Learning, Optimization and Generalization

Nati Srebro

TTI-Chicago

Menu

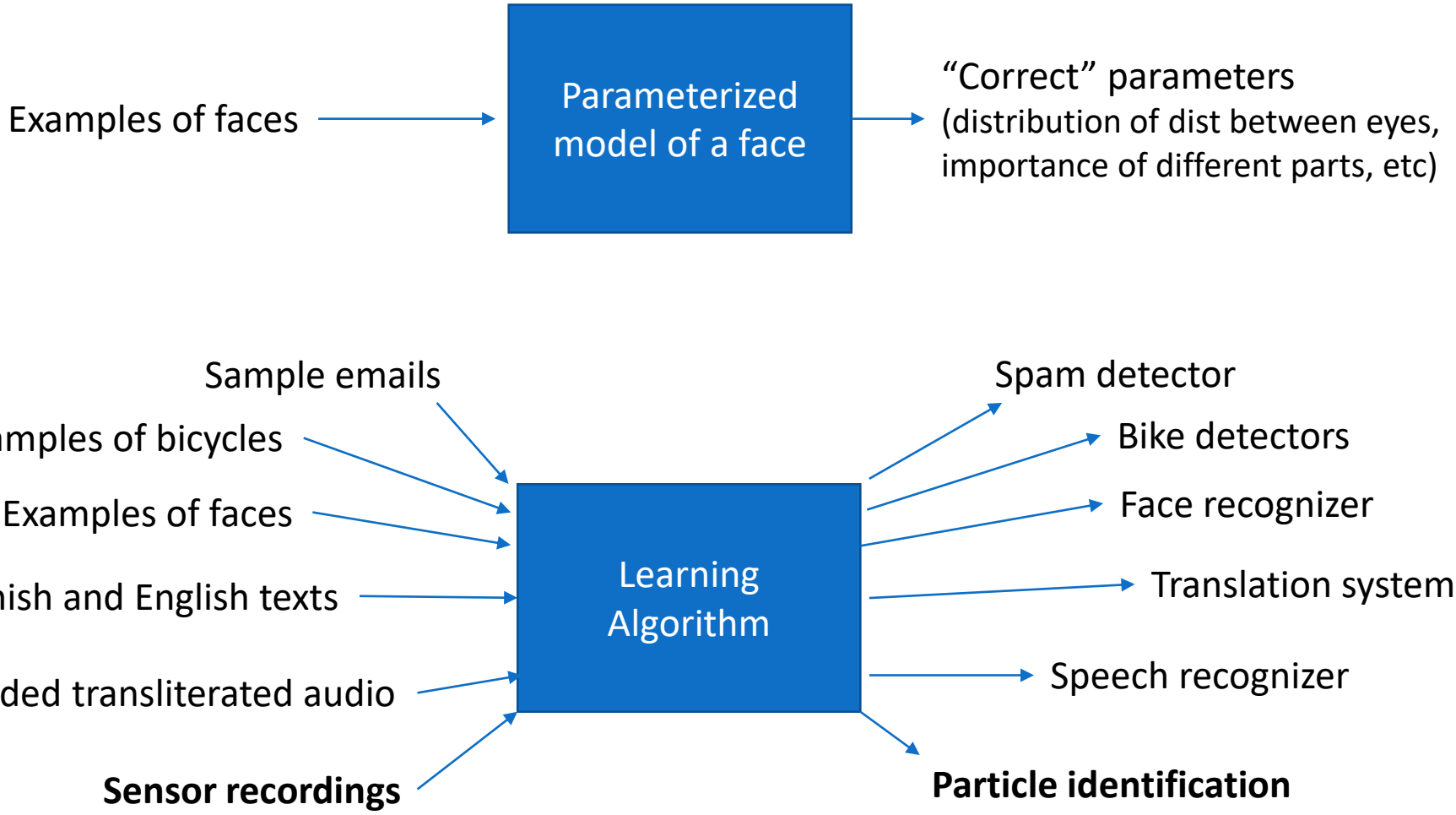
- Part I: (my view of) Learning
  … and Optimization

- Part II: Deep Learning and Optimizaiton

What is “Machine Learning”?

“Machine Learning” as an Engineering Paradigm: Use data and examples, instead of expert knowledge, to automatically create systems that perform complex tasks

Generic Learning



The ability to learn grammars is **hard-wired** into the brain. It is not possible to "learn" linguistic ability—rather, we are born with a brain apparatus specific to language representation.

There exists some "universal" learning algorithm that can learn ***anything***: language, vision, speech, etc. The brain is based on it, and we're working on uncovering it. (Hint: the brain uses neural networks)

There is no "free lunch": no learning is possible without *some* prior assumption about the structure of the problem (prior knowledge)

