Inductive Bias, Generalization and the role of Optimization in Deep Learning

Nati Srebro (TTIC)

Based on work with **Behnam Neyshabur (TTIC→IAS/NYU→??)**, **Suriya Gunasekar (TTIC→??)**,
Ryota Tomioka (TTIC→MSR), Srinadh Bhojanapalli (TTIC→Google),
Blake Woodworth, Pedro Savarese, Arturs Backurs, David McAllester (TTIC),
Daniel Soudry, Elad Hoffer, Mor Shpigel, Itay Sofer (Technion), Jason Lee (USC)
Russ Salakhutdinov (CMU), Ashia Wilson, Becca Roelofs, Mitchel Stern, Ben Recht (Berkeley),
Zhiyuan Li (Princeton), Yann LaCun (NYU/Facebook), Charlie Smart (UChicago).

- **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
$$\mathbb{E}_{S\sim\mathcal{D}^m}\left[L\left(A(S)\right)\right] \geq \frac{1}{2} - \frac{m}{2|\mathcal{X}|}$$
- **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 $S \sim \mathcal{D}^m$):
$$L\big(ERM_{\mathcal{H}}(S)\big) \leq L(h^*) + \mathcal{R}_m(\mathcal{H}) \approx L(h^*) + \sqrt{\frac{O\big(capacity(\mathcal{H})\big)}{m}}$$

  ➔ can learn with $O(capacity(\mathcal{H}))$
- E.g.
  - For binary classification, $capacity(\mathcal{H}) = VCdim(\mathcal{H})$

    Vapnik-Chrvonenkis (VC) dimension: largest number of points D that can be labeled (by some $h \in \mathcal{H}$) in every possible way (i.e. for which the inductive bias is uninformative)
  - For linear classifiers over d features, $VCdim(\mathcal{H}) = d$
  - Usually with d parameters, $VCdim(\mathcal{H}) \approx \tilde{O}(\#params)$
  - Always: $VCdim(\mathcal{H}) \leq \log|\mathcal{H}| = \#bits$

Flat Inductive Bias

- **"Flat" inductive bias**: $\exists h^* \in \mathcal{H}$ with low $L(h^*)$
- (Almost) optimal learning rule:

$$ERM_{\mathcal{H}}(S) = \hat{h} = \arg\min_{h \in \mathcal{H}} L_S(h)$$

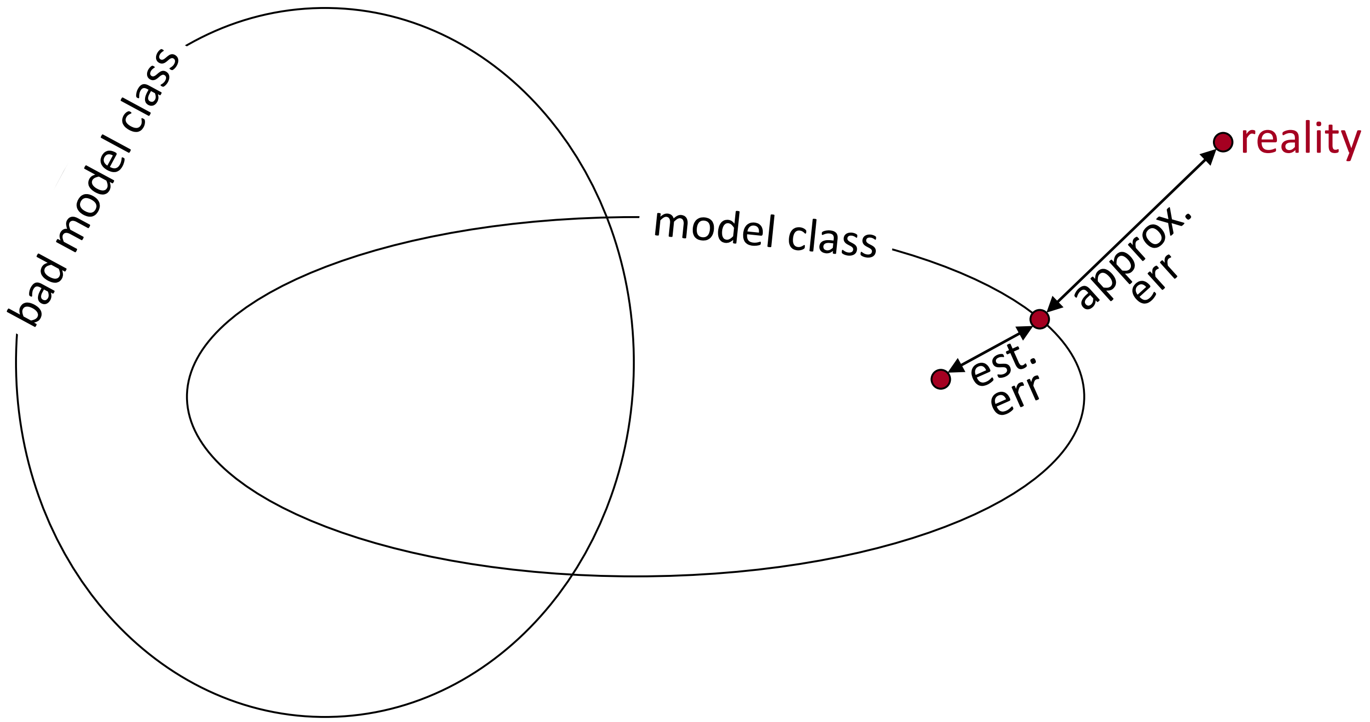
- Guarantee (in expectation over $S \sim \mathcal{D}^m$):

$$L\big(ERM_{\mathcal{H}}(S)\big) \leq L(h^*) + \mathcal{R}_m(\mathcal{H}) \approx L(h^*) + \sqrt{\frac{O\big(capacity(\mathcal{H})\big)}{m}}$$

➔ can learn with $O(capacity(\mathcal{H}))$

- E.g.
  - For binary prediction, $capacity(\mathcal{H}) = VCdim(\mathcal{H})$
  - For linear predictors over d features, $capacity(\mathcal{H}) = d$
  - Usually with d parameters, $capacity(\mathcal{H}) \approx \tilde{O}(\#params)$
  - Always: $capacity(\mathcal{H}) \leq \#bits$
  - 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 $c: \mathcal{Y}^{\mathcal{X}} \rightarrow [0, \infty]$
- Associated inductive bias: $\exists h$ with small $c(h)$ and small $L(h)$
- Learning rule: $SRM_{\mathcal{H}}(S) = \arg\min \; L(h) \; , \; c(h)$

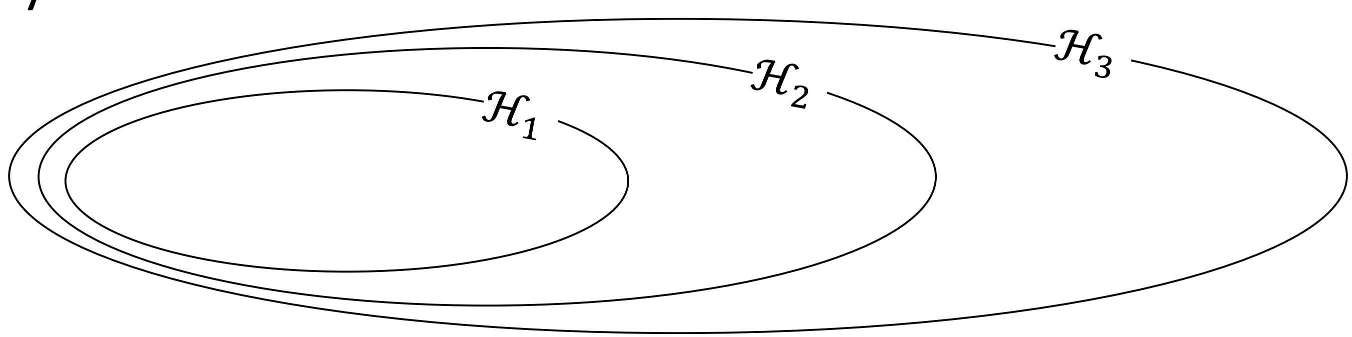
  e.g. $\arg\min L(h) + \lambda\, c(h)$ or $\arg\min L(h)$ s.t. $c(h) \leq B$

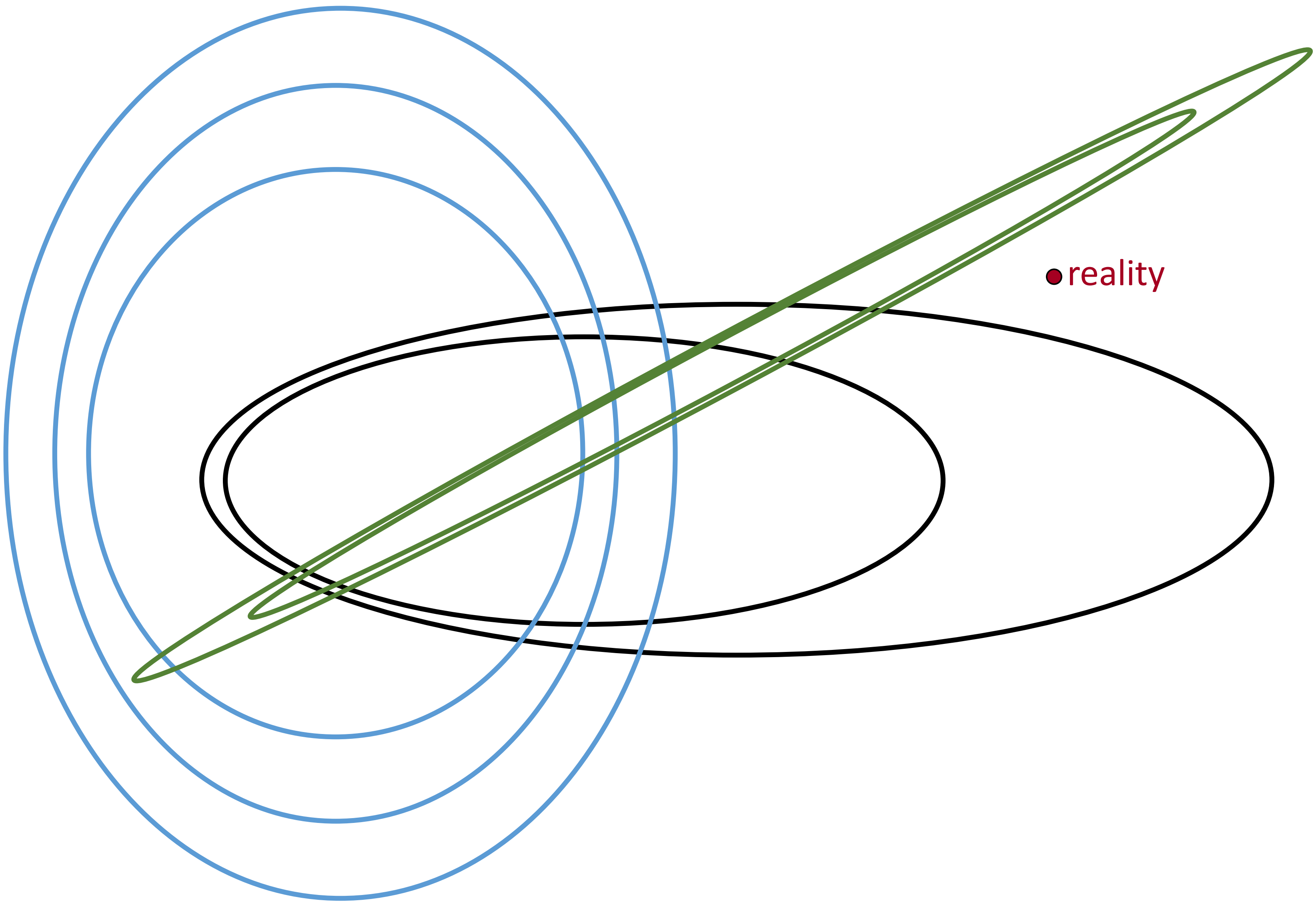
  and choose λ or B using cross-validation
- Guarantee:

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

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

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




reality

Beyond ERM: Implicit Inductive Bias

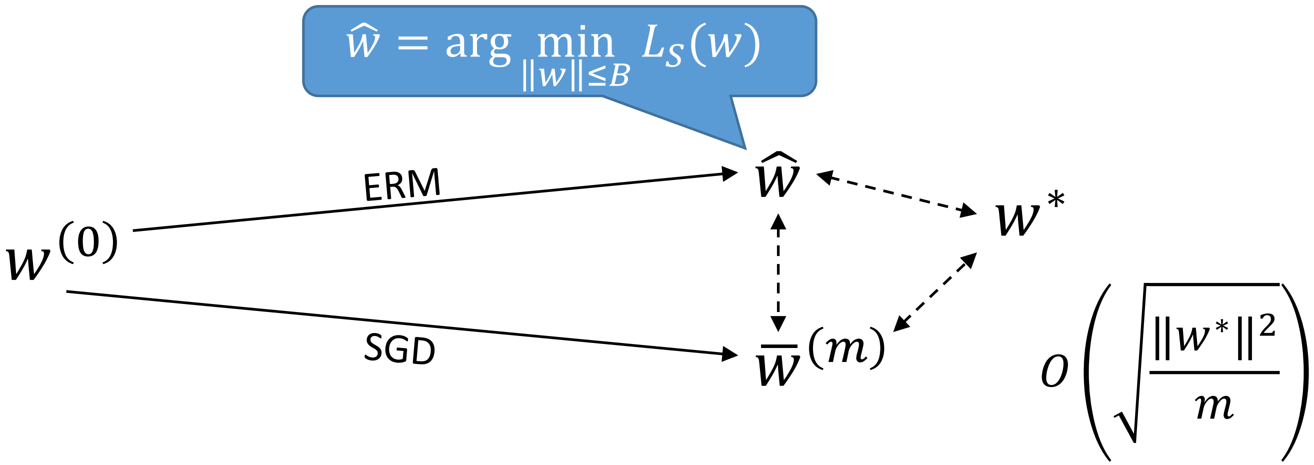
- The one-pass-SGD learning rule for linear predictors $h_w(x) = \langle w, x \rangle$

$$SGD(S) = \frac{1}{m}\sum_{i=1}^{m} w_i \quad \text{where} \quad w_{i+1} = w_i - \eta \nabla loss(\langle w, x_i \rangle; y_i)$$

