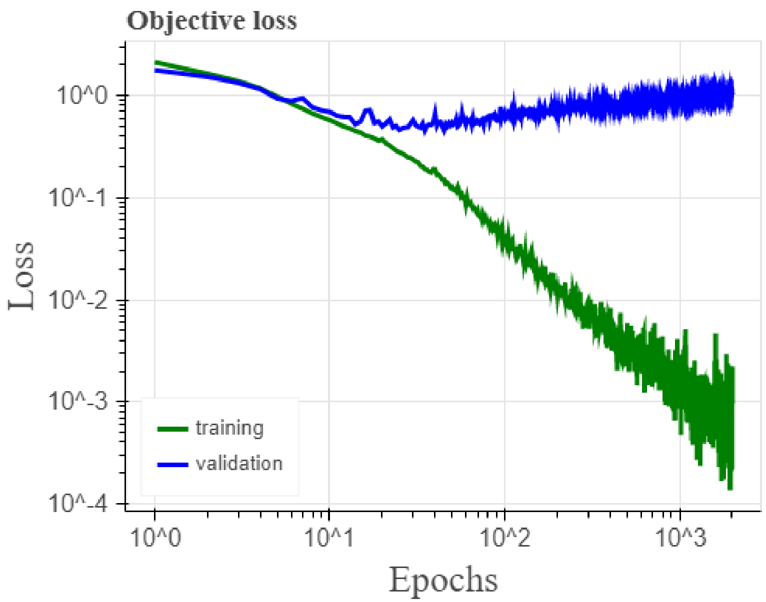
$$\max_{\|w\| \leq 1} \min_i y_i \langle w, x_i \rangle - \min_i y_i \left\langle \frac{w(t)}{\|w(t)\|}, x_i \right\rangle = O\left(\frac{1}{\log t}\right)$$

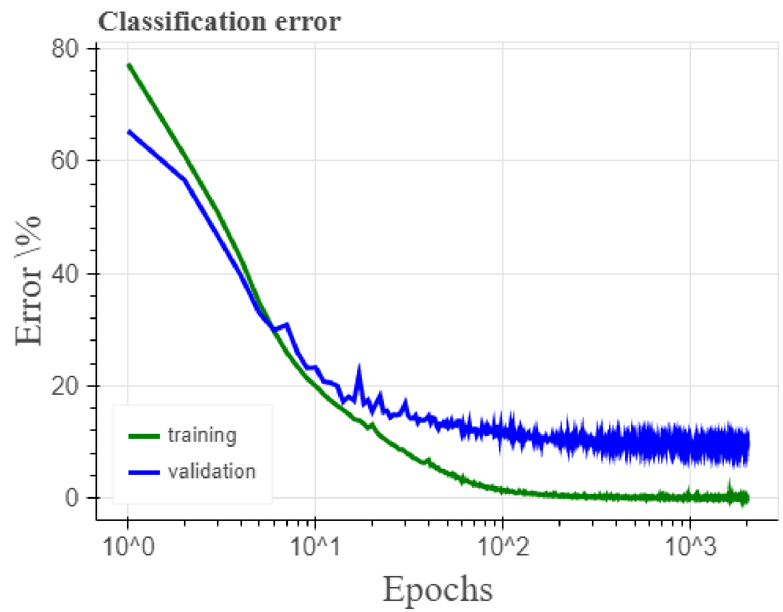
Contrast with convergence of the loss:

$$\mathcal{L}\big(w(t)\big) = O\left(\frac{1}{t}\right)$$

➔Even after we get extremely small loss, need to continue optimizing in order to maximize margin

*For data in general position. With degenerate data, $O(\log \log t \,/\, \log t)$





Epoch	50	100	200	400	2000	4000
L_2 norm	13.6	16.5	19.6	20.3	25.9	27.54
Train loss	0.1	0.03	0.02	0.002	10^{-4}	$3 \cdot 10^{-5}$
Train error	4%	1.2%	0.6%	0.07%	0%	0%
Validation loss	0.52	0.55	0.77	0.77	1.01	1.18
Validation error	12.4%	10.4%	11.1%	9.1%	8.92%	8.9%

Training a conv net using SGD+momentum on CFAIR10

Other Objectives and Opt Methods

- Single linear unit, logistic loss
  → hard margin SVM solution (regardless of init, stepsize)
- Multi-class problems with softmax loss
  → multiclass SVM solution (regardless of init, stepsize)
- Steepest Descent w.r.t. $\|w\|$

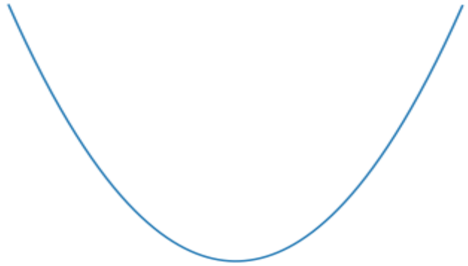
  → $\arg\min\|w\| \;\; s.t. \forall_i y_i\langle w, x_i\rangle \geq 1$ (regardless of init, stepsize)
- Coordinate Descent

  → $\arg\min\|w\|_1 \;\; s.t. \forall_i y_i\langle w, x_i\rangle \geq 1$ (regardless of init, stepsize)
- Matrix factorization problems $\mathcal{L}(U,V) = \sum_i \ell(\langle A_i, UV^\top\rangle)$, including 1-bit matrix completion

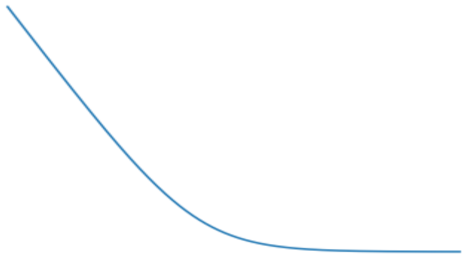
  → $\arg\min\|W\|_{tr} \;\; s.t. \langle A_i, W\rangle \geq 1$ (regardless of init)

Different Asymptotics

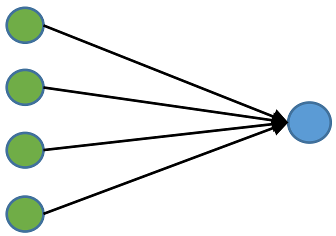
- For least squares (or any other loss with attainable minimum):
  - w_∞ depends on initial point w_0 and stepsize η
  - To get clean characterization, need to take $\eta \to 0$
  - If 0 is a saddle point, need to take $w_0 \to 0$



- For monotone decreasing loss (eg logistic)
  - w_∞ does NOT depend on initial w_0 and stepsize η
  - Don't need $\eta \to 0$ and $w_0 \to 0$
  - What happens at the beginning doesn't effect w_∞



Single Overparametrized Linear Unit



Train single unit with SGD using logistic ("cross entropy") loss

→ **Hard Margin SVM predictor**

$$w(\infty) \propto \arg\min \|w\|_2 \;\; s.t. \forall_i y_i \langle w, x_i \rangle \geq 1$$

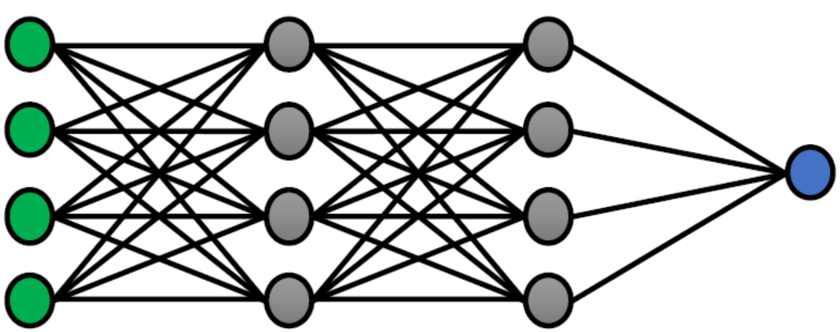
Even More Overparameterization: Deep Linear Networks

Network implements a linear mapping:

$$f_w(x) = \langle \beta_w, x \rangle$$

Training: same opt. problem as logistic regression:

$$\min_w \mathcal{L}(f_w) \equiv \min_\beta \mathcal{L}(x \mapsto \langle \beta, x \rangle)$$