More Data, Less Expert Knowledge

Expert knowledge:
full specific knowledge

Machine Learning

none

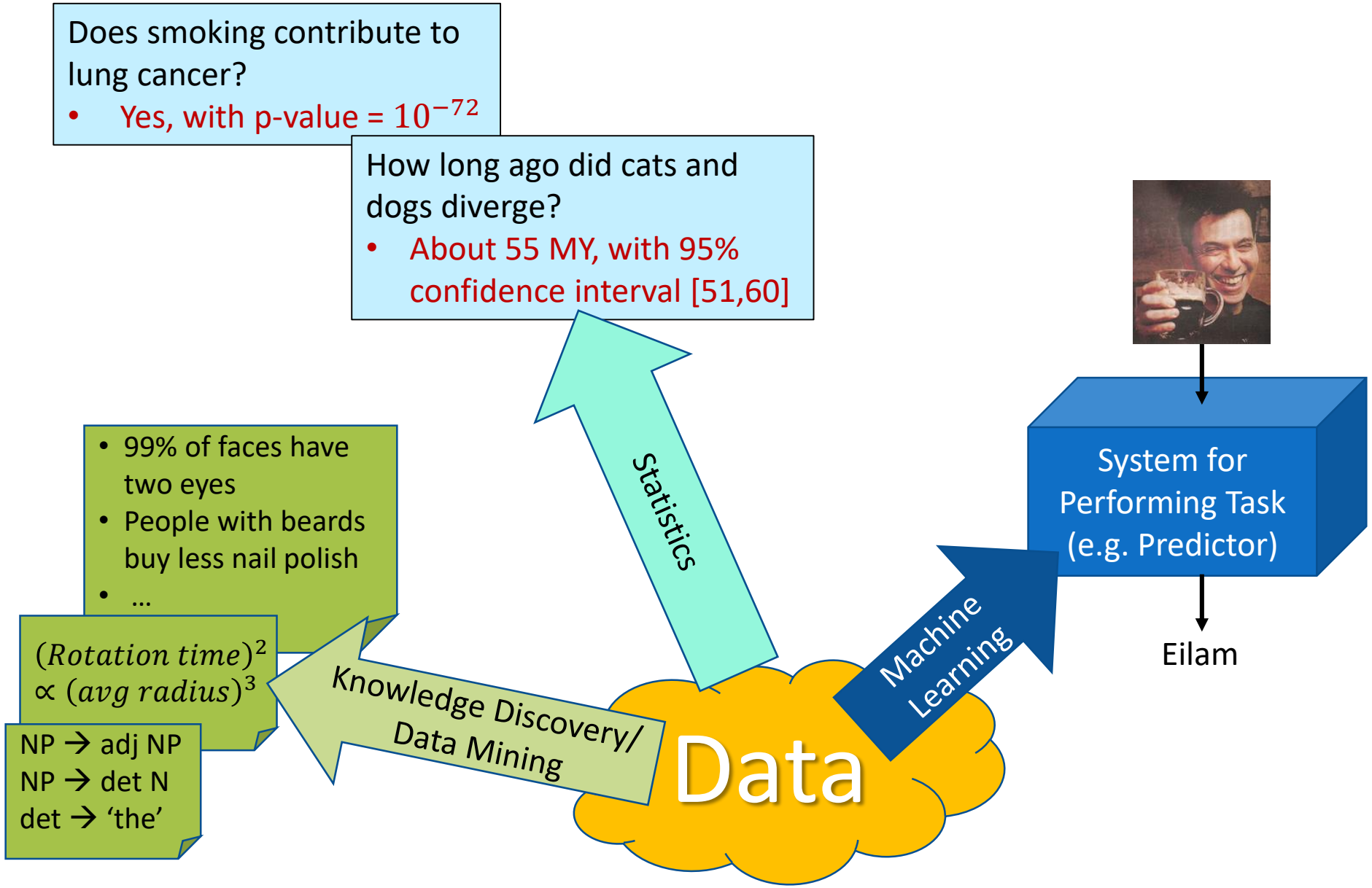
more data →

Use data to fit
specific model

Expert Systems
(no data at all)

no free lunch

"Machine Learning": Use data and examples, instead of expert knowledge, to automatically create systems that perform complex tasks



Learning and Optimization

- Optimization *for* learning:
  - Modeling: Choose hypothesis class and/or regularizer
  - Optimization: Optimize empirical objective (on training data)

$$\hat{h} = \arg\min_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} loss(h(x_i), y_i) + Reg(h)$$

  - Statistics: If $\mathcal{H}$ small or $Reg(h)$ low, $\hat{h}$ also has low generalization error

- Stochastic optimization (eg SGD) for learning:

$$w^{(t+1)} \leftarrow w^{(t)} - \eta_t \nabla_w loss\left(h_{w^{(t)}}\left(x_{i_t}, y_{i_t}\right)\right)$$

- Learning ***is*** (stochastic) optimization:

$$\min_h E_{x,y \sim \mathcal{D}}[loss(h(x), y)]$$

based on sample $(x_1, y_1), \ldots, (x_n, y_n) \sim \mathcal{D}$

Stochastic Optimization Setting

$$\min_{w \in \mathcal{W}} F(w) = \mathbb{E}_{z \sim \mathcal{D}}[f(w, z)]$$

based only on stochastic information on F

- Only unbiased estimates of $F(w), \nabla F(w)$
- No direct access to F

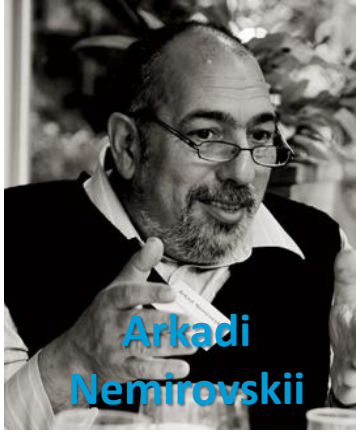
E.g., fixed $f(w, z)$ but $\mathcal{D}$ unknown

- Optimize $F(w)$ based on iid sample $z_1, z_2, \ldots, z_m \sim \mathcal{D}$
- $g = \nabla f(w, z_t)$ is unbiased estimate of $\nabla F(w)$

- Traditional applications
  - Optimization under uncertainty
    - Uncertainty about network performance
    - Uncertainty about client demands
    - Uncertainty about system behavior in control problems
  - Complex systems where its easier to sample then integrate over z