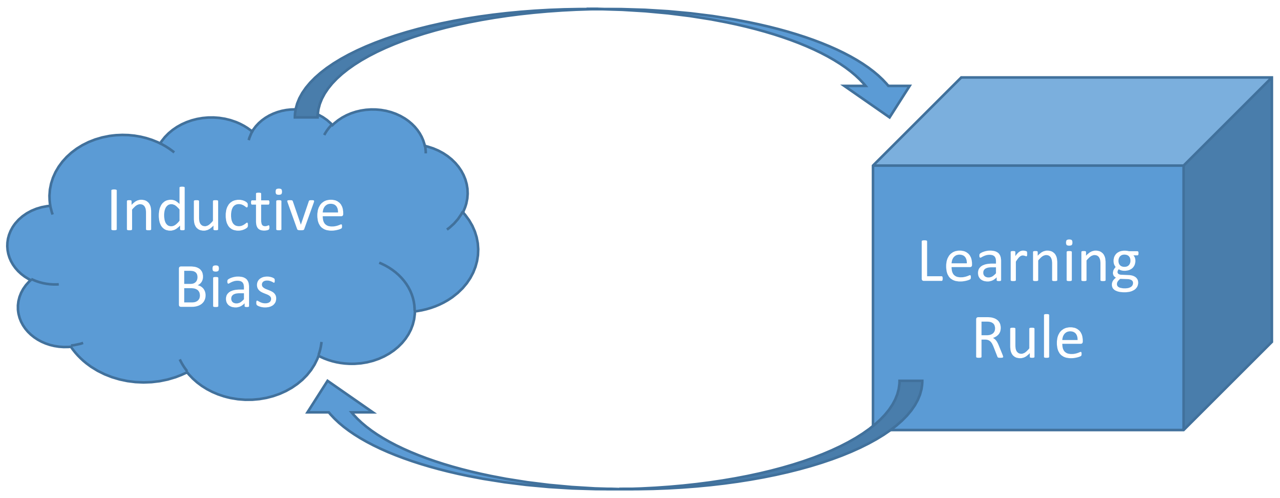
- **Theorem:** If $|loss'| \leq 1$ and $\|x\| \leq 1$, then with $w_0 = 0$ and appropriate η

$$L\big(SGD(S)\big) \leq L(w^*) + \sqrt{\frac{\|w\|_2}{m}}$$

- Inductive bias: $c(h_w) = \|w\|_2$

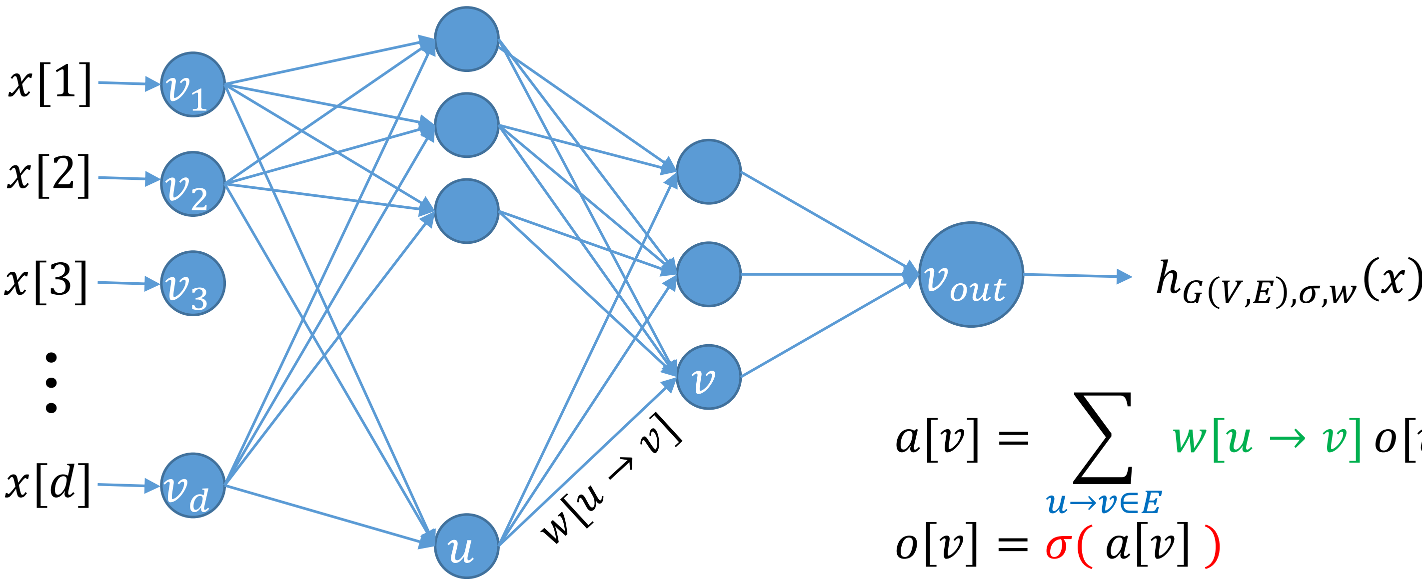


Explicit and Implicit Inductive Bias



$\mathcal{H}$ or $c(h)$	ERM or SRM
$\|w\|_2$	SGD
$P(y\|x)$ smooth w.r.t $d(x, x')$	Nearest Neighbor
sparsity or $\|w\|_1$	Exp GD (Mult Weights)
$c(h)$	Mirror Descent with potential $\approx c(h)$

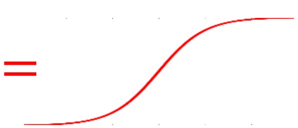
Feed-Forward Neural Networks (The Multilayer Perceptron)



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

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

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. $\sigma(z) = sign(z)$ or $\sigma(z) =$ 

Parameters:

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

Feed-Forward Neural Networks as a Hypothesis Class

$$\mathcal{H}_{G(V,E),\sigma} = \{\, h_{G(V,E),\sigma,w} \mid w: E \to \mathbb{R} \,\}$$

or $\mathcal{H}^{sign}_{G(V,E),\sigma} = \{\, sign(h_{G(V,E),\sigma,w}) \mid w: E \to \mathbb{R} \,\}$

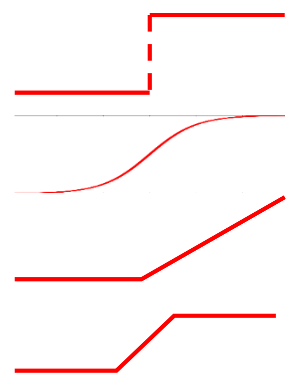
- Hypothesis class specified by: (ie we decide on this in advance)
  - Graph G(V,E)
    - V includes input, output and “hidden” nodes
  - Activation function σ

    e.g. $\mathrm{sign}(z)$,

    $\tanh(z)$, $\mathrm{sigmoid}(z) = \frac{1}{1+e^{-z}}$,

    $\mathrm{relu}(z) = \max(0, z)$,

    $\mathrm{ramp}(z) = clip_{[-1,1]}(z)$



- Hypothesis specified by: (ie we need to learn)
  - Weights w, with 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

- Expressive Power / Approximation

Capacity (Sample Complexity) of NN

- #params = $|E|$ (number of weights we need to learn)
- More formally: $VCdim\left(\mathcal{H}_{G(V,E),sign}\right) = O(|E|\log|E|)$
- Other activation functions?
  - $VCdim\left(\mathcal{H}_{G(V,E),\sin}\right) = \infty$ even with single unit and single real-valued input
  - For $\sigma(z) = sigmoid(z) = \frac{1}{1+e^{-z}}$:

$$\Omega(|E|^2) \leq VCdim(\mathcal{H}_{G(V,E),\mathrm{sigmoid}}) \leq O(|E|^4)$$

  - For piecewise linear, e.g. $ramp(z) = clip_{[-1,1]}(z)$ or $ReLU(z) = \max(0, z)$:

$$\Omega\left(|E|L\log {|E|}/_{L}\right) \leq VCdim(\mathcal{H}_{G,\sigma}) \leq O(|E|L\log|E|)$$

L=depth

- With integer weights $\in [-B, .., B]$:

$$VCdim\left(\mathcal{H}_{G(V,E),\sigma}\right) \leq \log\left|\mathcal{H}_{G(V,E),\sigma}\right| \leq 2|E|\log B$$

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

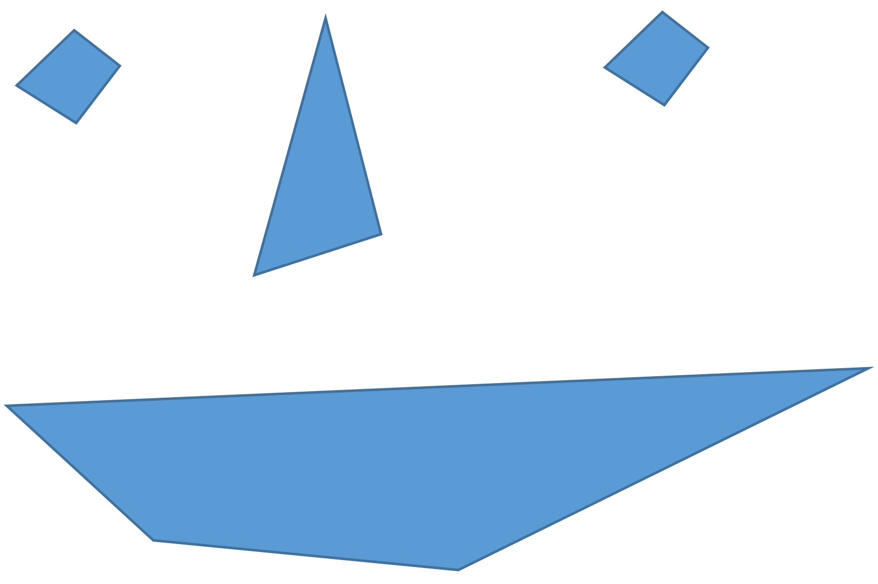
What can Feed-Forward Networks Represent?

- Any function over $\mathcal{X} = \{\pm 1\}^n$
  - With a single hidden layer, using DNF (hidden layer does AND, output does OR)
  - $|V| = 2^n$, $|E| = n2^n$
  - Like representing the truth table directly…

- Universal Representation Theorem: Any continuous functions $f: [0,1]^n \to \mathbb{R}$ can be approximated to within any ϵ by a feed-forward network with sigmoidal (or almost any other) activation and a single hidden layer.
  - Size of layer exponential in *n*

What can SMALL Networks Represent?

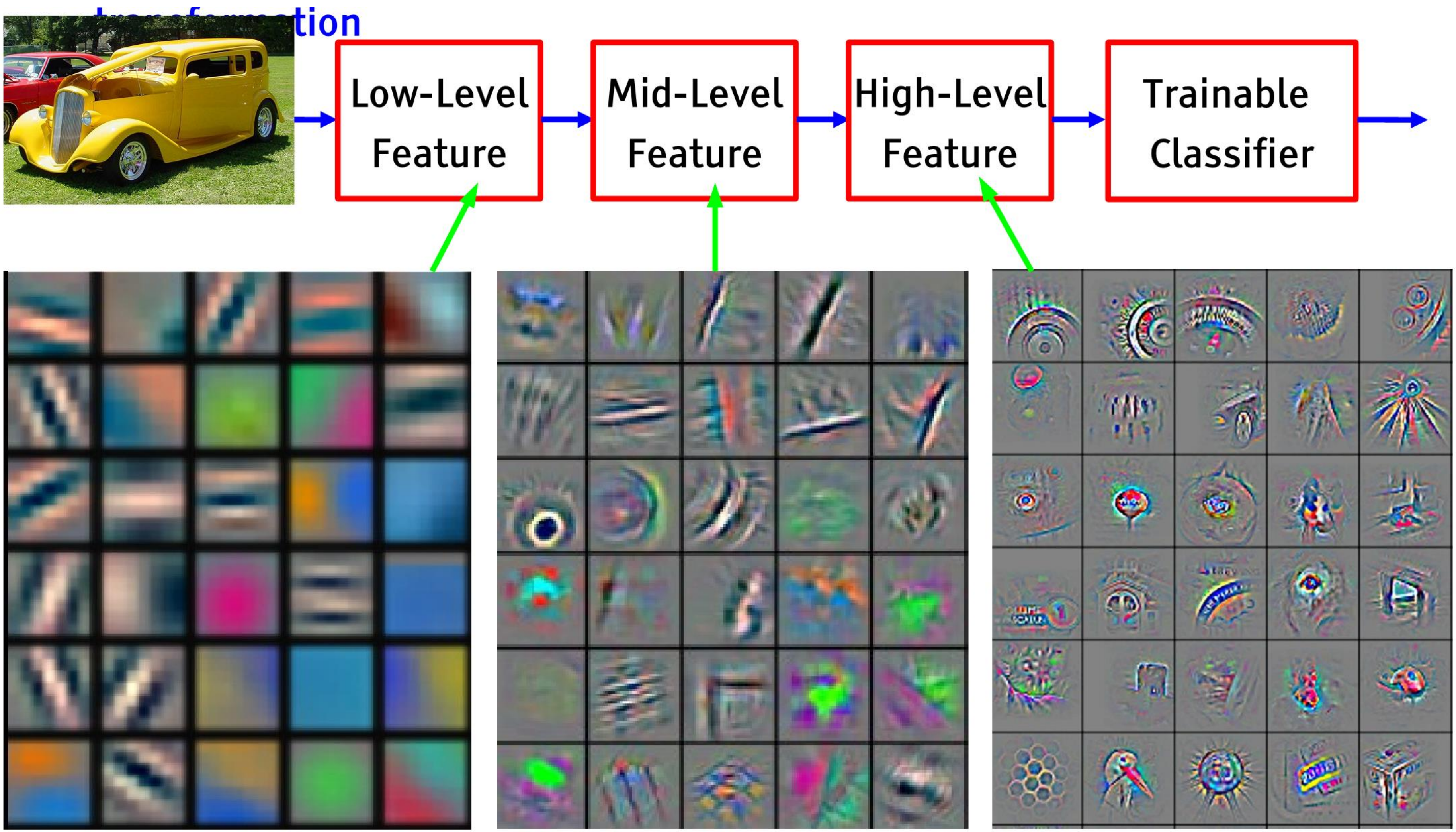
- Intersection of halfspaces
  - Using single hidden layer
- Union of intersection of halfspaces (and also sorting, more fun stuff, …)
  - Using two hidden layers



What can SMALL Networks Represent?