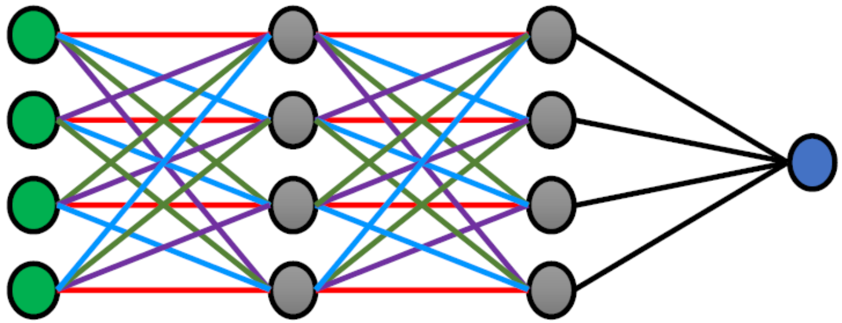


Train w with SGD

→ **Hard Margin SVM predictor**

$$\beta_{w(\infty)} \to \arg\min \|\beta\|_2 \;\; s.t. \forall_i y_i \langle \beta, x_i \rangle \geq 1$$

Linear Conv Nets



L-1 hidden layers, $h_l \in \mathbb{R}^n$, each with (one channel) full-width cyclic "convolution" $w_\ell \in \mathbb{R}^D$:

$$h_l[d] = \sum_{k=0}^{D-1} w_l[k] h_{l-1}[d + k \bmod D] \qquad h_{out} = \langle w_L, h_{L-1} \rangle$$

With single conv layer (L=2), training weights with SGD

→ $\arg\min \|DFT(\boldsymbol{\beta})\|_1 \;\; s.t.\, \forall_i y_i \langle \beta, x_i \rangle \geq 1$

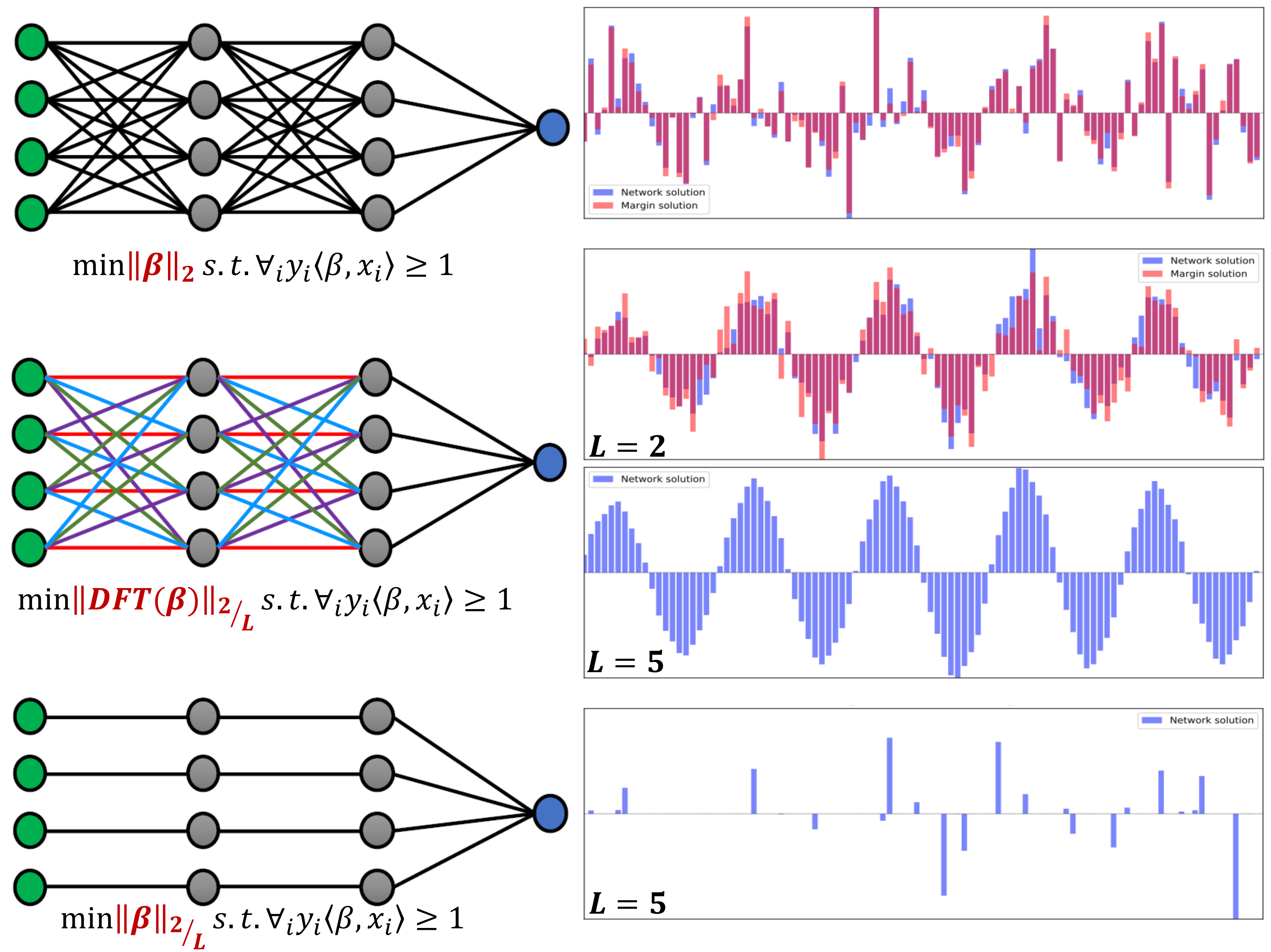
Discrete Fourier Transform

With multiple conv layers

→ critical point of $\min \|DFT(\boldsymbol{\beta})\|_{2/L} \;\; s.t.\, \forall_i y_i \langle \beta, x_i \rangle \geq 1$

for $\ell(z) = \exp(-z)$, almost all linearly separable data sets and initializations $w(0)$ and any bounded stepsizes s.t. $\mathcal{L} \to 0$, and $\Delta w(t)$ converge in direction

[Gunasekar Lee Soudry S 2018]

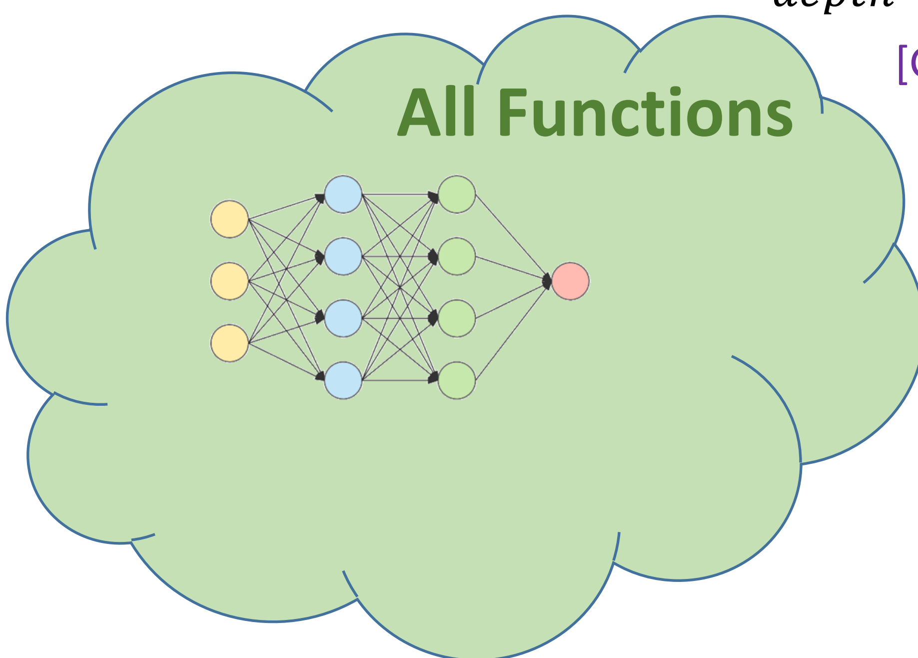


$\min \|\boldsymbol{\beta}\|_2 \; s.t. \forall_i y_i \langle \beta, x_i \rangle \geq 1$
Network solution
Margin solution
$\min \|\boldsymbol{DFT(\beta)}\|_{2/L} \; s.t. \forall_i y_i \langle \beta, x_i \rangle \geq 1$
Network solution
Margin solution
$L = 2$
Network solution
$L = 5$
$\min \|\boldsymbol{\beta}\|_{2/L} \; s.t. \forall_i y_i \langle \beta, x_i \rangle \geq 1$
Network solution
$L = 5$