Stochastic Optimization ≡ "Generalized Learning"

$$\min_h \mathbb{E}_{z\sim\mathcal{D}}[f(h,z)]$$

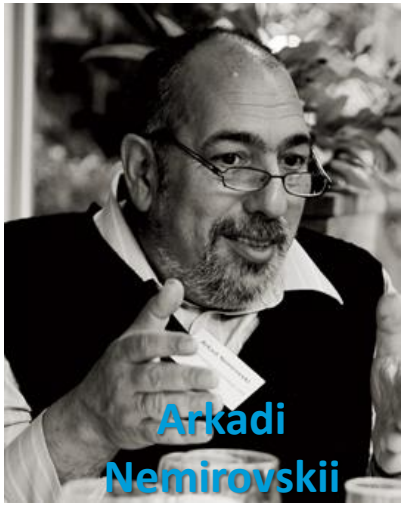


- Supervised learning: $z = (x, y)$
  h specifies a predictor $h: \mathcal{X} \to \mathcal{Y}$
  $f(h; (x, y)) = loss(h(x), y)$
- k-means clustering: $z = x \in \mathbb{R}^d$
  $h = (\mu[1], \mu[2], \ldots, \mu[k])$ specify k centers
  $f\big((\mu[1], \mu[2], \ldots, \mu[k]); x\big) = \min_j \|\mu[j] - x\|^2$
- Density estimation: h specifies probability density $p_h(x)$
  $f(h; x) = -\log p_h(x)$
- More general learning: z = traffic delays on each road segment
  h = route chosen (indicator over road segments)
  $f(h; z) = \langle z, h \rangle$ = total delay along route

Optimization	Statistics	COLT	NIPS
x	β	h	w

Stochastic Optimization vs Statistical Learning

- Focus on computational efficiency
- Generally assumes unlimited sampling
  - as in monte-carlo methods for complicated objectives
- Optimization variable generally a vector in a normed space
  - complexity control through norm
- Mostly convex objectives



- Focus on sample size
- What can be done with a fixed number of samples?
- Abstract hypothesis classes
  - linear predictors, but also combinatorial hypothesis classes
  - generic measures of complexity such as VC-dim, fat shattering, Radamacher
- Also non-convex classes and loss functions

Two Approaches to Stochastic Optimization / Learning

$$\min_{w \in \mathcal{W}} F(w) = \mathbb{E}_{z \sim \mathcal{D}}[f(w, z)]$$

- Empirical Risk Minimization (ERM) / Sample Average Approximation (SAA):
  - Collect sample $z_1, \ldots, z_m$
  - Minimize $F_S(w) = \frac{1}{m} \sum_i f(w, z_i)$
  - Analysis typically based on Uniform Concentration

- Stochastic Approximation (SA): [Robins Monro 1951]
  - Update $w^{(t)}$ based on z_t
    - E.g., based on $g^{(t)} = \nabla f(w, z_t)$
  - Simplest method: stochastic gradient descent
  - Similar to online approach in learning (more on this later)

Why is this important?

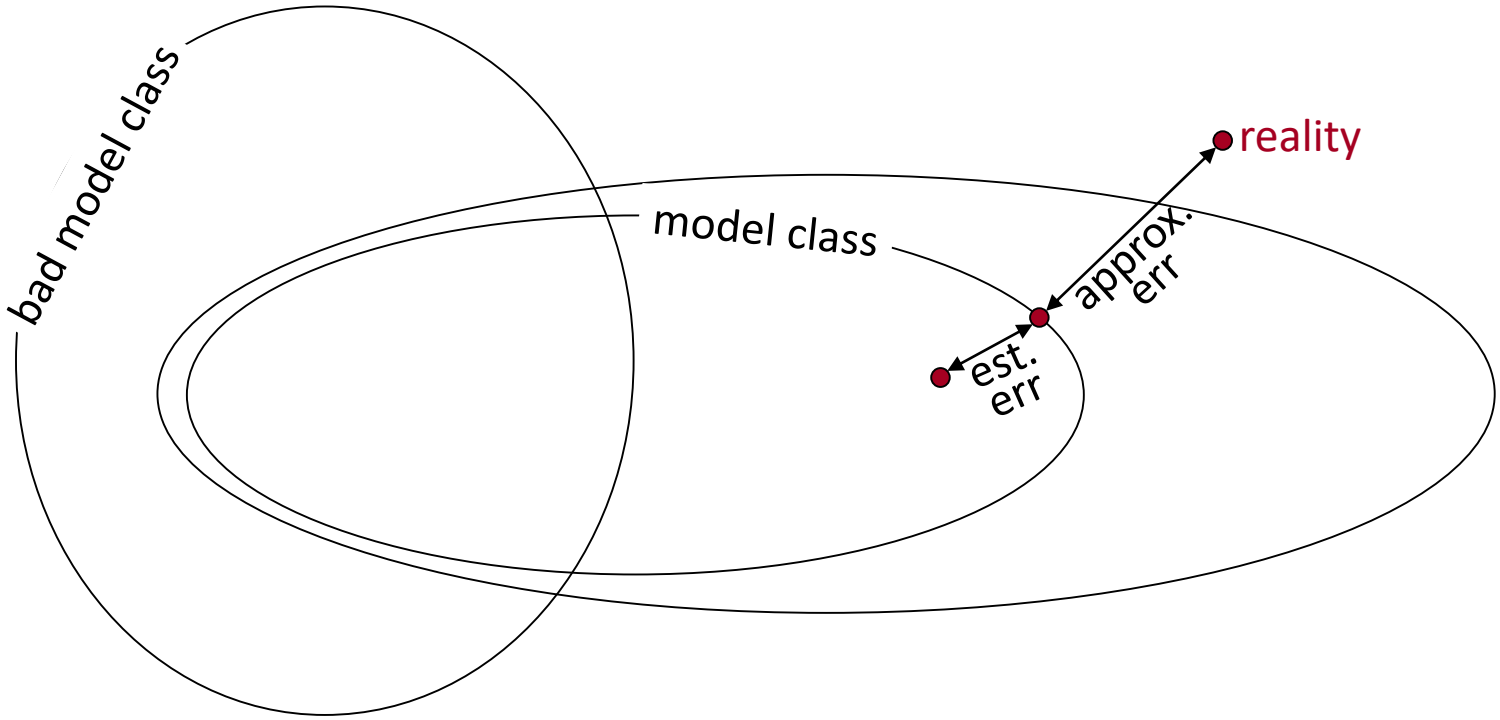
- Borrowing ideas and techniques, merging of communities
- Understanding optimality (in terms of sample complexity ***and*** runtime) of stochastic approximation algorithms
  - SGD optimal for SVM-type problems
  - Mirror Descent optimal* for any** convex problem
- Understanding how optimization algorithm can guarantee generalization directly, providing implicit inductive bias (regularization) without explicitly specifying hypothesis class / regularizer / model:

$$w^{(t+1)} \leftarrow w^{(t)} - \eta \nabla loss(w^{(t)}, (x_t, y_t))$$

$$\Rightarrow \quad \mathbb{E}\left[loss\left(w^{(t)}\right)\right] \leq \mathbb{E}[loss(w^*)] + \sqrt{\frac{\|w^*\|^2 \|x\|^2}{t}}$$

- Emphasize importance of computational aspects

Machine Learning



- We want model classes (hypothesis classes) that:
  - Are expressive enough to capture reality well
  - Have small enough capacity to allow generalization
- Use “expert knowledge” to design small model class that capture relevant reality well

Free Lunches

- No Free Lunch: For any learning rule, there exists a source (i.e. reality), for which the learning rule yields expected error ½
  - If we try learning all possible hypothesis (take model class to be all possible functions), estimation is impossible
  - Must introduce inductive bias / prior knowledge

- Universal Learning (Free Lunch):
  - "Universal" inductive bias, captures anything we might want to learn

$$\mathcal{H}_{\text{time}(T)} = \{\, all\ functions\ computable\ in\ time\ T \,\}$$

$$\mathcal{H}_{\text{desc-length}(L)} = \{functions\ computable\ by\ program\ of\ length \leq L \,\}$$

  - Sample complexity $\propto T$ or $\propto L$
  - If there isn't an efficient program (no way to perform task efficiently), no point in "learning" how to perform task

Minimum Description Length Learner

- Task: predict y from x
- Input: labeled training set $S = \{(x_1, y_1), (x_2, y_2), \ldots\}$

- `Return shortest program` $p: x \mapsto y$ `s.t.` $p(x_i) = y_i$ `for all` $(x_i, y_i) \in S$

Minimum Description Length (Noisy)

- Task: predict y from x
- Input: labeled training set $S = \{(x_1, y_1), (x_2, y_2), \dots\}$

- Split training set to S_{tr}, S_{val}
- For each length L:
  - $h_L =$ program $p: x \mapsto y$ of length $|p| \leq L$ with minimum error on S_{tr}
- Return h_L with minimum error on S_{val}

- "Universal Learner": learns any poly-time function to within any error with polynomial sample complexity
- Theoretically: only a constant more training examples compared to any programmable learning rule
- "In Practice": beats your (and anybody else's) learning method

No Free Lunch After All…

- Problem: "`find shortest program consistent with` S", or "`find program of length` $\leq L$ `minimizing training error`" is not computable
- Also "`find short program consistent with` S `and with short run time on` S" is NP-hard
- Not only NP-hard, its really really really hard.

- In fact, Universal Learning is hopeless:
  Unless crypto collapses, there is no polytime learning algorithm that can learn "functions computable in time T"
  - i.e. even if we know ∃ time-T function with zero error, we can't even ensure error < ¼ in poly-time

- Machine Learning Challenge
  - Expressive power: capture reality well
  - Low capacity: generalize well, low sample complexity
  - **Computationally efficient**

Hypothesis Class of Feed Forward Neural Networks

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

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

- Expressive Power / Approximation:
  - What functions can we represent/approximate?
  - Does $\mathcal{H}$ "capture reality" well?
- Estimation:
  - What is the capacity / VC-dimension of $\mathcal{H}$?
  - How well do we generalize to new data if we choose weights that minimize error on training data
  - How many samples do we need in order to generalize?
- Computation

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
  - $VCdim\left(\mathcal{H}_{G(V,E),\mathrm{RELU}}\right) = \widetilde{\Theta}(|E| \cdot depth)$
  - With finite precision (or a bit of regularization):

$$VCdim\left(\mathcal{H}_{G(V,E),\sigma}\right) = O(|E|)$$

- Bottom line: **$|E|$ (number of weights) controls sample complexity**

What can Feed-Forward Networks Represent?

- Any function over $\mathcal{X} = \{\pm 1\}^d$
  - With a single hidden layer, using DNF (hidden layer does AND, output does OR)
  - $|V| = 2^d, |E| = d2^d$
  - Like representing the truth table directly…

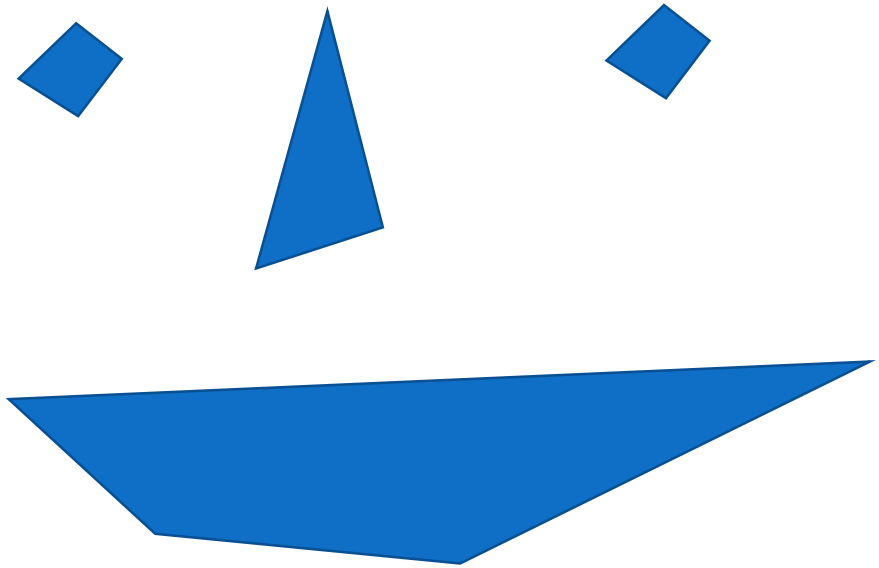
- Universal Representation Theorem: Any continuous functions $f: [0,1]^d \to \mathbb{R}$ can be approximated to within any ϵ by a feed-forward network with sigmoidal (or almost any other) activation and a single hidden layer.
  - Size of layer exponential in d