- Intersection of halfspaces
  - Using single hidden layer
- Union of intersection of halfspaces (and also sorting, more fun stuff, …)
  - Using two hidden layers
- Functions representable by a small logical circuit
  - Implement AND using single unit, negation by reversing weight
- Functions that depend on lower level features

Multi-Layer Feature Learning



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 $\{h | h \text{ is an efficient system}\}$
- <u>**Free Lunch**</u>: $\boldsymbol{TIME_T} = \{h | h \text{ comp. in time } T\}$ 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 $O(T)$
  - $|PROG_T| = 128^T$ ➔ capacity $\leq \log|PROG_T| = O(T)$
- 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 $\mathbb{E}\big[L\big(A(S)\big)\big] \leq 0.499$)

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^*)).
- **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

- 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 $\tilde{O}(T)$**
- Computation / Optimization
  - Non-convex
  - No known algorithm guaranteed to work
  - NP-hard to find weights even with 2 hidden units
  - Even if function exactly representable with single hidden layer with $\Theta(\log d)$ units, even with no noise, and even if we train a much larger network or use any other method when learning: no poly-time algorithm can ensure better-than-chance prediction
    [Kearns Valiant 94; Klivans Sherstov 06; Daniely Linial Shalev-Shwartz '14]

Choose your universal learner:

Short Programs

- Universal
- Captures anything we want with reasonable sample complexity
- Provably (worst case) hard to optimize
- Hard to optimize in practice

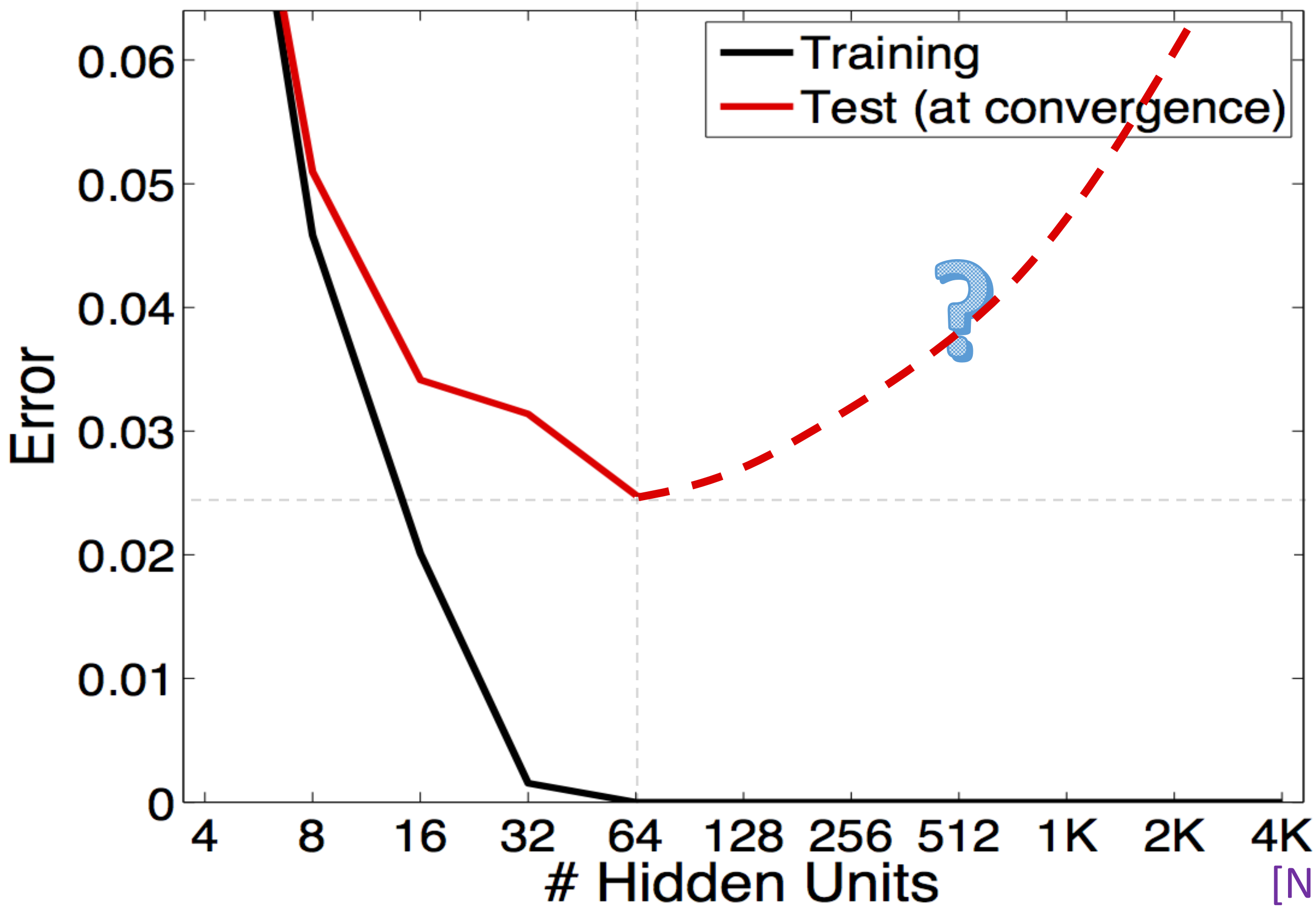
Deep Networks

- Universal
- Captures anything we want with reasonable sample complexity
- Provably (worst case) hard to optimize
- Often easy to optimize
  - Continuous
  - Amenable to local search, stochastic local search
  - **Lots of empirical success**

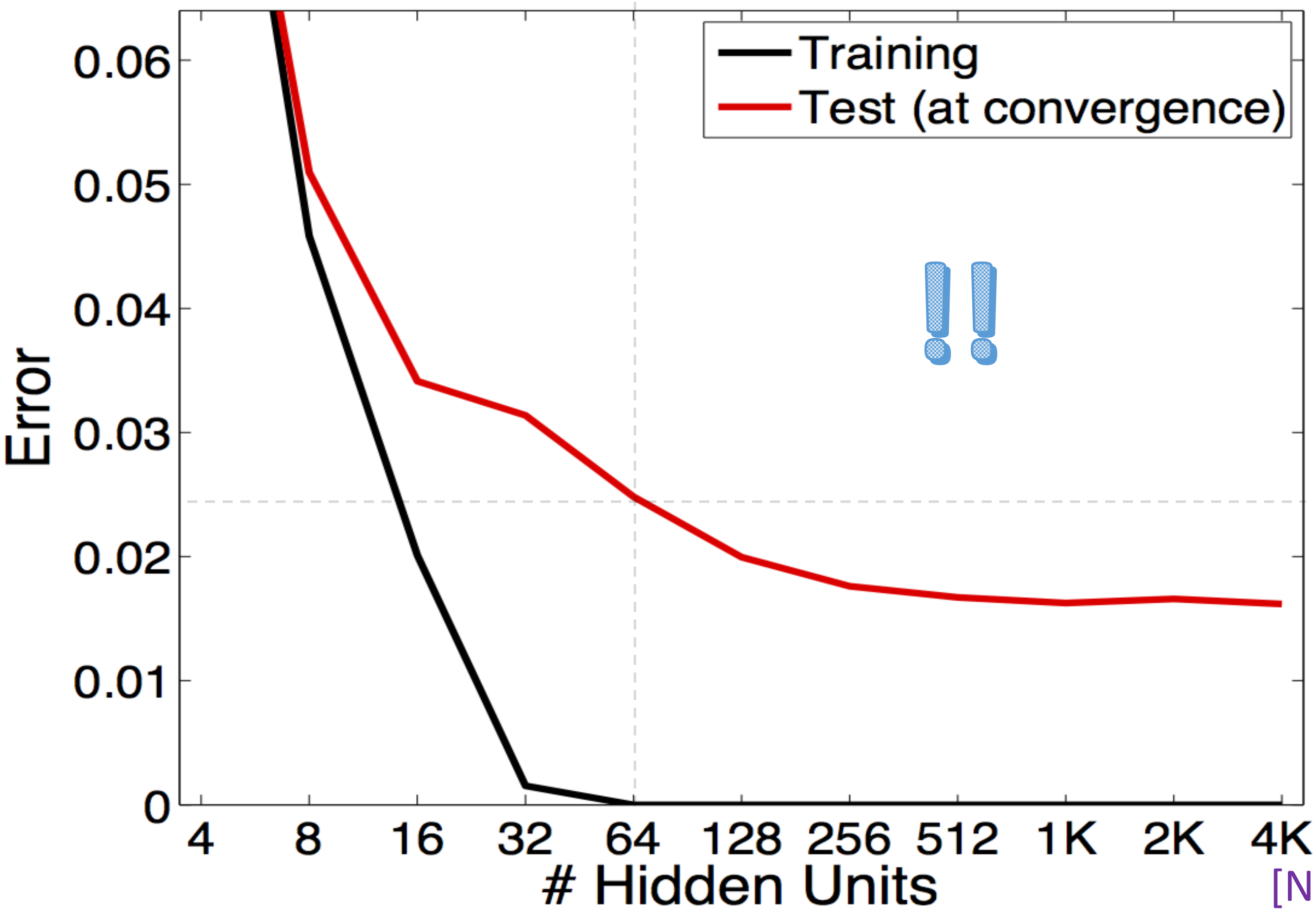
Feed Forward Neural Networks

- 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]
  - Often easy to optimize in practice, on interesting/useful problems
  - Magic property of reality that makes local search "work"

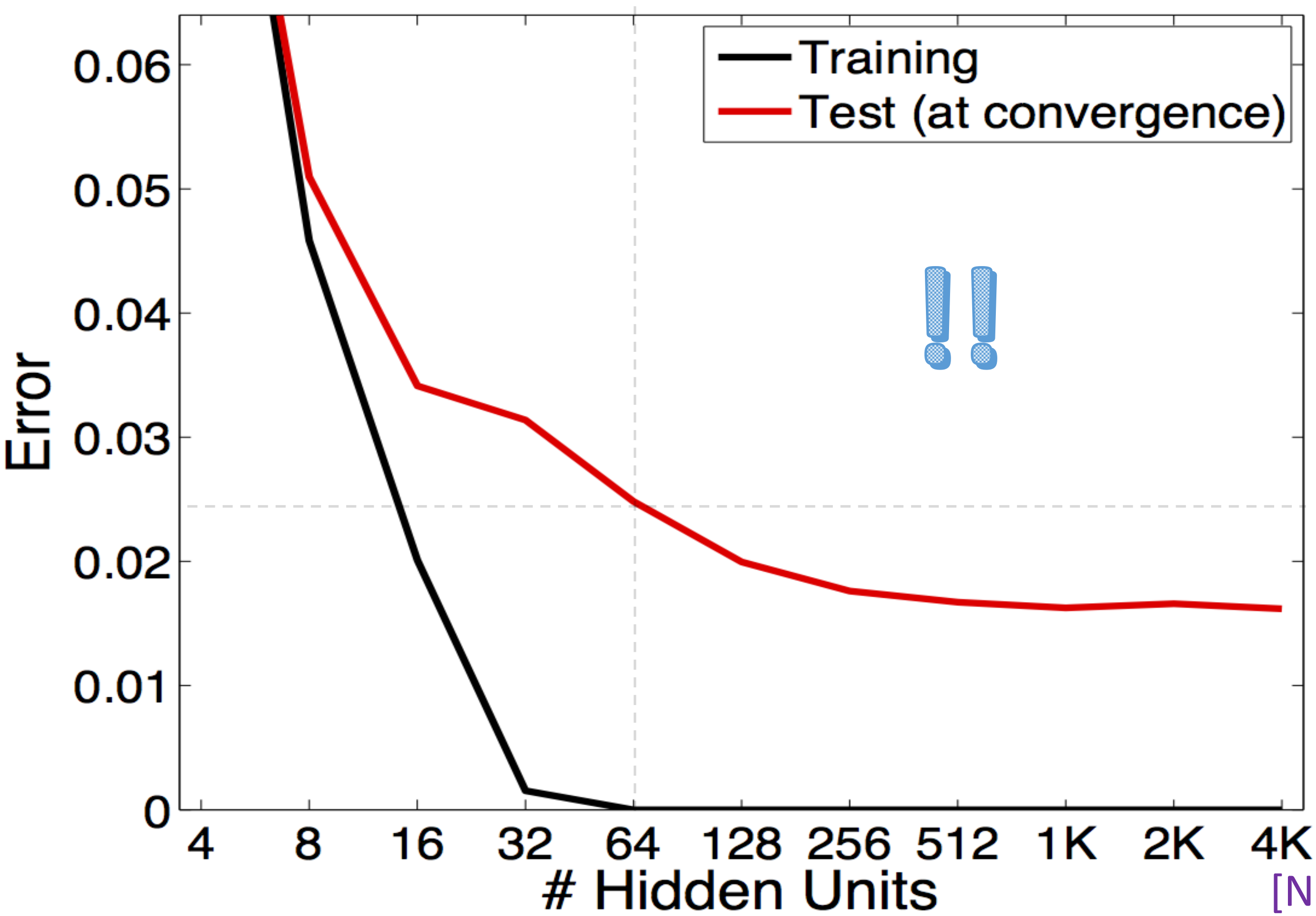
?