- **Binary matrix completion** (also: reconstruction from linear measurements)
  - $\boldsymbol{X} = UV$ is over-parametrization of **all matrices** $\boldsymbol{X} \in \mathbb{R}^{n \times m}$
  - GD on U, V
    ➔ implicitly minimize $\|\boldsymbol{X}\|_*$ [Gunasekar Lee Soudry S 2018a]

- **Linear Convolutional Network**:
  - Complex over- β
  - GD on weights **(or explicitly minimize $\|\boldsymbol{weights}\|_2^2$)**
    ➔ implicitly min $\|\boldsymbol{DFT(\beta)}\|_{\boldsymbol{p}}$ for $p = \frac{2}{depth}$ (sparsity in freq domain)

[Gunasekar Lee Soudry S 2018b]



- **Binary matrix completion** (also: reconstruction from linear measurements)
  - $\boldsymbol{X} = UV$ is over-parametrization of **all matrices** $\boldsymbol{X} \in \mathbb{R}^{n \times m}$
  - GD on U, V

    ➔ implicitly minimize $\|\boldsymbol{X}\|_*$ [Gunasekar Lee Soudry S 2018a]

- **Linear Convolutional Network**:
  - Complex over-parametrization of **all linear predictors** β
  - GD on weights

    ➔ implicitly min $\|\boldsymbol{DFT(\beta)}\|_{\boldsymbol{p}}$ for $p = \frac{2}{depth}$ (sparsity in freq domain)

    [Gunasekar Lee Soudry S 2018b]

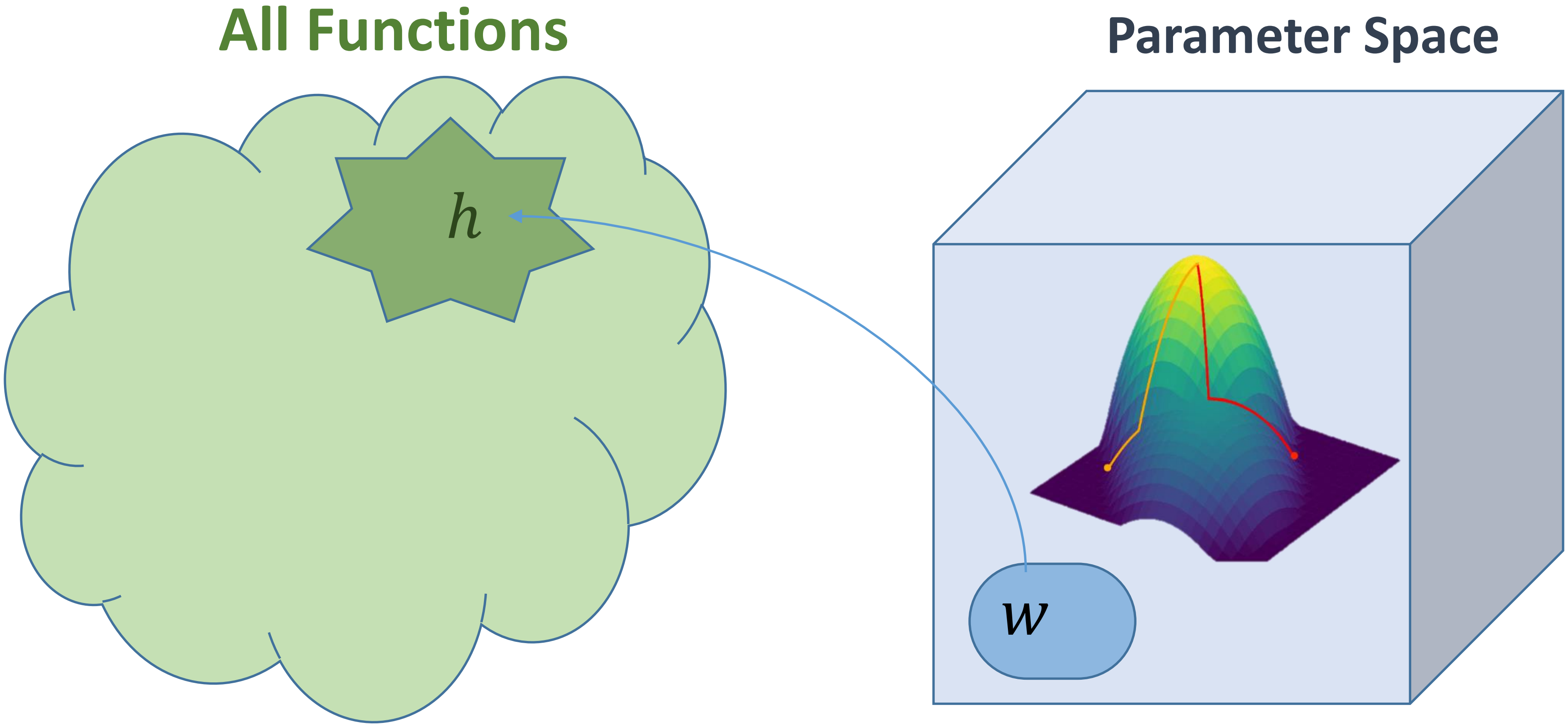
- **Infinite Width ReLU Net**:
  - Parametrization of essentially **all functions** $h: \mathbb{R}^d \to \mathbb{R}$
  - GD on weights

    ➔ implicitly minimize $\max\left(\int |\boldsymbol{h''}| \boldsymbol{dx} \;, |h'(-\infty) + h'(+\infty)| \right)$ (d=1)

    $\int \left| \boldsymbol{\partial_b^{d+1} Radon(h)} \right|$ (d>1)

    (need to define more carefully to handle non-smoothness; correction term for linear part)

    [Savarese Evron Soudry S 2019][Ongie Willett Soudry S 2020][Chizat Bach 2020]



Optimization Geometry and hence Inductive Bias effected by:

- Geometry of local search in parameter space
- Choice of parameterization

Artificial Neural Networks | Deep Learning | Computer Netwo

How is an Embedding layer useful to a learning task if it is just a dense layer with no activation function? If this 'linear hidden layer' is taken out, the network should still be able to learn the same function.

Answer · Interesting 16 · Request · More

- **Binary matrix completion** (also: reconstruction from linear measurements)
  - $\boldsymbol{X} = UV$ is over-parametrization of **all matrices** $\boldsymbol{X} \in \mathbb{R}^{n \times m}$
  - GD on U, V **(or explicitly minimize $\|\boldsymbol{U}\|_F^2 + \|\boldsymbol{V}\|_F^2$)**

    ➔ implicitly minimize $\|\boldsymbol{X}\|_*$ [Gunasekar Lee Soudry S 2018a]

- **Linear Convolutional Network**:
  - Complex over-parametrization of **all linear predictors** β
  - GD on weights **(or explicitly minimize $\|\boldsymbol{weights}\|_2^2$)**

    ➔ implicitly min $\|\boldsymbol{DFT(\beta)}\|_{\boldsymbol{p}}$ for $p = \frac{2}{depth}$ (sparsity in freq domain)

    [Gunasekar Lee Soudry S 2018b]

- **Infinite Width ReLU Net**:
  - Parametrization of essentially **all functions** $h: \mathbb{R}^d \to \mathbb{R}$
  - GD on weights **(or explicitly min $\|\boldsymbol{weights}\|_2^2$)**

    ➔ implicitly minimize $\max\left(\int |\boldsymbol{h''}| \boldsymbol{dx} \; , |h'(-\infty) + h'(+\infty)| \right)$ (d=1)

    $\int \left| \boldsymbol{\partial_b^{d+1} Radon(h)} \right|$ (d>1)

    (need to define more carefully to handle non-smoothness; correction term for linear part)

    [Savarese Evron Soudry S 2019][Ongie Willett Soudry S 2020][Chizat Bach 2020]

- Does Implicit Bias of Gradient Descent just boil down to regularizing $\|weights\|_2$?