- Compare: With a large enough #params (large enough #features, small enough margin) even a *linear* model can approximate any continuous function arbitrary well (e.g. using Gaussian kernel)

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

- Everything we want:

$$\{ f \mid f \text{ computable in time } T \} \subseteq \mathcal{H}_{G(V,E),\sigma}$$

sign, sigmoidal or ReLU

with $|E| = \tilde{O}(T)$

⇒ Universal Learning (learn anything computable in time T) with $poly(T)$ samples

- Compare: to get "universal approximation" with linear models / kernels, margin must shrink (and #features must grow) exponentially

Optimization

$$ERM(S) = \arg\min_{w} L_S(f_w)$$

- Highly non-convex problem, even if $loss$ and activation σ are convex
- NP-Hard even with single hidden layer and three hidden units

- Not surprising: otherwise, can learn hypothesis class of all poly-time functions
- Conclusion: Under crypto assumptions, no algorithm for learning $\mathcal{H}_{G(V,E),\sigma}$ in time $poly(|E|)$

- In fact, even two-layer networks are hard:
  For $x \in \mathbb{R}^d$, and binary labels generated by two-layer network with $\log(d)$ hidden units, no poly-time learning algorithm ensuring error < ¼
  - Even in noiseless case (labels *exactly* follow small two-layer network)
  - Even if algorithm allowed to use much larger network (or any type of predictor)

[Kearns Valiant '94, $\log(d)$-depth; **Klivans Sherstov '06**, two-layer $O(d)$ units;
Daniely Linial Shalev-Shwartz '14, $\log(d)$ units**]**

Choose your universal learner:

Short Programs

- Universal
- Captures anything we want with reasonable sample complexity
- NP-hard to learn
- Hard to optimize in practice
  - No practical local search
  - Highly non-continuous, disconnected discrete space
  - Not much success

Deep Networks

- Universal
- Captures anything we want with reasonable sample complexity
- NP-hard to learn
- Often easy to optimize
  - Continuous
  - Amenable to local search, stochastic local search
  - **Lots of empirical success**

Theory of Neural Network Learning: Interim Summary

- Expressive Power
  - Universal, all poly-time functions
- Capacity Control (Sample Complexity)
  - $\propto$ number of weights
- Optimization

  ?????

Not: "what about reality is captured by my NN architecture"

Rather: "what about reality makes it easy to optimize my NN"

"its easy to optimize my NN ***on real data,***
because ***real data has such and such properties***"

You want convexity?

- Consider learning with a hypothesis class $\mathcal{H} = \{ h: \mathcal{X} \to \mathbb{R} \}$
$$\hat{L}(h) = \Sigma_t loss(h(x_t); y_t)$$

- With any meaningful loss, $\hat{L}(h_w)$ can be convex in a parameterization w, **only if $\boldsymbol{h_w(x)}$ is affine in $\boldsymbol{w}$**, i.e.
$$h_w(x) = \langle w, \phi(x) \rangle + \phi_0(x)$$

- Rich variety of learning problems obtained with different (sometimes implicit) choices of linear hypothesis classes, feature mappings Á, and loss functions.

Shallow Learning

(For 0/1 error, which is what we really care about, even linear learning is non-convex, NP-hard to optimize, and crypto-hard to learn)