[Neyshabur Tomioka S ICLR’15]



[Neyshabur Tomioka S ICLR'15]

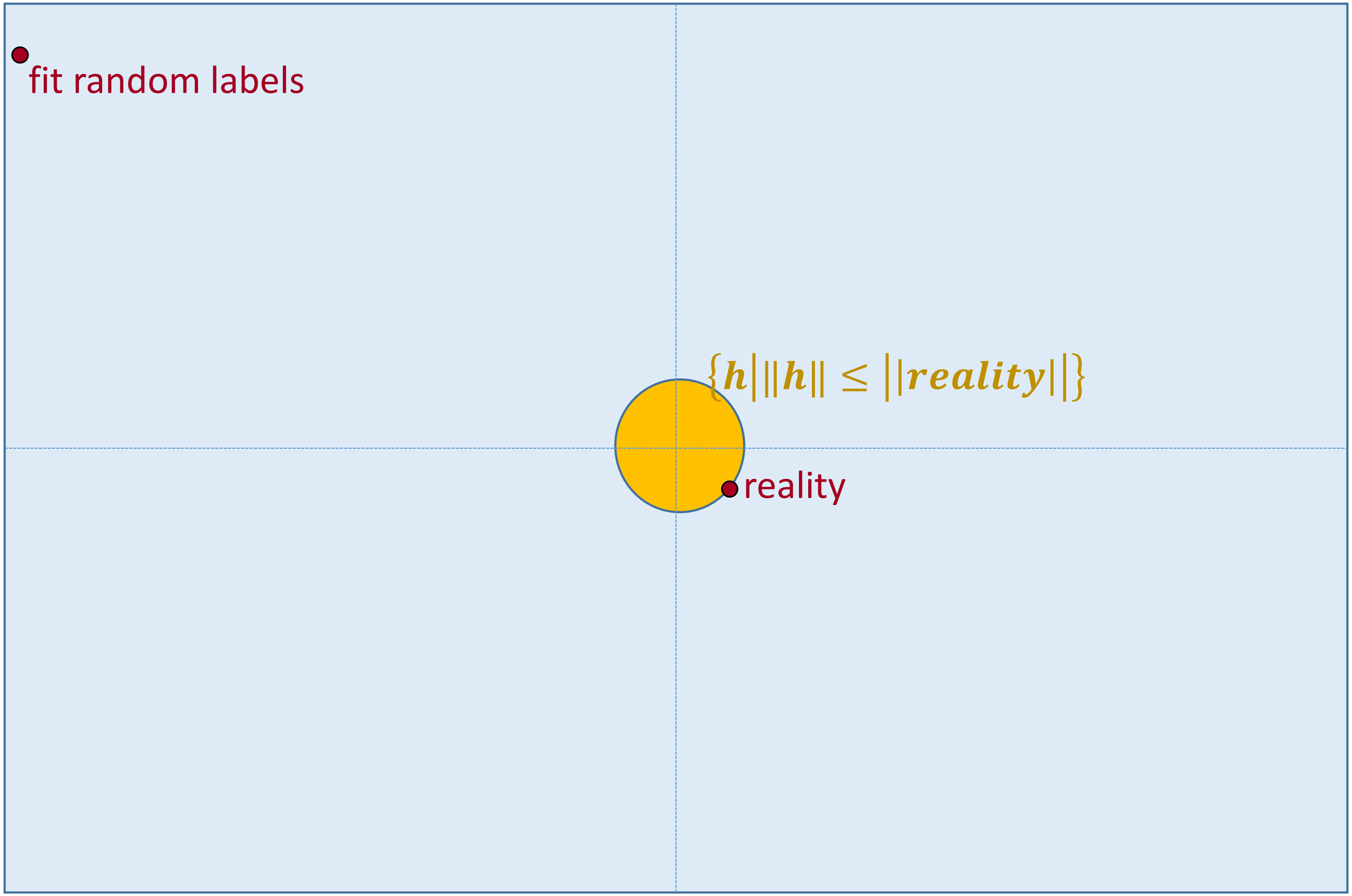


[Neyshabur Tomioka S ICLR'15]

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

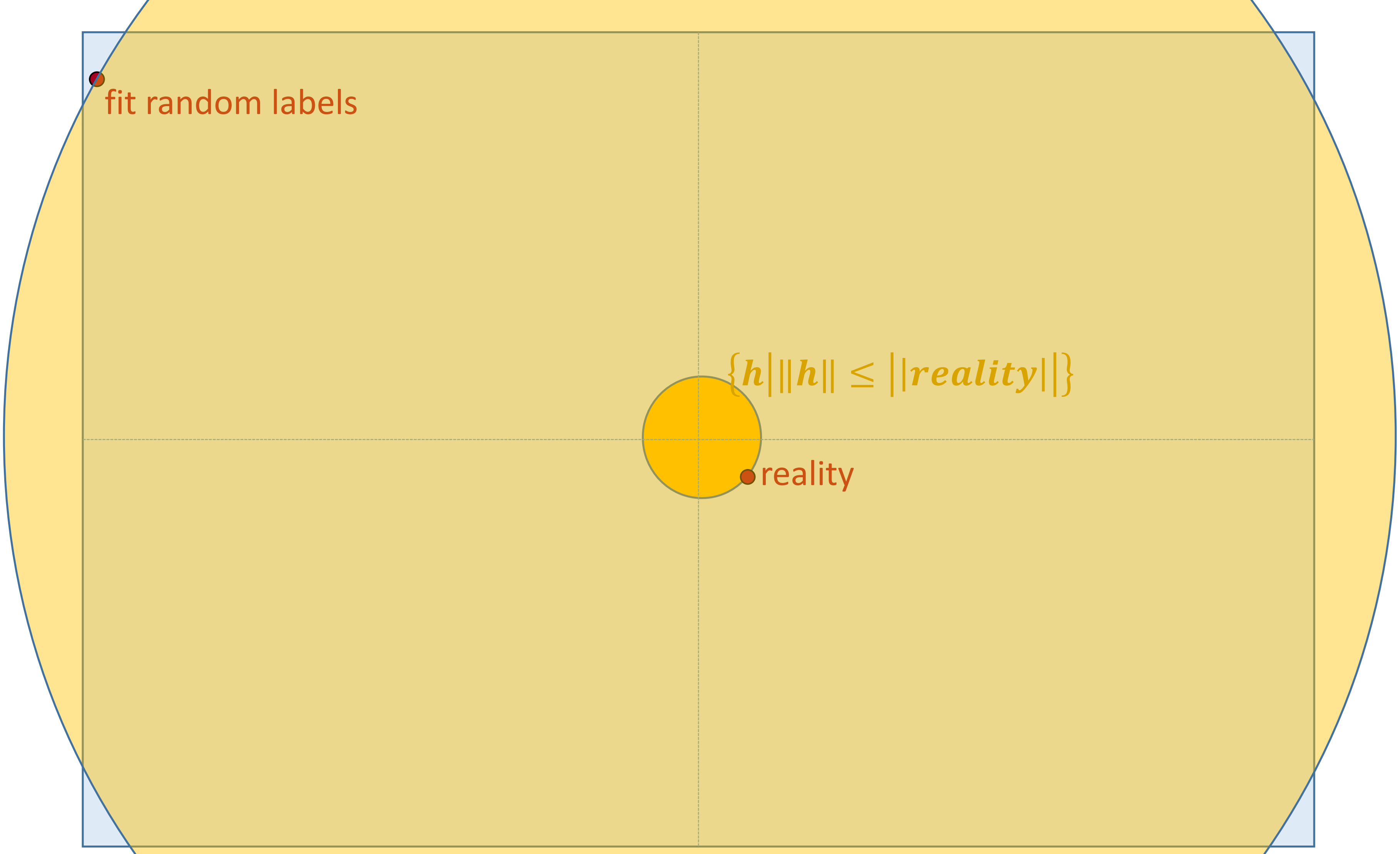
1997

Peter L. Bartlett



E.g., hard margin SVM: $\min \|w\|$ s.t. $L_S^{margin}(w) = 0$

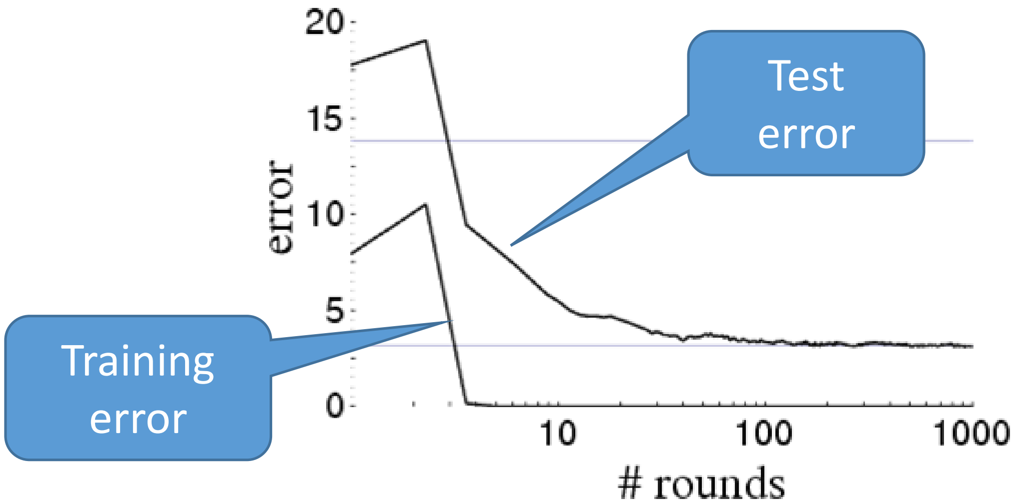
for $h_w = \langle w, \phi(x) \rangle$ with inf dim ϕ