- Answer: sort of, at least asymptotically with logistic/exp loss, for D-homogenous models (details soon)

…but we'll later see that not quite

Model: $F(w) = h_w$ Model Class: $\mathcal{H} = \mathrm{range}(F)$

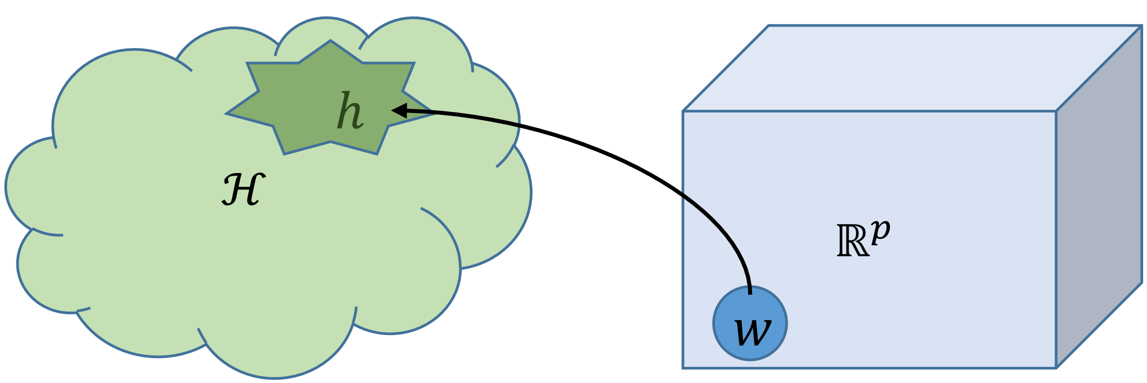
$f(w, x) = h_w(x) =$ prediction on x with params ("weights") w

Linear models: $f(w, x) = \langle \beta_w, x \rangle$ $F(w) = \beta_w$

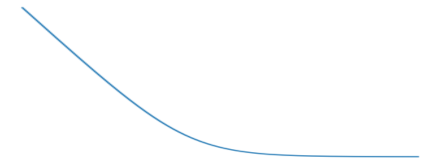
Loss: $L_S(w) = \frac{1}{m} \sum_i \ell(f(w, x_i), y_i)$

D-homogenous: $F(cw) = c^D F(w)$, i.e. $f(cw, x) = c^D f(w, x)$

- 1-homogenous: standard linear $F(w) = w$, $f(w, x) = \langle w, x \rangle$
- 2-homogenous:
  - Matrix factorization $F(U, V) = UV$
  - 2-Layer ReLU: $f(W, x) = \sum_j w_{2,j} \left[\langle w_{1,j}, x \rangle\right]_+$
- D-homogenous:
  - D layer linear network
  - D layer linear conv net
  - D layer ReLU net

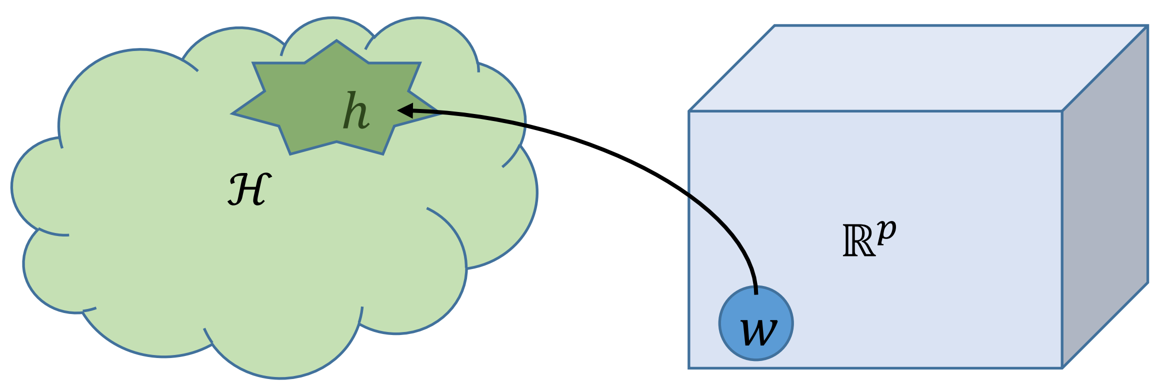


$$\ell_{\text{logistic}}(h(w), y) = \log\left(1 + e^{-yh(w)}\right) \approx e^{-yh(w)} = \ell_{\exp}(h(w), y)$$

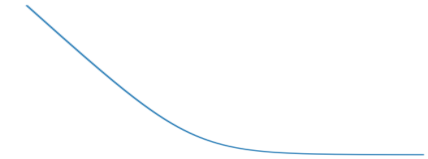


Consider gradient descent w.r.t. logistic loss $L_s(w) = \sum_i \ell(f(w, x_i); y_i)$
(or other exp-tail loss) on a D-homogenous model $f(w, x)$

- 1-homogenous: standard linear $F(w) = w$, $f(w, x) = \langle w, x \rangle$
- 2-homogenous:
    - Matrix factorization $F(U, V) = UV$
    - 2-Layer ReLU: $f(W, x) = \sum_j w_{2,j} \left[\langle w_{1,j}, x \rangle \right]_+$
- D-homogenous:
    - D layer linear network
    - D layer linear conv net
    - D layer ReLU net



$$\ell_{\text{logistic}}(h(w), y) = \log\left(1 + e^{-yh(w)}\right) \approx e^{-yh(w)} = \ell_{\exp}(h(w), y)$$



Consider gradient descent w.r.t. logistic loss $L_s(w) = \sum_i \ell(f(w, x_i); y_i)$

(or other exp-tail loss) on a D-homogenous model $f(w, x)$:

<u>**Theorem**</u> [Nacson Gunasekar Lee S Soudry 2019][Lyu Li 2019]:

If $L_s(w) \to 0$, and small enough stepsize (ensuring convergence in direction):

$\boldsymbol{w_\infty}$ **∝ first order stationary point of**

$$\boldsymbol{\arg\min \|w\|_2 \; s.t. \forall_i y_i f(w, x_i) \geq 1} \qquad \textbf{(*)}$$

Suggests implicit bias defined by $\boldsymbol{R_F(h) = \arg\min_{F(w)=h} \|w\|_2}$ and

$\boldsymbol{h_\infty = F(w_\infty)}$ **∝ first order stationary point of**

$$\boldsymbol{\arg\min R_F(h) \; s.t. \; y_i f(x_i) \geq 1} \qquad \textbf{(**)}$$

But need to be careful: f.o.s.p of **(*)** does ***not*** imply f.o.s.p of **(**)**

- But what about squared loss?

$$\ell(h(w); y) = (h(w) - y)^2$$

$$\text{GD on } L_s(w) = \sum_i \ell(f(w, x_i); y_i)$$

- What optimization choices and hyperparameters effect the implicit bias and how? E.g.
  - Stepsize
  - Initialization

- Initialize $w(0) = \alpha w_0$ (we will want to take $\alpha \to 0$)
- Stepsize$\to$ 0, so i.e. **gradient flow**:

$$\dot{w}_\alpha = -\nabla L_S(w) \quad \text{and} \quad w_\alpha(0) = \alpha w_0$$

We are interested in $w_\alpha(\infty) = \lim_{t \to \infty} w_\alpha(t)$

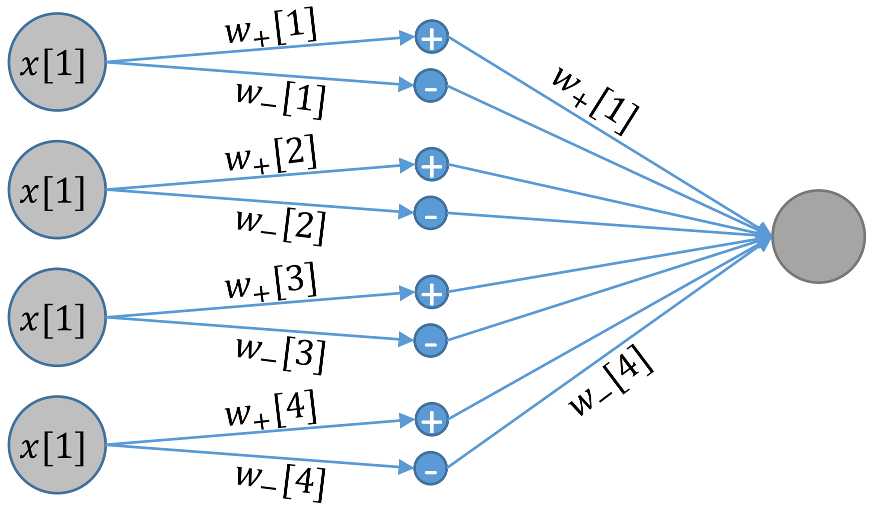
Consider a "linear diagonal net" (ie linear regression with squared parametrization):

$$f(w,x) = \sum_j (w_+[j]^2 - w_-[j]^2)x[j] = \langle \beta(w), x \rangle \qquad \text{with } \beta(w) = w_+^2 - w_-^2$$

And initialization $w_\alpha(0) = \alpha \mathbf{1}$ (so that $\beta\big(w_\alpha(0)\big) = 0$).