Matrix Factorization
(two layer, linear transfer)
deep
convex:
shallow
linear

Feed Forward Neural Networks

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

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

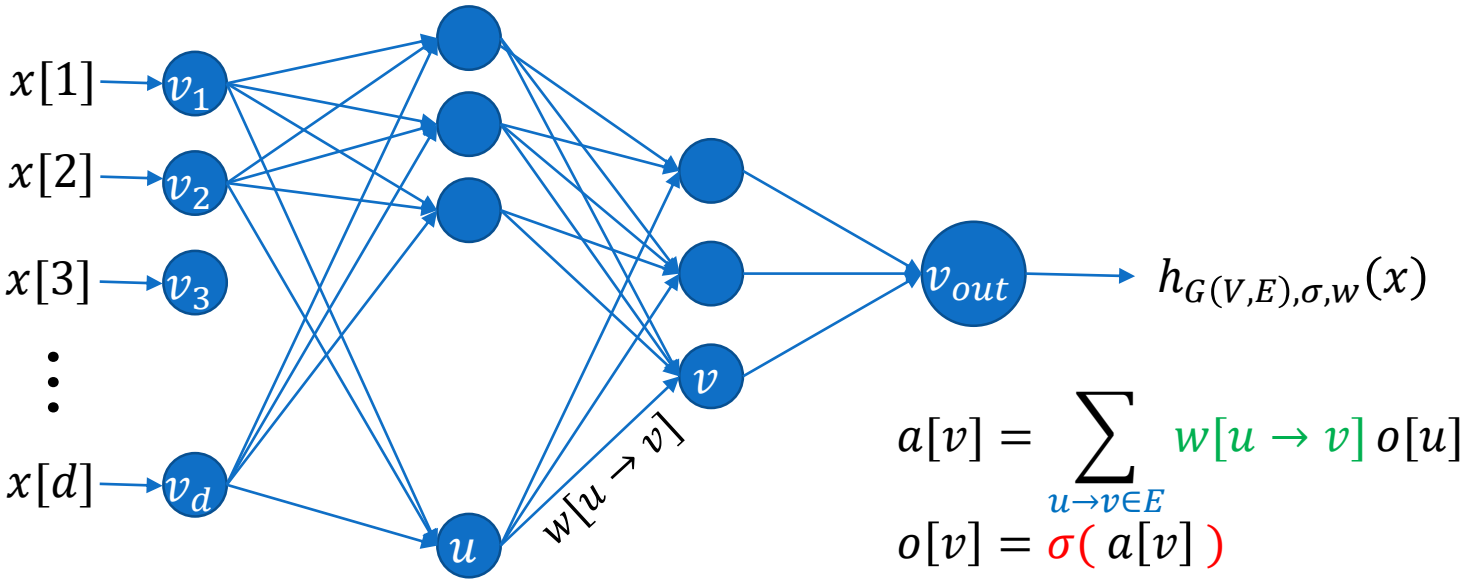
- Capacity / Generalization ability / Sample Complexity
  - $\widetilde{O}(|E|)$ (number of edges, i.e. number of weights)
    (with threshold σ, or with RELU and finite precision; RELU with inf precision: $\widetilde{\Theta}(|E| \cdot \text{depth})$)

- Expressive Power / Approximation
  - Lots of interesting things naturally with small networks
  - **Any T computable function with network of size $\widetilde{O}(T)$**

- 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]

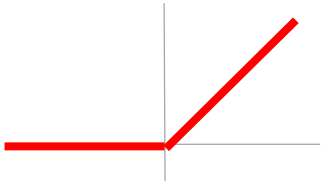
  - Magic property of reality that makes local search "work"

Feed-Forward Neural Networks (The Multilayer Perceptron)



Architecture:

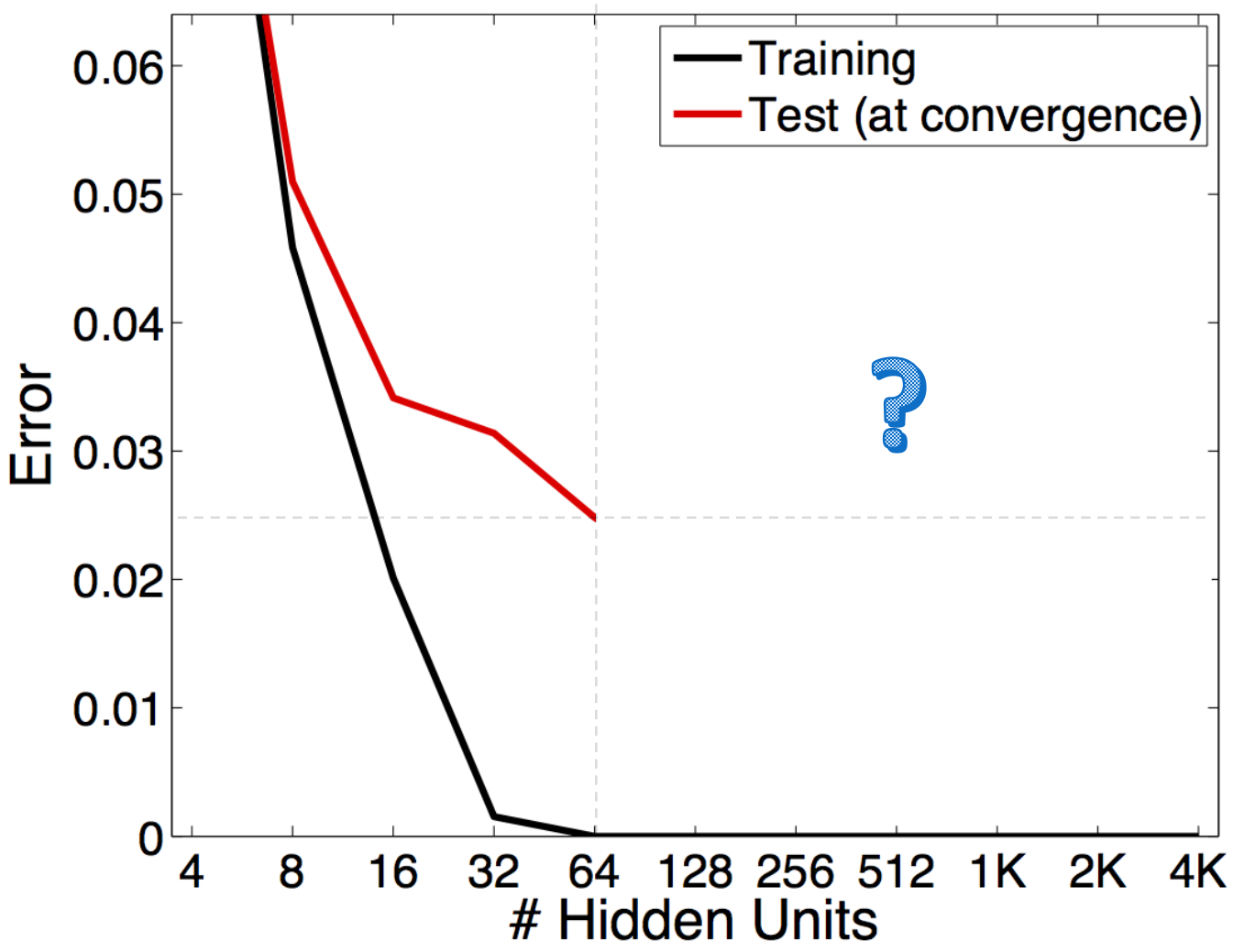
- Directed Acyclic Graph G(V,E). Units (neurons) indexed by vertices in V.
  - "Input Units" $v_1 \dots v_d \in V$, with no incoming edges and $o[v_i] = x[i]$
  - "Output Unit" $v_{out} \in V$, $h_w(x) = o[v_{out}]$
- "Activation Function" σ: $\mathbb{R} \to \mathbb{R}$. E.g. $\sigma_{RELU}(z) = [z]_+$