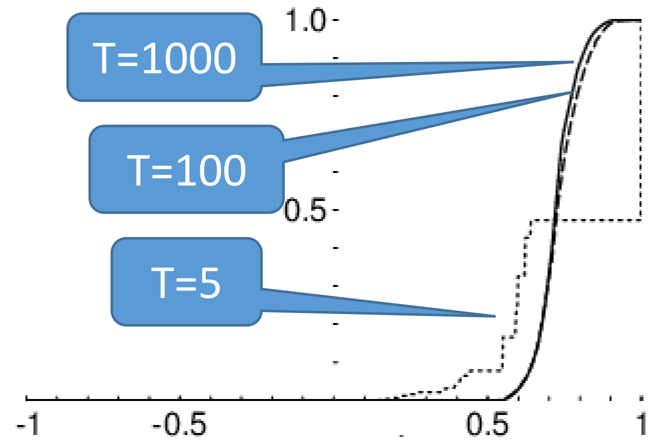


E.g., hard margin SVM: $\min \|w\|$ s.t. $L_S^{margin}(w) = 0$

for $h_w = \langle w, \phi(x) \rangle$ with inf dim ϕ

AdaBoost

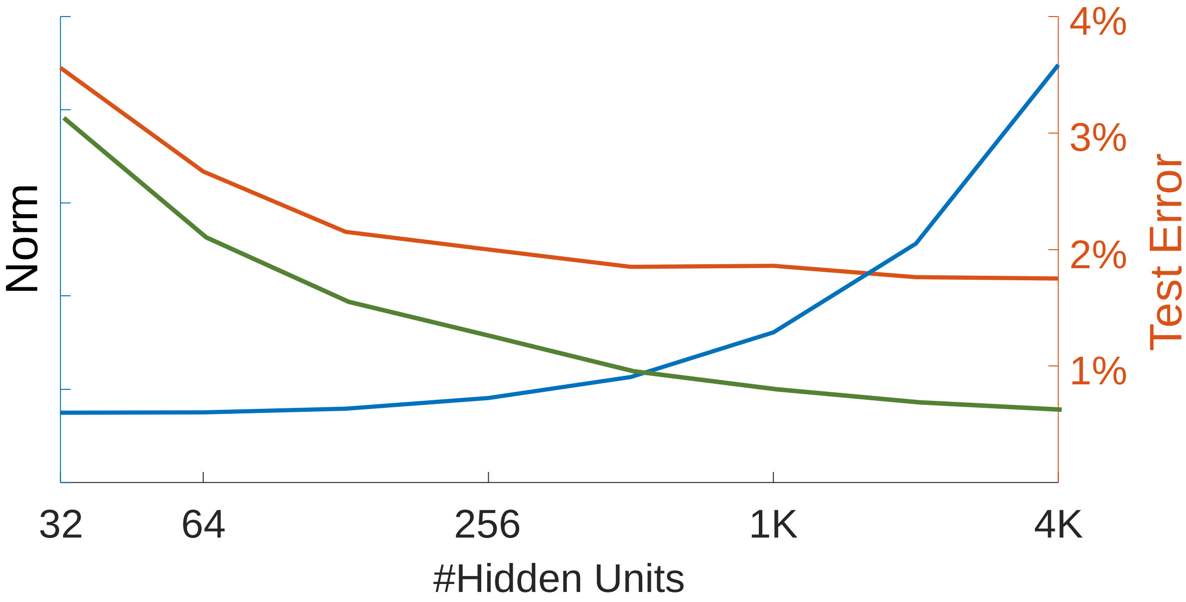




ℓ_1 margin: $\frac{y\langle w,x\rangle}{\|w\|_1}$

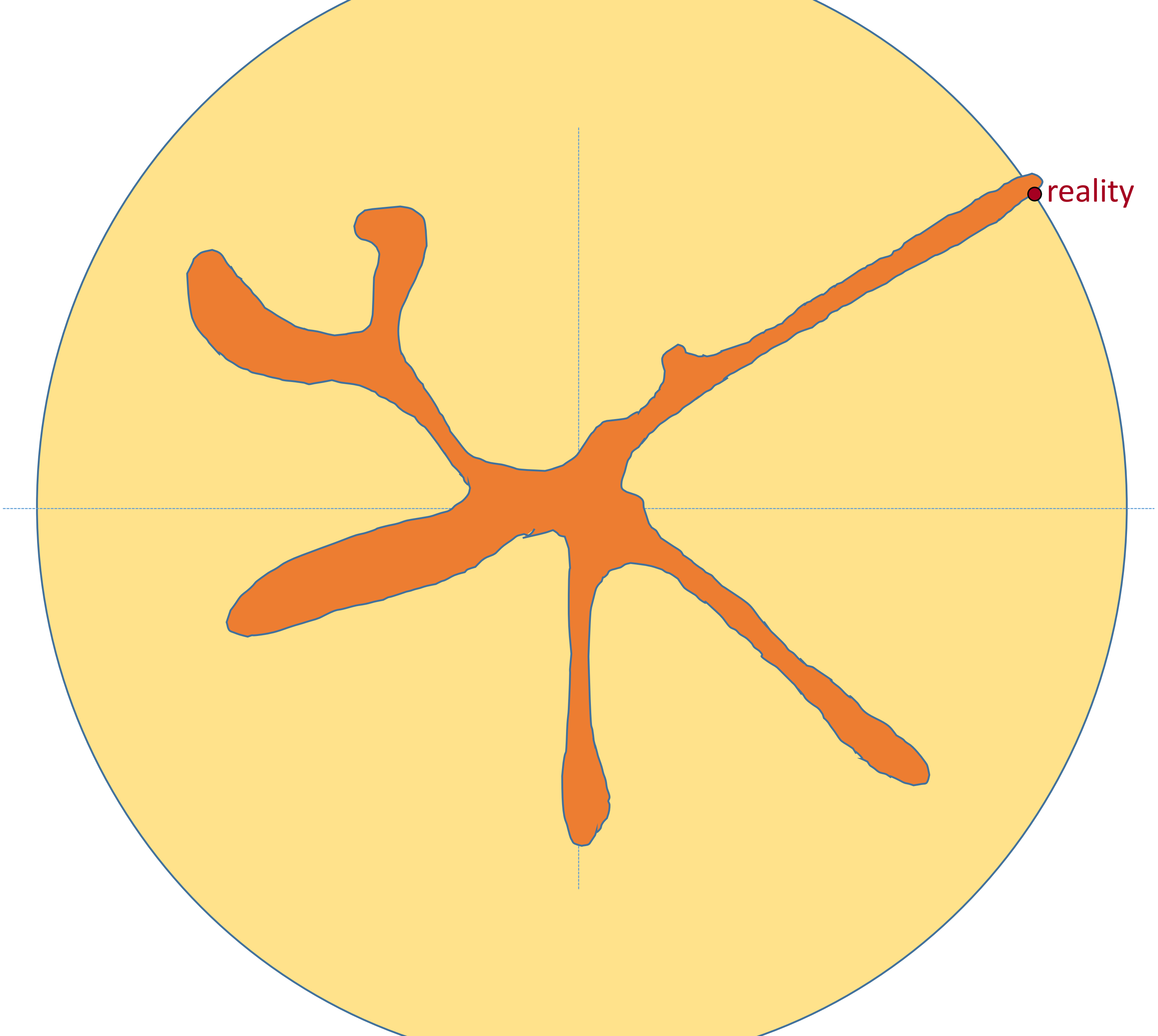
[Bartlett, Freund, Lee, Schapire 1998]

"Size of Weights" and Generalization

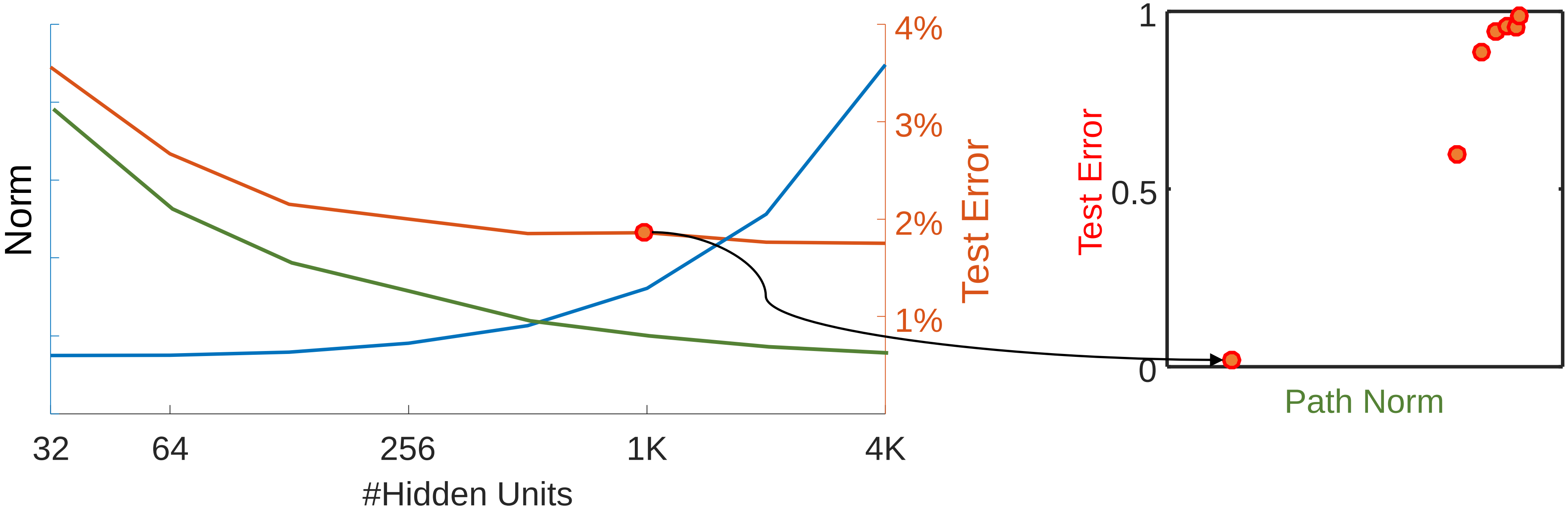


Norm = $\|W\|_2 = \sqrt{\sum_e w(e)^2}$

Path-Norm = $\sqrt{\sum_{\text{path}} \prod_{e \in \text{path}} w(e)^2}$


reality

“Size of Weights” and Generalization



Norm = $\|W\|_2 = \sqrt{\sum_e w(e)^2}$

Path-Norm = $\sqrt{\sum_{\text{path}} \prod_{e \in \text{path}} w(e)^2}$

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

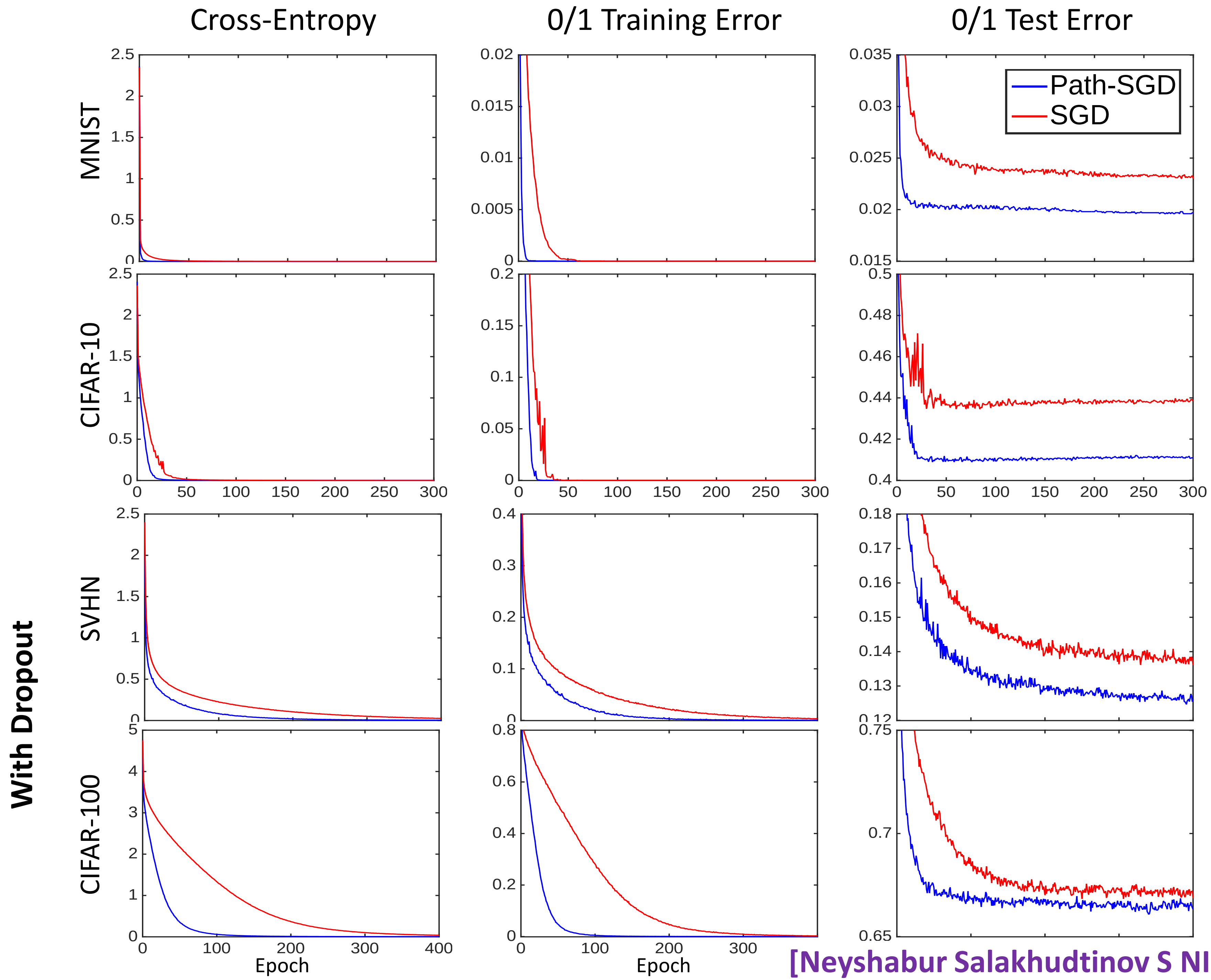
Where is the Regularization?

- What we did: minimize **unregularized** error **to convergence**

- In convex models, we understand how one-pass SGD (or with *early stopping*) provides for implicit ℓ_2 regularization
  - More generally, Mirror Descent provides generalization w.r.t. any* inductive bias [S Sridharan Tewar, On the Universality of Mirror Descent, NIPS'11]
  - Inductive Bias ⇔ choice of potential for Mirror Descent

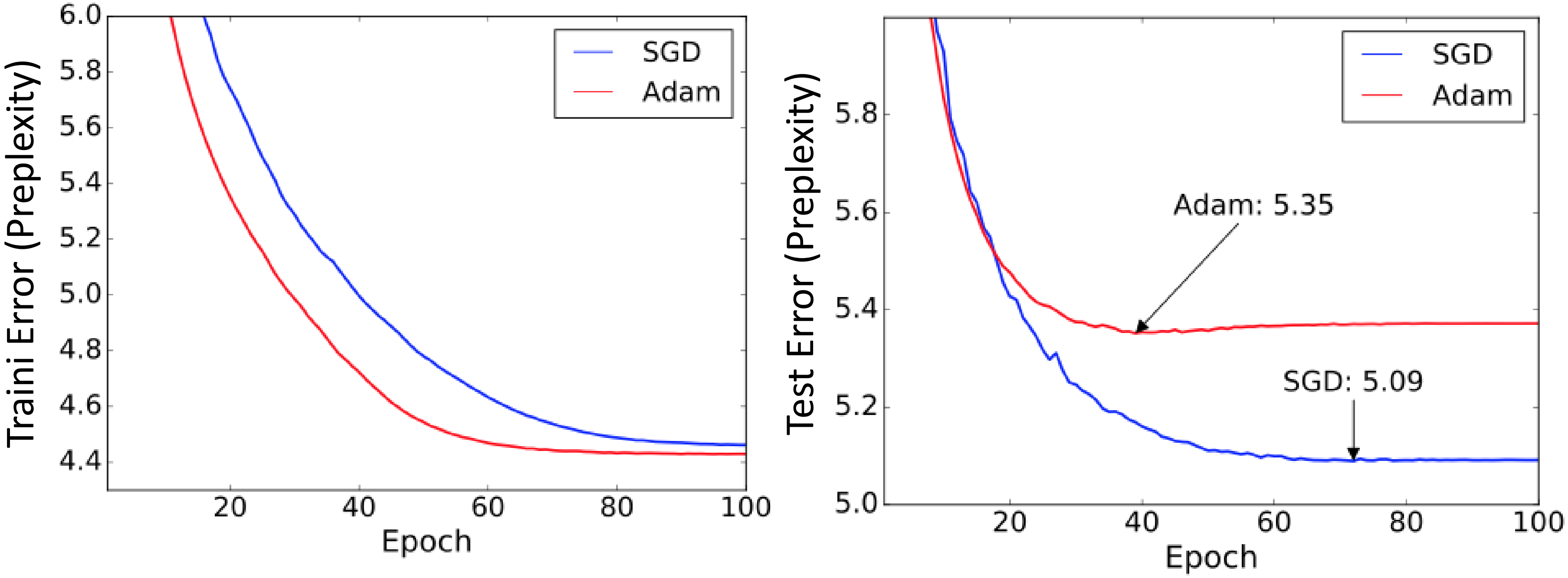
- Here: implicit regularization, **without early stopping**, and even with deterministic optimization
- In underdetermined problem (lots of global optima), optimization is biasing us toward specific global optimum.