What's the implicit bias of grad flow w.r.t square loss $L_S(w) = \sum_i (f(w,x_i) - y_i)^2$?

$$\beta_\alpha(\infty) = \lim_{t\to\infty} \beta(w_\alpha(t))$$



$$f(w,x) = w^\top \, diag(w) \begin{bmatrix} +x \\ -x \end{bmatrix}$$

$$\beta(t) = w_+(t)^2 - w_-(t)^2$$

$$L = \|X\beta - y\|_2^2$$

$$\dot{w}_+(t) = -\nabla L(t) = -2X^\top r(t) \circ \frac{d\beta}{dw_+}$$

$$r(t) = X\beta(t) - y$$

$$\beta(t) = w_+(t)^2 - w_-(t)^2 \qquad\qquad L = \|X\beta - y\|_2^2$$

$$\dot{w}_+(t) = -\nabla L(t) = -2X^\top r(t) \circ 2w_+(t) \qquad w_+(t) = w_+(0) \circ \exp\left(-2X^\top \int_0^t r(\tau)\, d\tau\right)$$

$$\dot{w}_-(t) = -\nabla L(t) = +2X^\top r(t) \circ 2w_-(t) \qquad w_-(t) = w_-(0) \circ \exp\left(+2X^\top \int_0^t r(\tau)\, d\tau\right)$$

$$\boxed{\beta(t) = \alpha^2\left(e^{-4X^\top \int_0^t r(\tau)\, d\tau} - e^{4X^\top \int_0^t r(\tau)\, d\tau}\right) \qquad r(t) = X\beta(t) - y}$$

$$s = 4\int_0^\infty r(\tau) d\tau \in \mathbb{R}^m$$

$$\beta(\infty) = \alpha^2\left(e^{-X^T s} - e^{X^\top s}\right) = 2\alpha^2 \sinh X^\top s$$

$$X\beta(\infty) = y$$

$$\min Q(\beta) \quad s.t.\ X\beta = y$$

$$\nabla Q(\beta^*) = X^\top \nu$$

$$X\beta^* = y$$

$$\beta(\infty) = \alpha^2 \left(e^{-X^T s} - e^{X^\top s}\right) = 2\alpha^2 \sinh X^\top s$$

$$X\beta(\infty) = y$$

$$\nabla Q(\beta) = \sinh^{-1} \frac{\beta}{2\alpha^2}$$

$$Q(\beta) = \sum_i \int \sinh^{-1} \frac{\beta[i]}{2\alpha^2} = \alpha^2 \sum_i \left(\frac{\beta[i]}{\alpha^2} \sinh^{-1} \frac{\beta[i]}{2\alpha^2} - \sqrt{4 + \left(\frac{\beta[i]}{\alpha^2}\right)^2} \right)$$

$$\min Q(\beta) \quad s.t. \; X\beta = y$$

$$\nabla Q(\beta^*) = X^\top \nu$$

$$X\beta^* = y$$

$$\sinh^{-1} \frac{\beta(\infty)}{2\alpha^2} = X^\top s$$

$$X\beta(\infty) = y$$

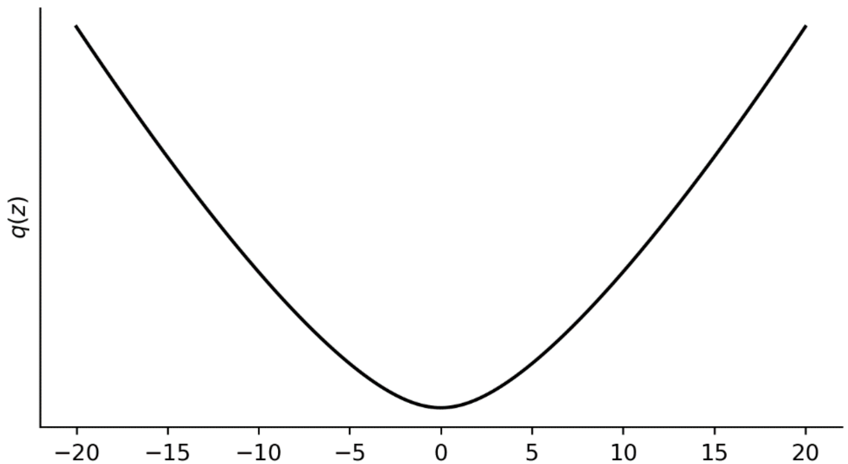
Linear Diagonal Nets

$$f(w,x) = \sum_j (w_+[j]^2 - w_-[j]^2)x[j] = \langle \beta(w), x\rangle \qquad \text{with } \beta(w) = w_+^2 - w_-^2$$

With initialization $w_\alpha(0) = \alpha\mathbf{1}$ (so that $\beta\big(w_\alpha(0)\big) = 0$).

Implicit bias of grad flow w.r.t square loss: $\beta_\alpha(\infty) = \boldsymbol{\arg\min_{X\beta=y} Q_\alpha(\beta)}$

where $\boldsymbol{Q_\alpha(\beta) = \sum_j q\left(\frac{\beta[j]}{\alpha^2}\right)}$ **and** $\boldsymbol{q(b) = 2 - \sqrt{4+b^2} + b\sinh^{-1}\left(\frac{b}{2}\right)}$



Induced dynamics:

$$\dot{\beta}_\alpha = -\sqrt{\beta_\alpha^2 + 4\alpha^4} \odot \nabla L_s(\beta_\alpha)$$

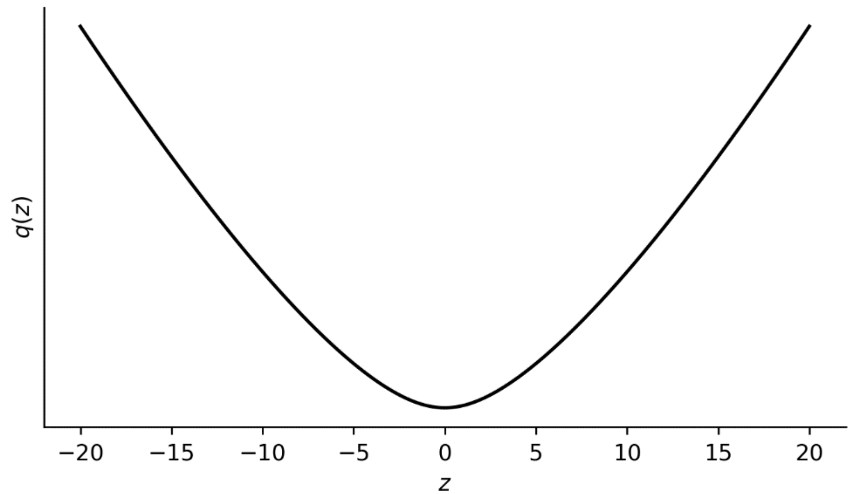
If $\alpha \to \infty$ (Kernel Regime): $\beta_\alpha(\infty) \xrightarrow{\alpha\to\infty} \hat{\beta}_{L2} = \arg\min_{X\beta=y} \|\beta\|_2$

If $\alpha \to 0$ (“Rich” Regime): $\beta_\alpha(\infty) \xrightarrow{\alpha\to 0} \hat{\beta}_{L1} = \arg\min_{X\beta=y} \|\beta\|_1$

(special case of matrix factorization with commutative measurements)

$$\beta_{\alpha}(\infty) = \arg\min_{X\beta=y} Q_{\alpha}(\beta)$$

where $Q_{\alpha}(\beta) = \sum_j q\left(\frac{\beta[j]}{\alpha^2}\right)$ and $q(b) = 2 - \sqrt{4+b^2} + b\sinh^{-1}\left(\frac{b}{2}\right)$