Parameters:

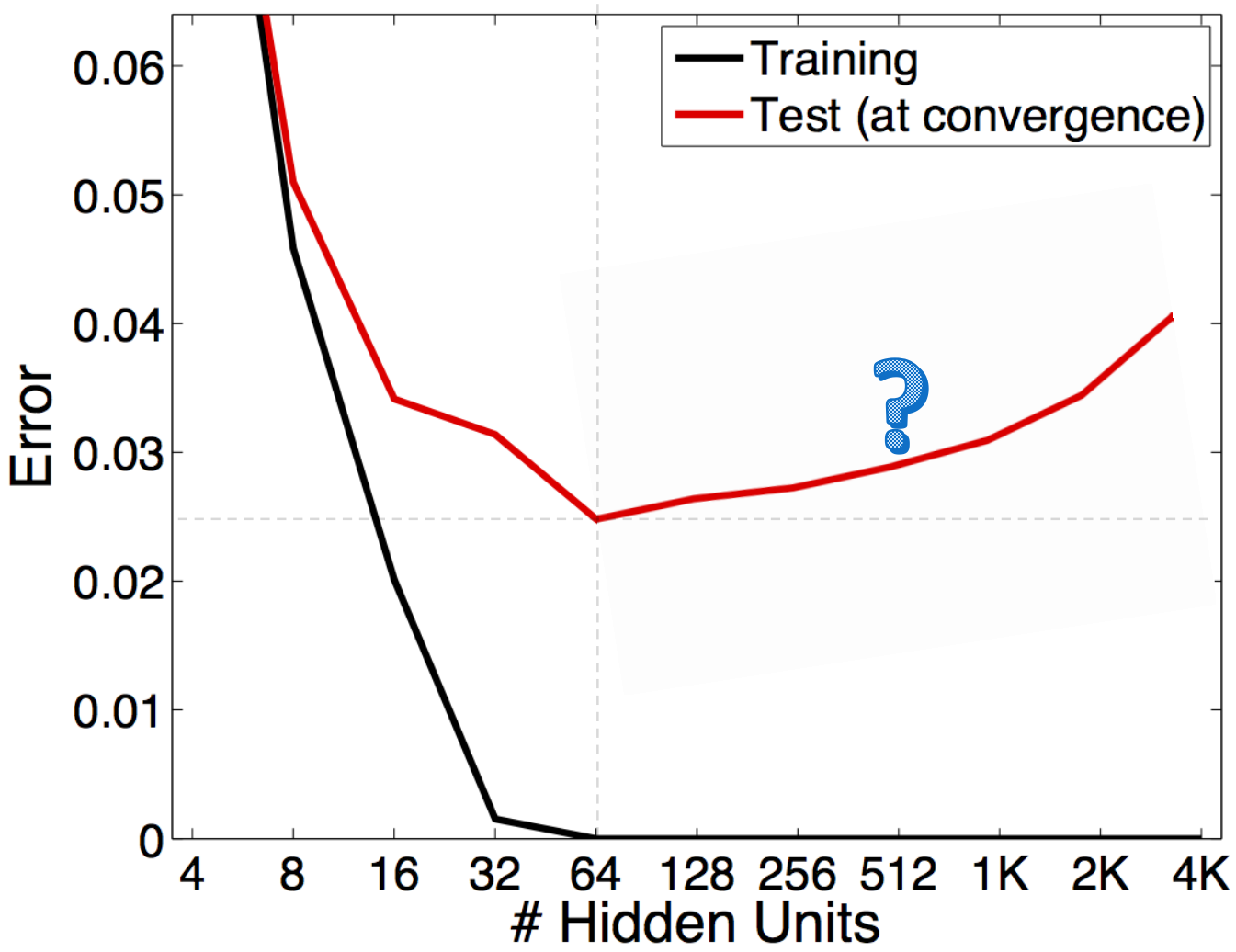
- Weight $w[u \to v]$ for each edge $u \to v \in E$

Increasing the Network Size (Number of Hidden Units)