Different optimization algorithm
➔ Different Bias
➔ Different generalization properties



[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]

Simple Example: Least Squares

- Consider an under-constraint least-squares problem ($n < m$):

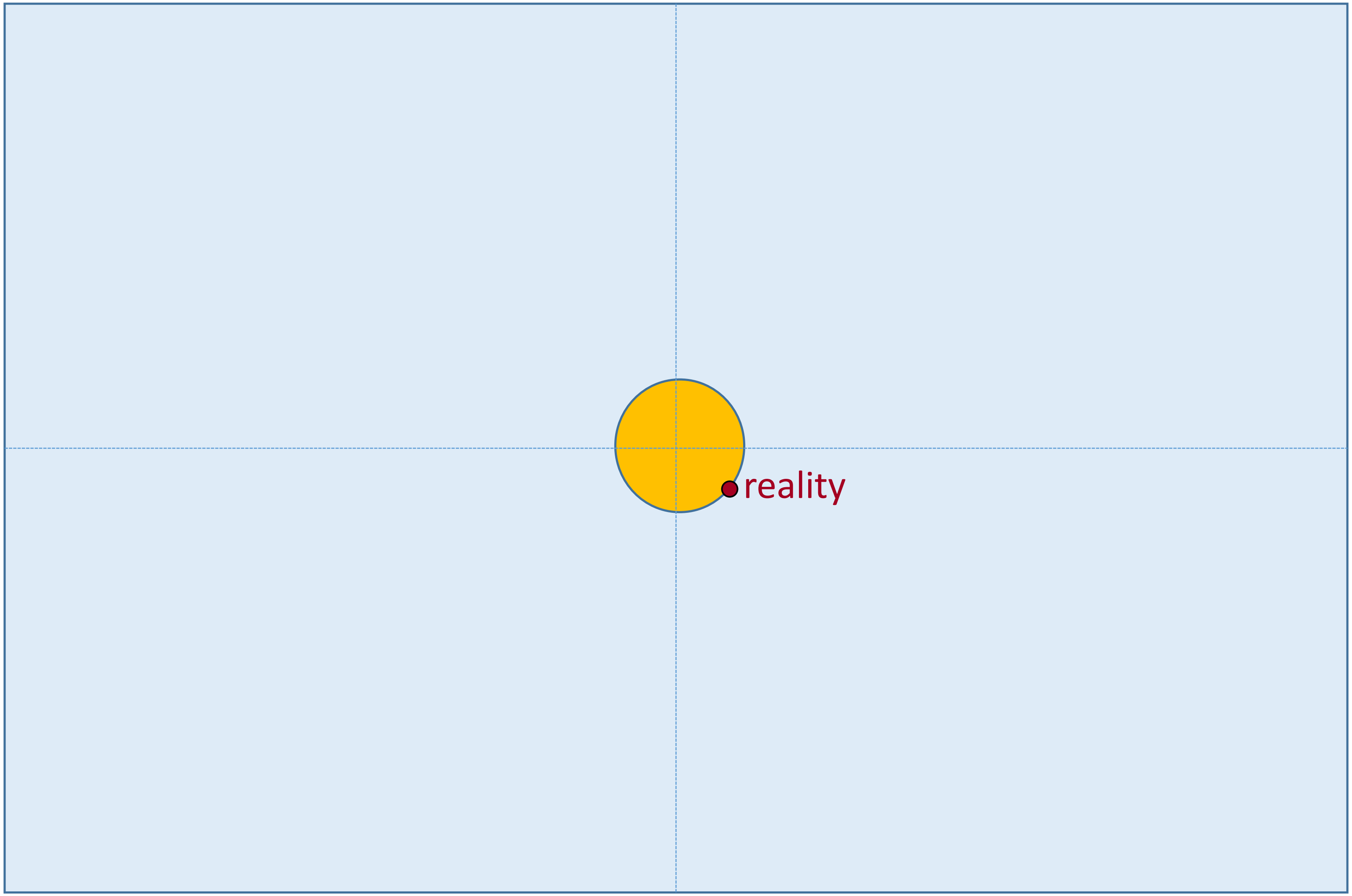
$$\min_{w \in \mathbb{R}^n} \| Aw - b \|^2$$

$A \in \mathbb{R}^{m \times n}$

- 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)


reality

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:**

$$\mathbf{arg\,min}_{\boldsymbol{w}} \frac{\mathbf{1}}{\boldsymbol{m}} \sum_{\boldsymbol{i}=\mathbf{1}}^{\boldsymbol{m}} \boldsymbol{loss}(\boldsymbol{h}_{\boldsymbol{w}}(\boldsymbol{x}); \boldsymbol{y}) \qquad (*)$$

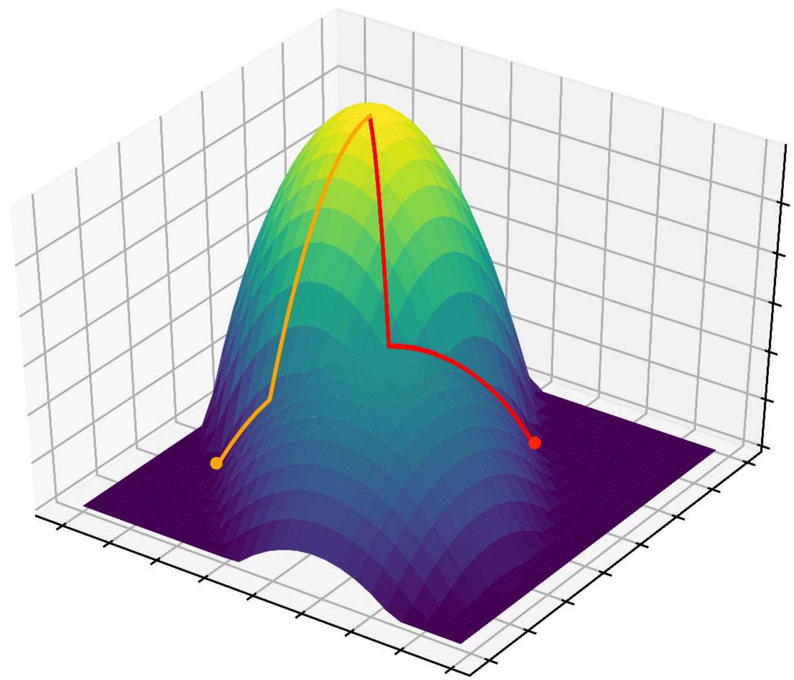
Section 3: Optimization

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

Section 4: Experiments

It works!

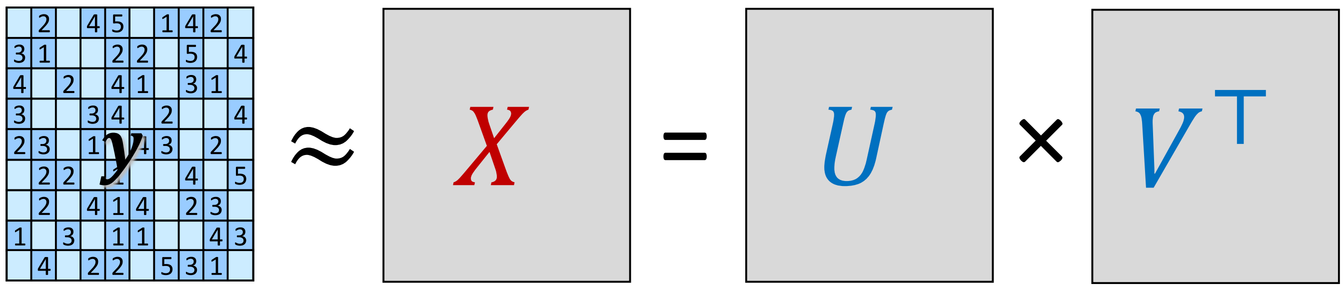
Different optimization algorithm
➔ Different bias in optimum reached
➔ Different Inductive bias
➔ Different generalization properties



To Understand Deep Learning

- **Ultimate Question**: What is the true Inductive Bias? What makes reality *efficiently* learnable by fitting a huge (infinite) neural net with a specific algorithm?
- **The "complexity measure" approach**: identify $c(h)$ s.t.
  - Reality is well explained by low $c(h)$
  - $\mathcal{H}_{c(reality)} = \{h | c(h) \leq c(reality)\}$ has low capacity
  - Opt. algorithm (with or w/o regularization?) biases towards low $c(h)$
- Mathematical questions:
  - What is the capacity (≡sample complexity) of the sublevel sets $\mathcal{H}_c$?
  - What is the bias of optimization algorithms?
- Question about reality (scientific Q?): does it have low $c(h)$?
- Alternative empirical questions:
  - Do models we actually learn have low $c(h)$?
  - Does it explain generalization?
  - Can we at least corelate generalization with $c(h)$?

Unconstrained Matrix Completion

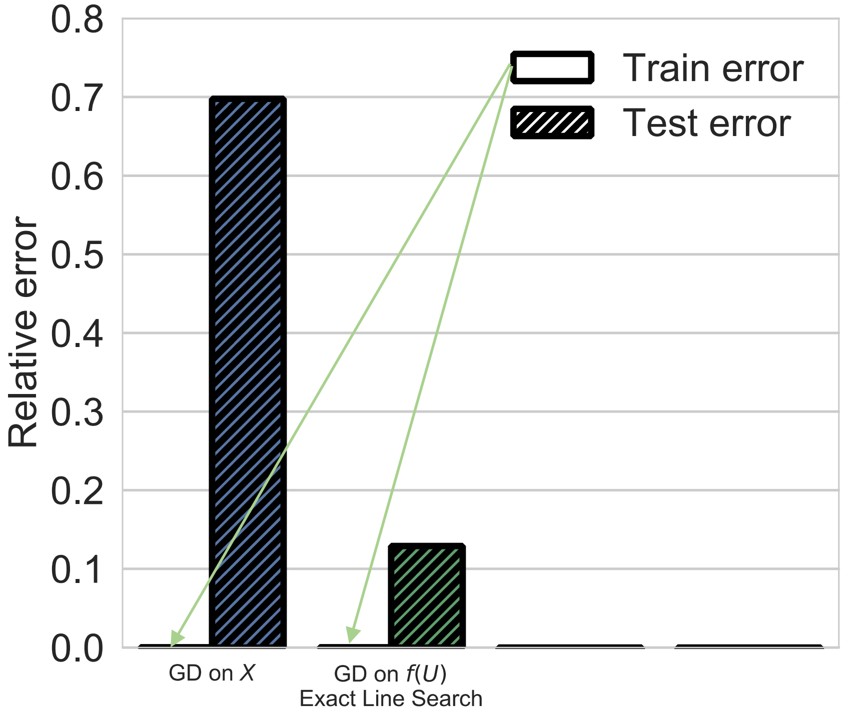


$$\min_{X\in\mathbb{R}^{n\times n}} \|observed(X) - y\|_2^2 \equiv \min_{U,V\in\mathbb{R}^{n\times n}} \|observed(UV^\top) - y\|_2^2$$

- 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

What happens when we optimize by gradient descent on U, V ?

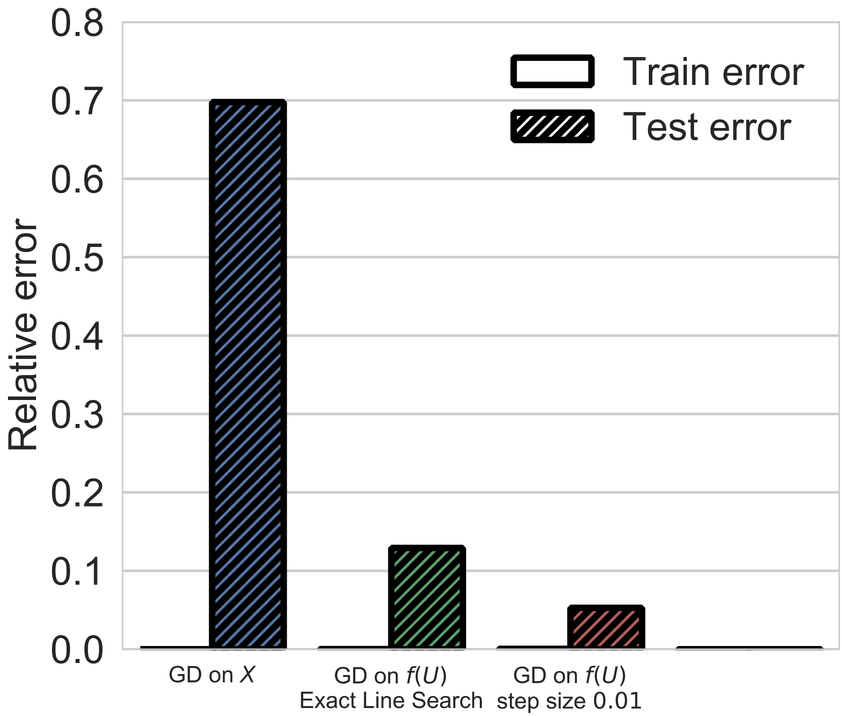
[Gunasekar Woodworth Bhojanapalli Neyshabur S 2017]



$n = 50,\ m = 300,\ A_i$ iid Gaussian, X^* rank-2 ground truth
$y = \mathcal{A}(X^*) + \mathcal{N}(0, 10^{-3}),\ y_{\text{test}} = \mathcal{A}_{\text{test}}(X^*) + \mathcal{N}(0, 10^{-3})$

Gradient descent on $f(U,V)$ gets to "good" global minima

[Gunasekar Woodworth Bhojanapalli Neyshabur S 2017]