Theorem 2. *For any* $0 < \epsilon < d$,

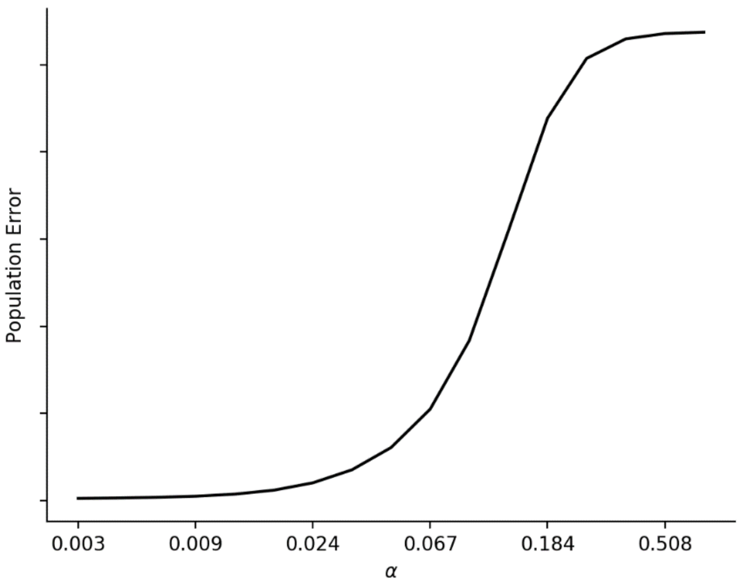
$$\alpha \leq \min\left\{\left(2(1+\epsilon)\left\|\boldsymbol{\beta}^*_{L1}\right\|_1\right)^{-\frac{2+\epsilon}{2\epsilon}}, \exp\left(-\frac{d}{\epsilon\left\|\boldsymbol{\beta}^*_{L1}\right\|_1}\right)\right\} \implies \left\|\hat{\boldsymbol{\beta}}_{\alpha}\right\|_1 \leq (1+\epsilon)\left\|\boldsymbol{\beta}^*_{L1}\right\|_1$$

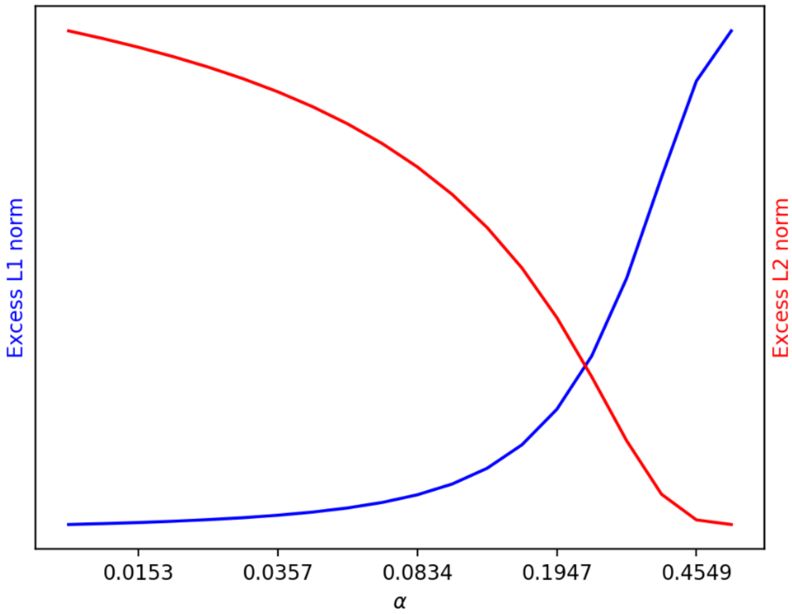
Theorem 3. *For any* $\epsilon > 0$

$$\alpha \geq \sqrt{2(1+\epsilon)\left(1+\frac{2}{\epsilon}\right)\left\|\boldsymbol{\beta}^*_{L2}\right\|_2} \implies \left\|\hat{\boldsymbol{\beta}}_{\alpha}\right\|_2^2 \leq (1+\epsilon)\left\|\boldsymbol{\beta}^*_{L2}\right\|_2^2$$

Sparse Learning

$$y_i = \langle \beta^*, x_i \rangle + N(0,0.01)$$
$$d = 1000, \qquad \|\beta^*\|_0 = 5, \qquad m = 100$$

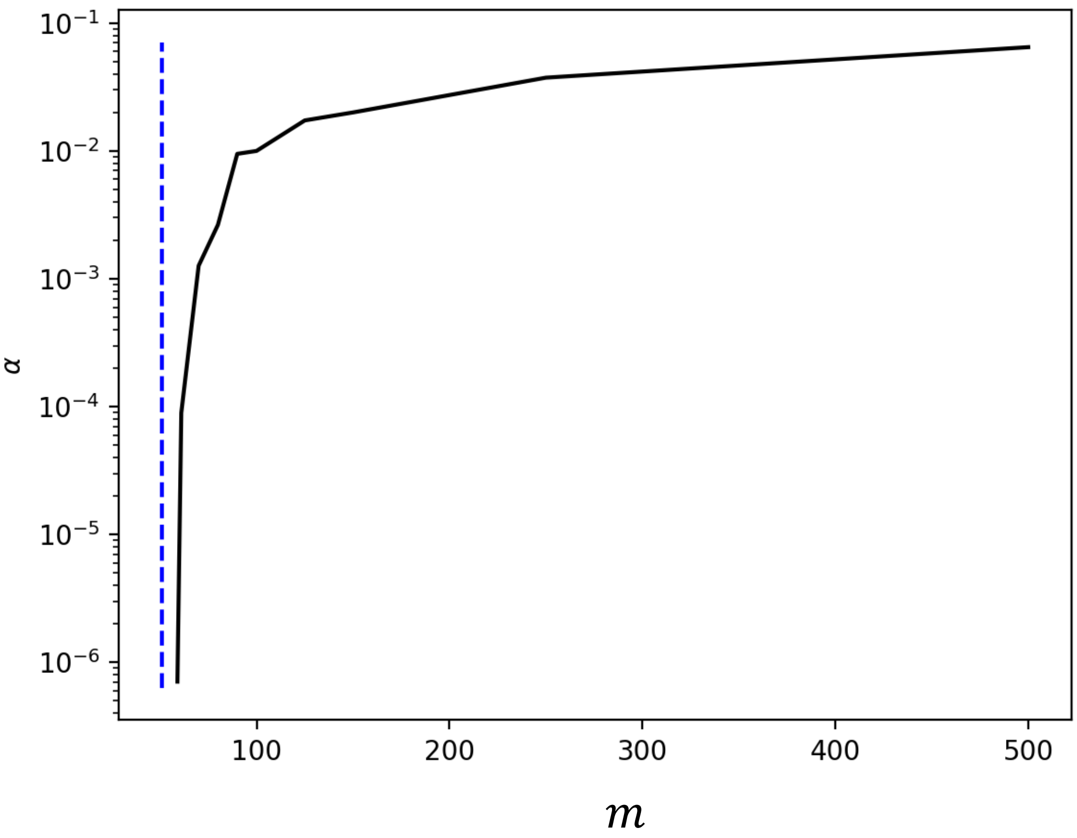




Sparse Learning

$$y_i = \langle \beta^*, x_i \rangle + N(0{,}0.01)$$
$$d = 1000, \qquad \|\beta^*\|_0 = k$$

How small does α need to be to get $L\big(\beta_\alpha(\infty)\big) < 0.025$

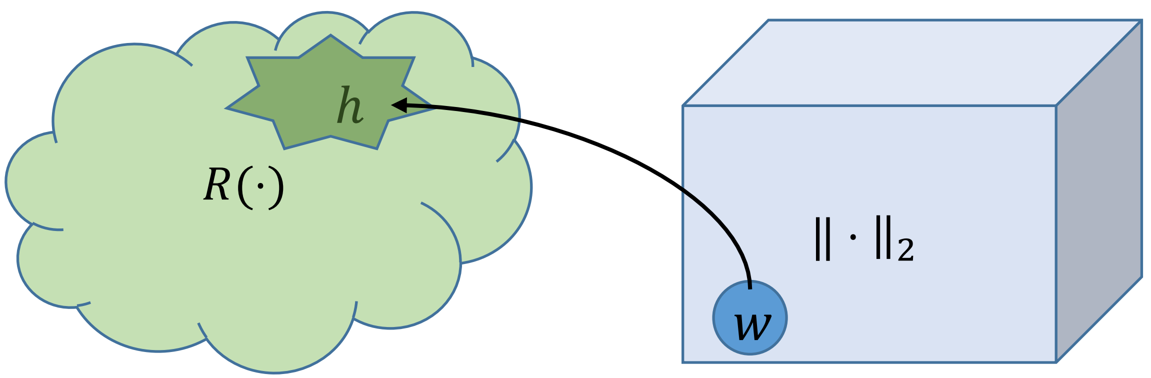


Is implicit bias of GD just ℓ_2 in param space + mapping to func space?

Is initializing to $w(0) = \alpha\mathbf{1}$ the same as regularizing distance to $\alpha\mathbf{1}$?

$$\beta_\alpha^R = F\left(\arg\min_{L_S(w)=0} \|w - \alpha\mathbf{1}\|_2^2\right) = \arg\min_{X\beta=y} R_\alpha(\beta)$$

Where $R_\alpha(\beta) = \min_{F(w)=\beta} \|w - \alpha\mathbf{1}\|_2^2$



Is implicit bias of GD just ℓ_2 in param space + mapping to func space?

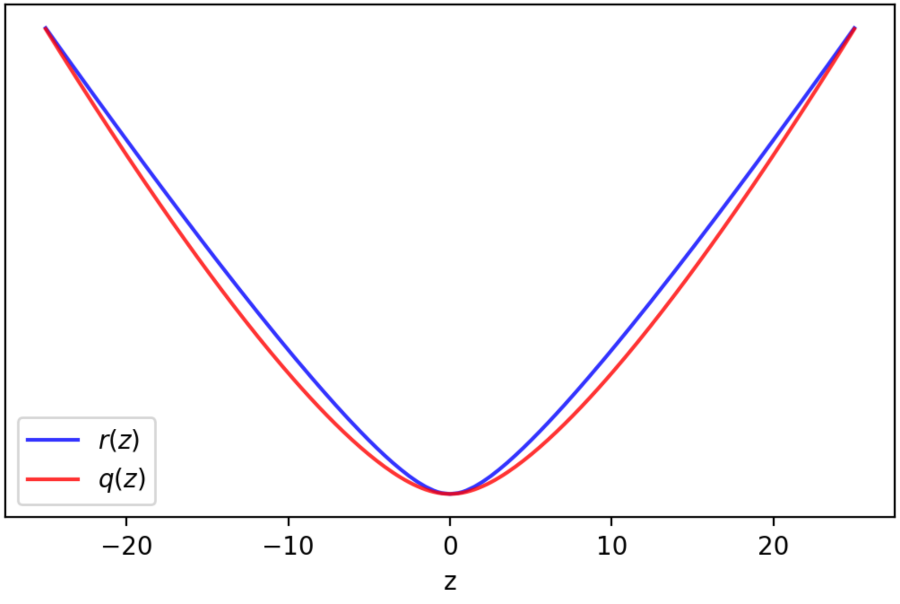
Is initializing to $w(0) = \alpha\mathbf{1}$ the same as regularizing distance to $\alpha\mathbf{1}$?

$$\beta_\alpha^R = F\left(\arg\min_{L_S(w)=0} \|w - \alpha\mathbf{1}\|_2^2\right) = \arg\min_{X\beta=y} R_\alpha(\beta)$$

Where $R_\alpha(\beta) = \min_{F(w)=\beta} \|w - \alpha\mathbf{1}\|_2^2$

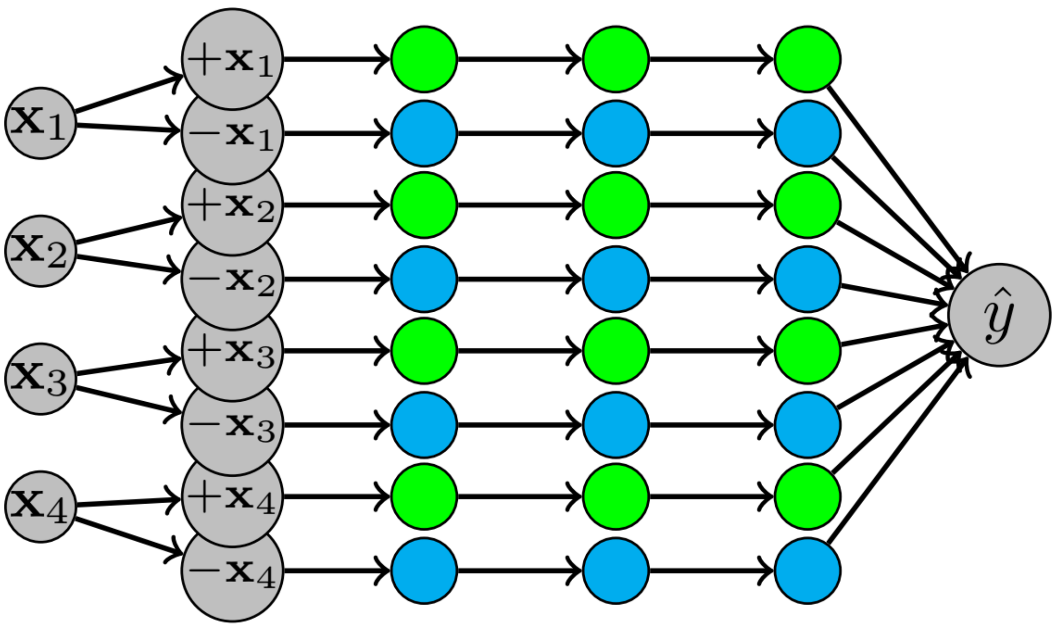
$R_\alpha(\beta) = \sum_j r\left(\frac{\beta[j]}{\alpha^2}\right)$ where $r(b)$ is solution of quartic equation:

$$r^4 - 6r^3 + (12 - 2b^2)r^2 - (8 + 10b^2)r + b^2 + b^4 = 0$$



Deep Diagonal Linear Net

$\beta(t) = w_+(t)^D - w_-(t)^D$



Deep Diagonal Linear Net

$$\beta(t) = w_+(t)^D - w_-(t)^D$$

$$r(t) = X\beta(t) - y$$

$$\beta(t) = \alpha^D \left(\left(1 + \alpha^{D-2} D(D-2) X^\top \int_0^t r(\tau)\, d\tau \right)^{\frac{-1}{D-2}} - \left(1 - \alpha^{D-2} D(D-2) X^\top \int_0^t r(\tau)\, d\tau \right)^{\frac{-1}{D-2}} \right)$$

KKT for $\min Q(\beta) \;\; s.t.\; X\beta = y$:

$$s = \alpha^{D-2} D(D-2) \int_0^\infty r(\tau) d\tau \in \mathbb{R}^m$$

$$\nabla Q(\beta^*) = X^\top \nu$$

$$X\beta^* = y$$

$$\beta(\infty) = \alpha^D h_D(X^\top s)$$

$$X\beta(\infty) = y$$

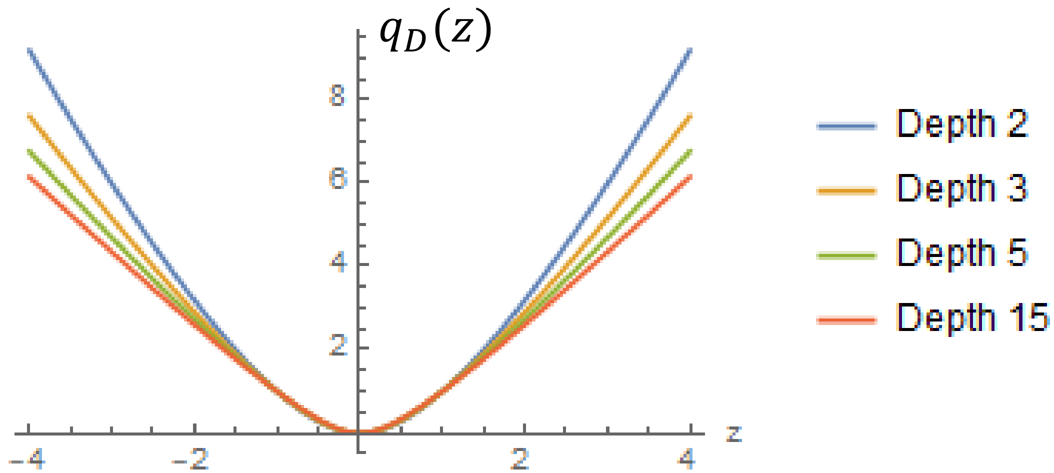
$$h_D(z) = \alpha^D \left((1 + \alpha^{D-2} D(D-2) z)^{\frac{-1}{D-2}} - (1 - \alpha^{D-2} D(D-2) z)^{\frac{-1}{D-2}} \right)$$

$$q_D = \int h_D^{-1}$$

$$Q_D(\beta) = \sum_i q_D\left(\frac{\beta[i]}{\alpha^D}\right)$$

Deep Diagonal Linear Net

$$\beta(t) = w_+(t)^D - w_-(t)^D \qquad \beta(\infty) = \arg\min Q_D\left({}^{\beta}/_{\alpha^D}\right) \; s.t. \; X\beta = y$$