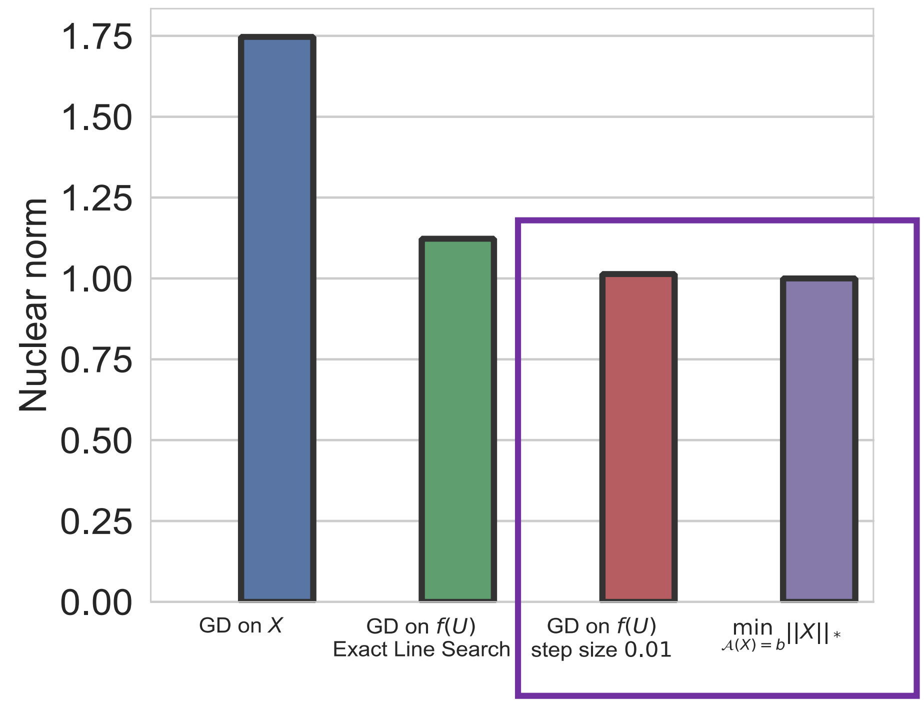


$n = 50,\ m = 300,\ A_i$ iid Gaussian, X^* rank-2 ground truth
$y = \mathcal{A}(X^*) + \mathcal{N}(0, 10^{-3}),\ y_{\text{test}} = \mathcal{A}_{\text{test}}(X^*) + \mathcal{N}(0, 10^{-3})$

Gradient descent on $f(U, V)$ gets to “good” global minima

Gradient descent on $f(U, V)$ generalizes better with smaller step size

[Gunasekar Woodworth Bhojanapalli Neyshabur S 2017]



Grad Descent on U, V → **min nuclear norm solution**

$$\arg\min \|X\|_* \ s.t. \ obs(X) = y$$

(with inf. small stepsize and initialization, exact and rigorous under additional conditions)

→ good generalization if Y (aprox) low rank

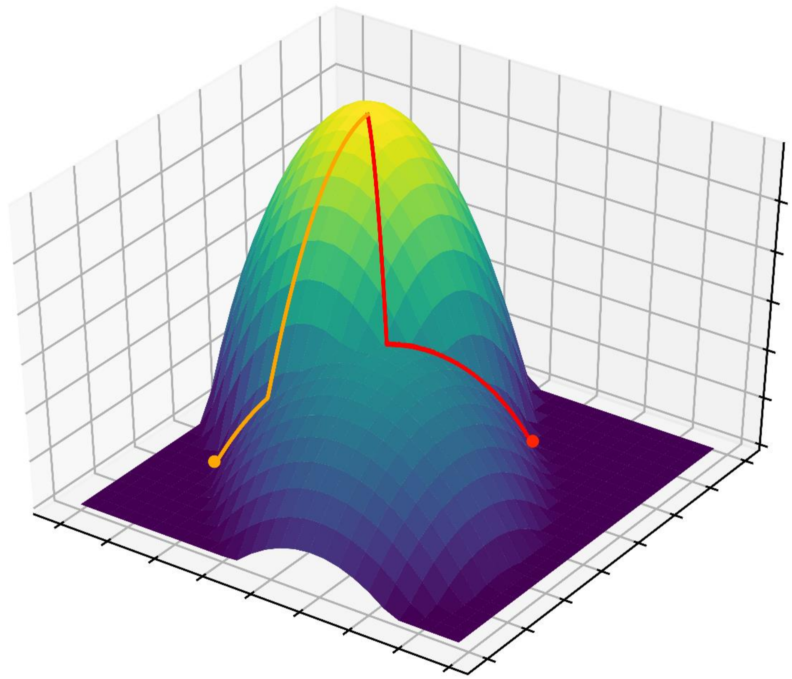
[Gunasekar Woodworth Bhojanapalli Neyshabur S 2017]

Conjecture: With stepsize→0 (i.e. gradient flow) and initialization→0, (and additional conditions?) gradient descent on U converges (approximately) to minimum nuclear norm solution:

$$UU^{\top} \rightarrow \min_{W \succcurlyeq 0} \|W\|_* \ s.t. \mathcal{A}(X) = y$$

[Gunasekar Woodworth Bhojanapalli Neyshabur S 2017]

- Rigorous proof of exact convergence:
  - when A_is commute
  - [Yuanzhi Li, Hongyang Zhang and Tengyu Ma, COLT 2018]: when $y = \mathcal{A}(\text{low rank } W^*)$, $\mathcal{A}$ RIP
- General A_i: empirical validation (approximate) + hand waving



Understand optimization algorithm not just as reaching ***some*** (global) optimum, but as reaching a ***specific*** optimum

Implicit Bias in Least Squared

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

- Gradient Descent (+Momentum) on w

  ➔ $\min_{Aw=b} \|w\|_2$

- Gradient Descent on factorization $W = UV$

  ➔ probably $\min_{A(W)=b} \|W\|_{tr}$ with stepsize$\searrow 0$ and init $\searrow 0$,
  but only in limit, depends on stepsize, init, proved only in special cases

- AdaGrad on w

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

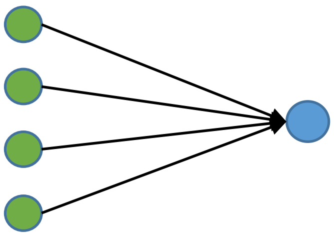
- Steepest Descent w.r.t. $\|w\|$

  ➔ ??? **Not** $\min_{Aw=b} \|w\|$, even as stepsize$\searrow 0$!
  and it depends on stepsize, init, momentum

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

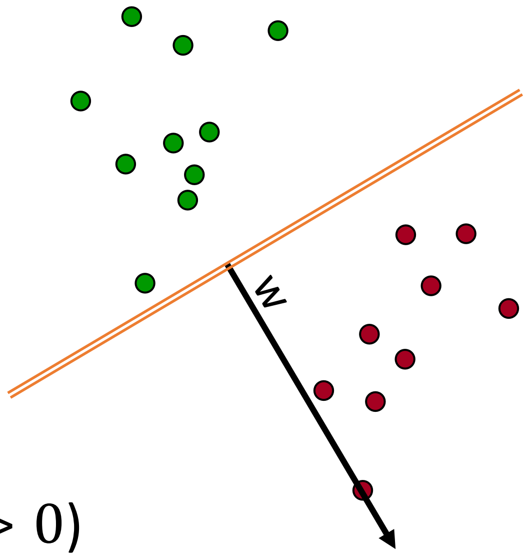
  ➔ Related to, but not quite the Lasso
  (with stepsize$\searrow 0$ and particular tie-breaking $\approx$ LARS)

Implicit Bias in Logistic Regression



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

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

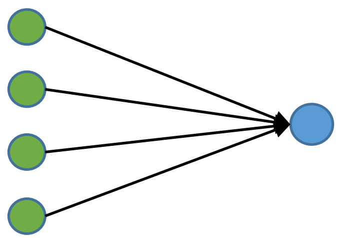


- Data $\{(x_i, y_i)\}_{i=1}^{m}$ linearly separable ($\exists_w \forall_i y_i\langle w, x_i\rangle > 0$)
- Where does gradient descent converge?

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

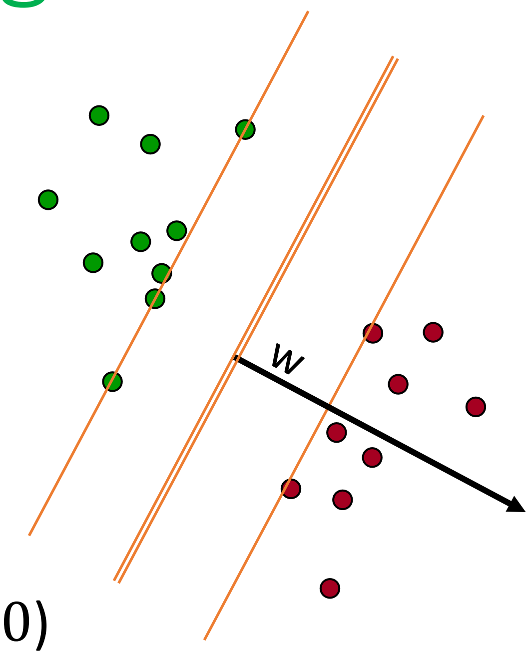
  - $\inf \mathcal{L}(w) = 0$, but minima unattainable
  - GD diverges to infinity: $w(t) \to \infty$, $\mathcal{L}\big(w(t)\big) \to 0$
- **In what direction?** What does $\frac{w(t)}{\|w(t)\|}$ converge to?

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>**: $\frac{w(t)}{\|w(t)\|_2} \to \frac{\hat{w}}{\|\hat{w}\|_2}$ $\quad \hat{w} = \arg\min \|w\|_2 \ s.t. \forall_i y_i\langle w, x_i\rangle \geq 1$

Logistic Regression on Separable Data

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

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

Theorem: $\frac{w(t)}{\|w(t)\|_2} \to \frac{\hat{w}}{\|\hat{w}\|_2}$ $\quad \hat{w} = \arg\min\|w\|_2 \;\; s.t. \forall_i y_i\langle w, x_i\rangle \geq 1$

- $w(t) = \hat{w}\log t + \rho(t)$, with $\rho(t)$ bounded*
- Holds for any initial point $w(0)$ and stepsize $\eta \leq 2$
- Holds for any monotonically decreasing strictly positive smooth loss s.t. $-\ell'(z)$ has a tight exponential tail

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

[Soudry Hoffer S ICLR 2018]

Proof sketch: ($y_i = 1$ w.l.og.)

Write $w(t) = g(t)w_\infty + \rho(t)$ with $g(t) \to \infty$ and $\rho(t) = o\big(g(t)\big)$.

Since we converge to zero error, $\forall_i \langle w_\infty, x_i\rangle > 0$

Since the loss derivative has an exponential tail:

$$-\nabla\mathcal{L}(w) \approx \sum_i e^{-\langle w(t), x_i\rangle} x_i^\top = \sum_i e^{-g(t)\langle w_\infty, x_i\rangle - \langle \rho(t), x_i\rangle} x_i^\top$$

As $g(t) \to \infty$, only points with minimal $\langle w_\infty, x_i\rangle$ (points on the margin, "support vectors") will dominate gradient

➔ $\nabla\mathcal{L}(w)$ spanned by support vectors

➔ $w(t)$ spanned by support vectors

Define $\hat{w} = \frac{w_\infty}{\min_i \langle w_\infty, x_i\rangle}$. We have:

$\hat{w} = \sum \alpha_i w_i \qquad \forall_i (\alpha_i \geq 0 \text{ and } \langle \hat{w}, x_i\rangle = 1) \text{ OR } (\alpha_i = 0 \text{ and } \langle \hat{w}, x_i\rangle > 1)$

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:

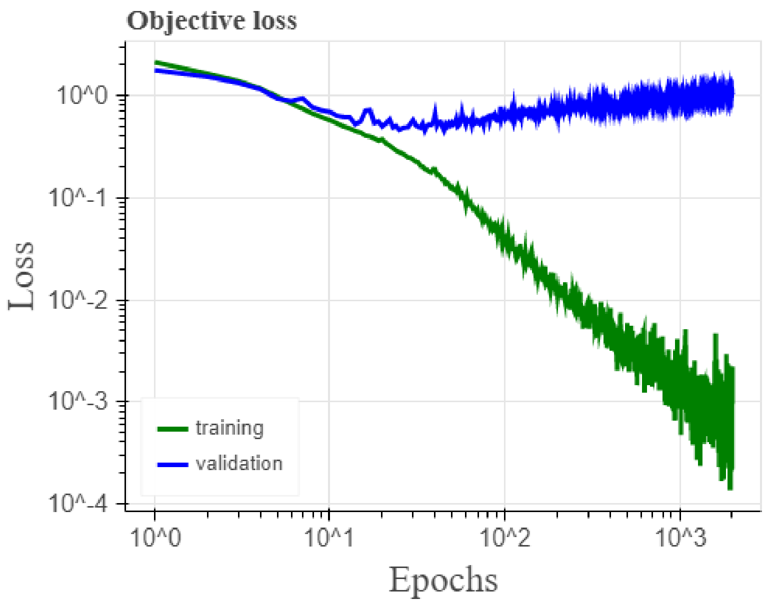
$$\max_{\|w\|\leq 1} \min_i y_i \langle w, x_i \rangle - \min_i y_i \left\langle \frac{w(t)}{\|w(t)\|}, x_i \right\rangle = O\left(\frac{1}{\log t}\right)$$

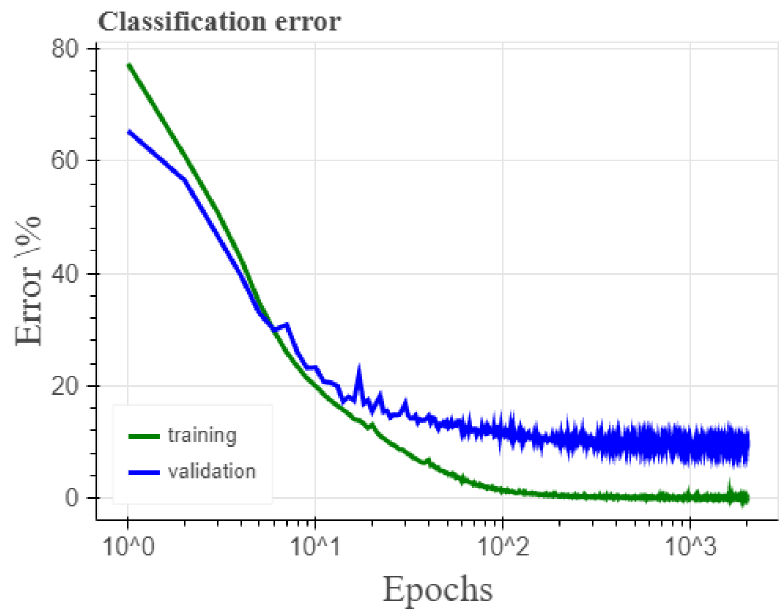
Contrast with convergence of the loss:

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

➔ 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%

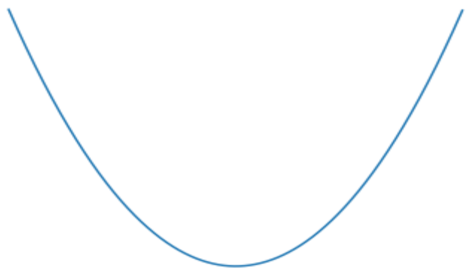
Training a conv net using SGD+momentum on CFAIR10

Other Objectives and Opt Methods

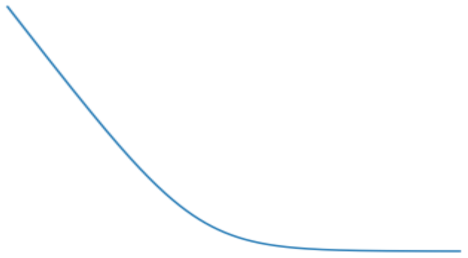
- Single linear unit, logistic loss
  $\Rightarrow$ hard margin SVM solution (regardless of init, stepsize)
- Multi-class problems with softmax loss
  $\Rightarrow$ multiclass SVM solution (regardless of init, stepsize)
- Steepest Descent w.r.t. $\|w\|$
  $\Rightarrow$ $\arg\min \|w\| \;\; s.t. \forall_i y_i \langle w, x_i \rangle \geq 1$ (regardless of init, stepsize)
- Coordinate Descent
  $\Rightarrow$ $\arg\min \|w\|_1 \;\; s.t. \forall_i y_i \langle w, x_i \rangle \geq 1$ (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
  $\Rightarrow$ $\arg\min \|W\|_{tr} \;\; s.t. \langle A_i, W \rangle \geq 1$ (regardless of init)

Different Asymptotics

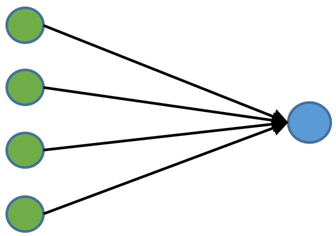
- For least squares (or any other loss with attainable minimum):
  - w_∞ depends on initial point w_0 and stepsize η
  - To get clean characterization, need to take $\eta \to 0$
  - If 0 is a saddle point, need to take $w_0 \to 0$



- 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**

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

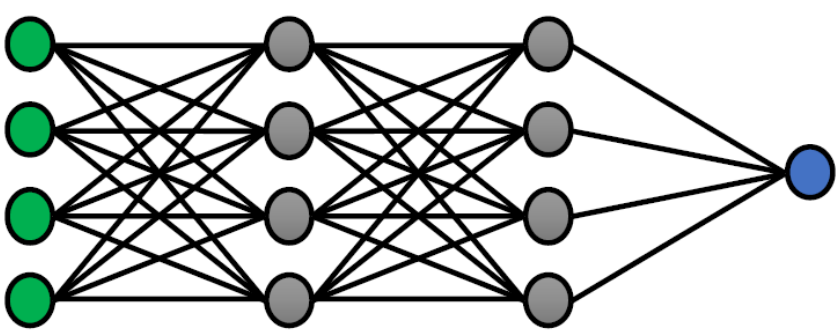
Even More Overparameterization: Deep Linear Networks

Network implements a linear mapping:

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

Training: same opt. problem as logistic regression:

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

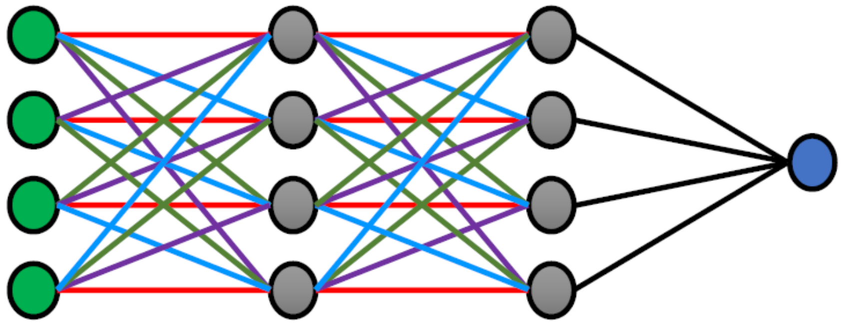


Train w with SGD

→ **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

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

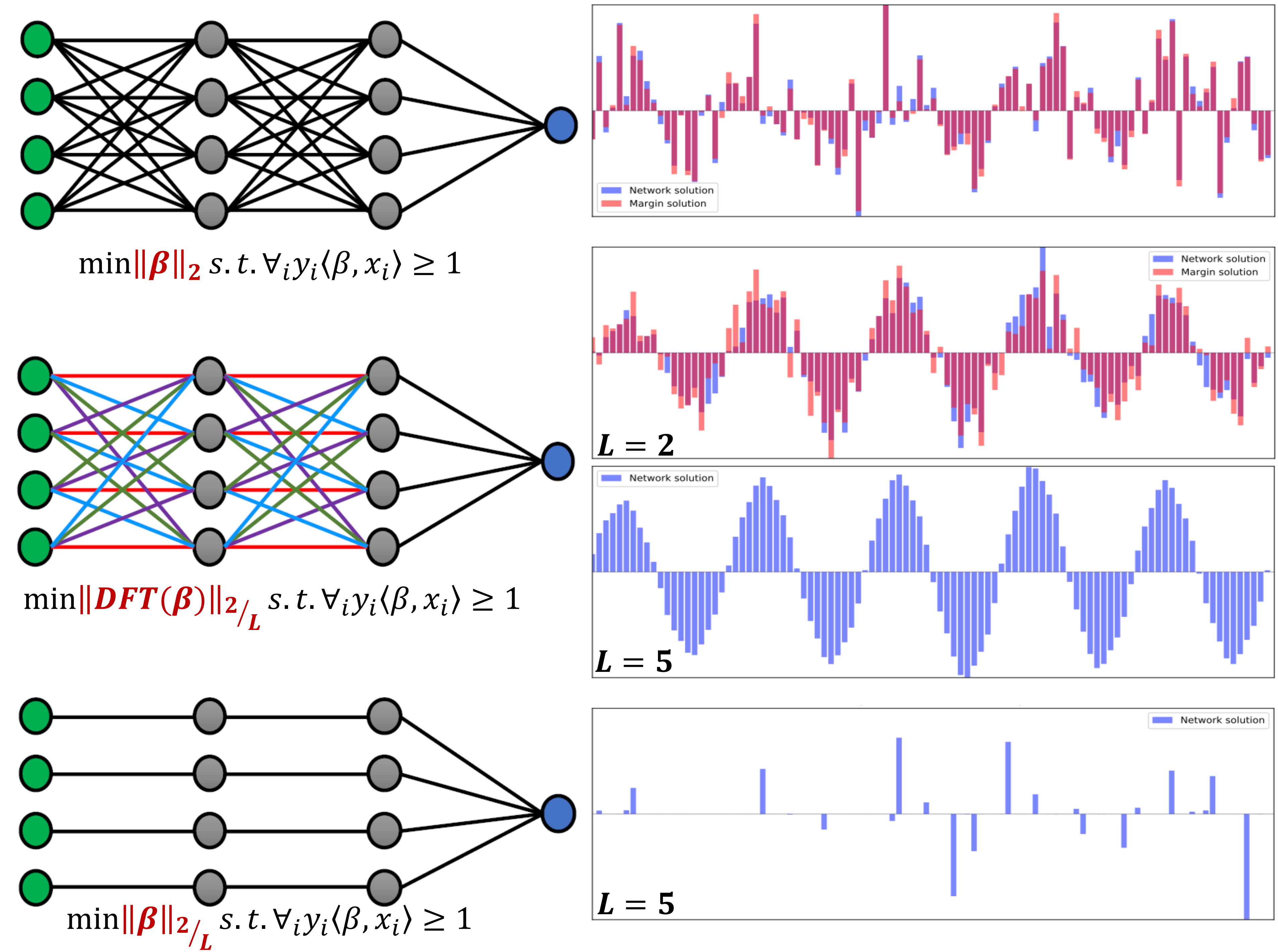
Discrete Fourier Transform

With multiple conv layers

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

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]


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

Effect of Parametrization

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

[Gunasekar Woodworth Bhojanapalli Neyshabur S 2017]

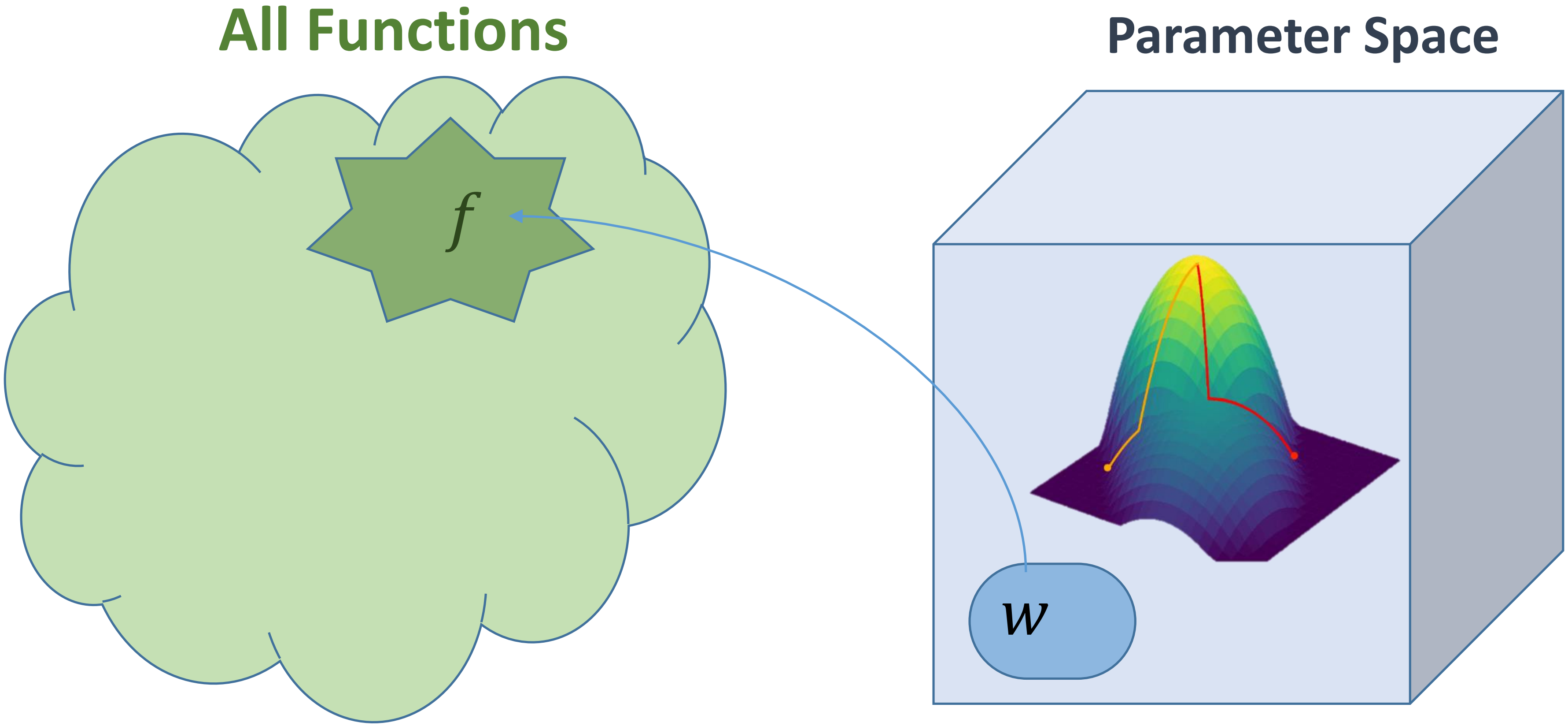
- Linear Convolutional Network:
  - Complex over-parametrization of linear predictors β
  - GD on weight ➔ implicitly minimize $\|\boldsymbol{DFT}(\boldsymbol{\beta})\|_{\boldsymbol{p}}$ for $p = \frac{2}{depth}$. (sparsity in frequency domain)

[Gunasekar Lee Soudry S 2018]

- Infinite Width ReLU Net with 1-d input:
  - Parametrization of essentially all functions $f: \mathbb{R} \to \mathbb{R}$
  - Weight decay ➔ implicitly minimize…

$$\max\left(\int |\boldsymbol{f''}| \boldsymbol{dx} \;, |f'(-\infty) + f'(+\infty)| \right)$$

[Savarese Evron Soudry S 2019]



Optimization Geometry and hence Inductive Bias effected by:

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

To Understand Deep Learning

- **Ultimate Question**: What is the true Inductive Bias? What makes reality *efficiently* learnable by fitting a huge (infinite) neural net with a specific algorithm?
- **The "complexity measure" approach**: identify $c(h)$ s.t.
  - Reality is well explained by low $c(h)$
  - $\mathcal{H}_{c(reality)} = \{h|c(h) \leq c(reality)\}$ has low capacity
  - **Opt. algorithm (with or w/o regularization?) biases towards low $\boldsymbol{c(h)}$**
- Mathematical questions:
  - What is the capacity (≡sample complexity) of the sublevel sets $\mathcal{H}_c$?
  - **What is the bias of optimization algorithms?**
- Question about reality (scientific Q?): does it have low $c(h)$?
- Alternative empirical questions:
  - Do models we actually learn have low $c(h)$?
  - Does it explain generalization?
  - Can we at least corelate generalization with $c(h)$?