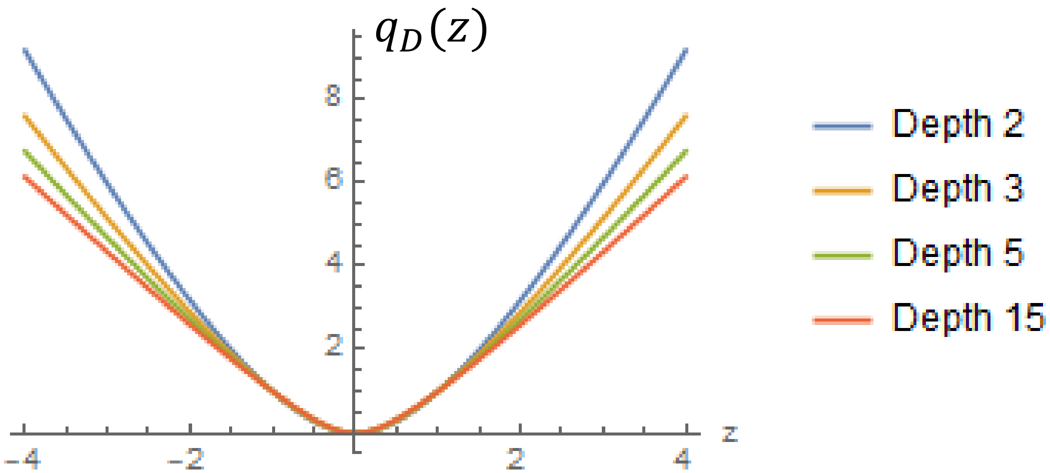
$$h_D(z) = \alpha^D\left((1+\alpha^{D-2}D(D-2)z)^{\frac{-1}{D-2}} - (1-\alpha^{D-2}D(D-2)z)^{\frac{-1}{D-2}}\right)$$

$$q_D = \int h_D^{-1}$$

$$Q_{D,\alpha}(\beta) = \sum_i q_D\left(\frac{\beta[i]}{\alpha^D}\right)$$

Deep Diagonal Linear Net

$$\beta(t) = w_+(t)^D - w_-(t)^D \qquad \beta(\infty) = \arg\min Q_D\left({}^{\beta}/_{\alpha^D}\right) \; s.t. \; X\beta = y$$



For all depth $D \geq 2$, $\beta(\infty) \xrightarrow{\alpha \to 0} \arg\min_{X\beta = y} \|\beta\|_1$

- Contrast with explicit reg: For $R_\alpha(\beta) = \min_{\beta = w_+^D - w_-^D} \|w - \alpha \mathbf{1}\|_2^2$, $\quad R_\alpha(\beta) \xrightarrow{\alpha \to 0} \|\beta\|_{2/D}$

  also observed by [Arora Cohen Hu Luo 2019]

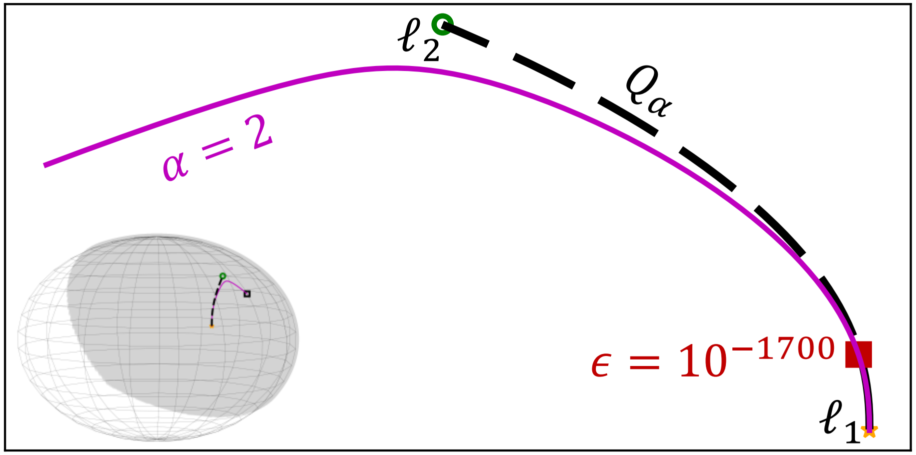
- Also with logistic loss, $\beta(\infty) \to \propto SOSP \; of \; \|\beta\|_{2/D}$ [Gunasekar Lee Soudry Srebro 2018] [Lyu Li 2019]

- With sq loss, always $\|\cdot\|_1$, but we get there if quicker depth is higher

Logistic Loss vs Squared Loss

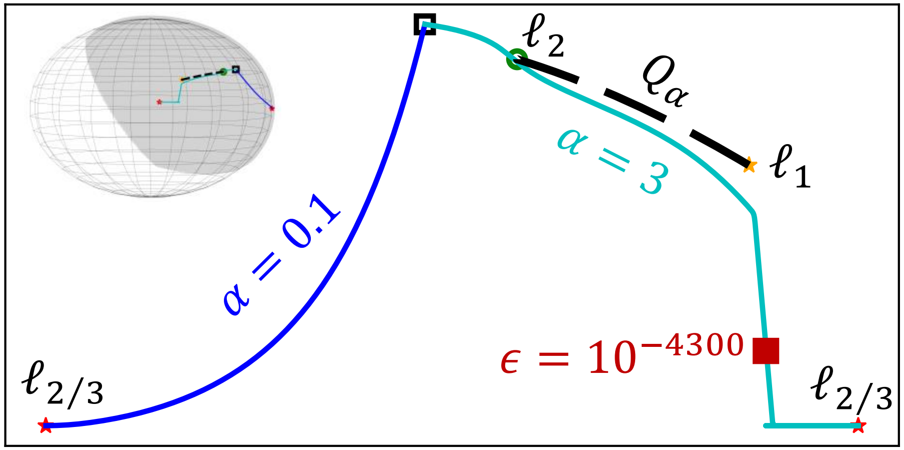
Depth two:

- Square loss: $\beta(\infty) \propto \arg\min_{X\beta=y} Q_\alpha(\beta)$
- Logistic loss: $\forall_\alpha \beta(\infty) \propto \arg\min_{X\beta=y} \|\beta\|_1$



Deeper Diagonal Nets:

- Squared loss, $\beta(\infty) \xrightarrow{\alpha\to 0} \propto \arg\min_{X\beta=y} \|\beta\|_1$
- Logistic loss, $\beta(\infty) \propto SOSP\ of\ \|\beta\|_{2/D}$



Depth=3

[Moroshko Gunasekar Woodworth Lee S Soudry 2020 "Implicit Bias in Deep Linear Classification: Initialization Scale vs Training Accuracy"]

Implicit bias of optimization (and hence inductive bias) effected by:

- Parametrization (architecture)
- Optimization "geometry" (GD vs AdaGrad vs coordinate methods)
- Type (asymptotics) of loss function
- Initialization
- Optimization accuracy
  - Early stopping
  - Not so early stopping
- Stepsize, momentum, other opt. parameters
- Stochasticity (SGD vs GD, mini-batch size, label noise)

  [Cheng Chatterji Bartlett Jordan 2018][HaoChen Wei Lee Ma 2020]
- ???

The "complexity measure" approach

Identify $c(h)$ s.t.

- Optimization algorithm biases towards low $c(h)$
- $\mathcal{H}_{c(reality)} = \{h | c(h) \leq c(reality)\}$ has low capacity
- Reality is well explained by low $c(h)$

Can optimization bias can be described as $\mathbf{arg\,min}\, \boldsymbol{c(h)}\ \boldsymbol{s.t.} \boldsymbol{L_S(h) = 0}$??

- Not always [Dauber Feder Koren Livni 2020]
- Approximately? Enough to explain generalization??

Ultimate Question: What is the true Inductive Bias? What makes reality *efficiently* learnable by fitting a (huge) neural net with a specific algorithm?

Deep Learning

- Expressive Power
  - We are searching over the space of all functions…
    … but with what inductive bias?
  - How does this bias look in function space?
  - Is it reasonable/sensible?
- Capacity / Generalization ability / Sample Complexity
  - What's the true complexity measure (inductive bias)?
  - How does it control generalization?
- Computation / Optimization
  - How and where does optimization bias us? Under what conditions?
  - Magic property of reality under which deep learning "works"