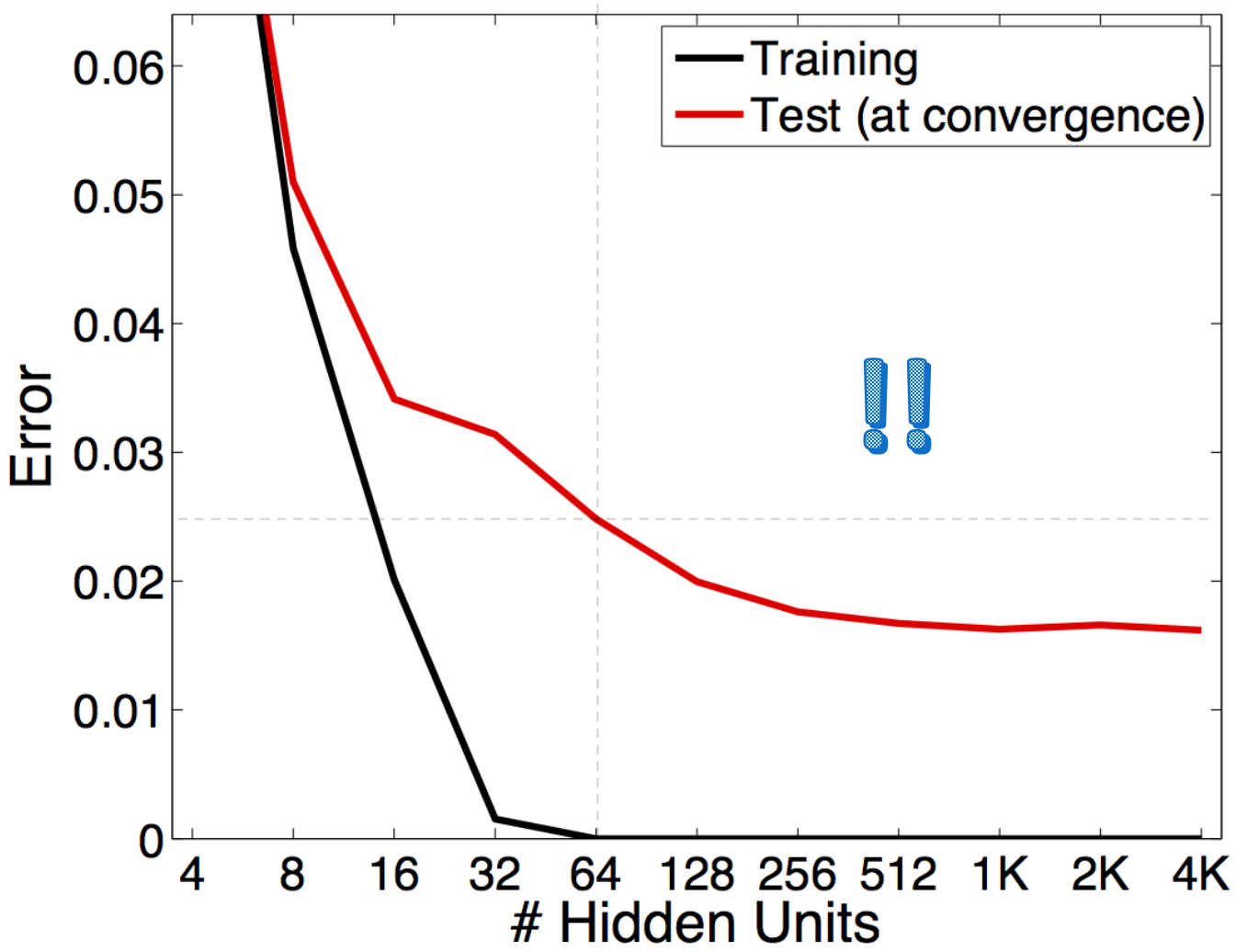
[Neyshabur Tomioka S ICLR'15]

Increasing the Network Size (Number of Hidden Units)



[Neyshabur Tomioka S ICLR'15]

Increasing the Network Size (Number of Hidden Units)



[Neyshabur Tomioka S ICLR'15]

Feed Forward Neural Networks

- Capacity / Generalization ability

????

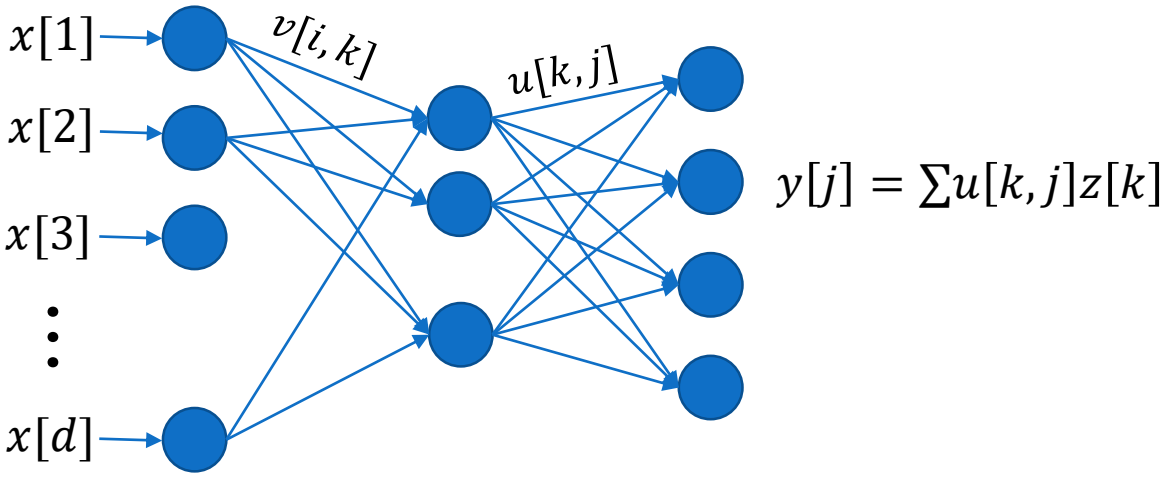
- Expressive Power / Approximation

?

- Computation / Optimization

??

Two Layer Networks with Linear Transfer
(aka Matrix Factorization)



$$y = U(Vx) = Wx$$

$$W = UV$$

r hidden units $\Leftrightarrow rank(W) \leq r$

Norm-Bounded Factorization

- Instead of $rank(W)$ [number of hidden units]
  consider $\|W\|_{tr} = \min_{W=UV} \|U\|_{Fro}\|V\|_{Fro}$ [magnitude of weights]

(or other factorization norms such as weighted trace norm, max-norm aka $\gamma_2: 1 \to \infty$ norm, etc)

- Convex, easier to optimize, no spurious local minima in high enough dimension even when optimizing over U, V

- But also: better inductive bias
  - Richer and more expressive model, allowing unbounded number of factors.
  - Infinite factor model with bound sum of "importance" of factors, not their number

Increasing the Rank

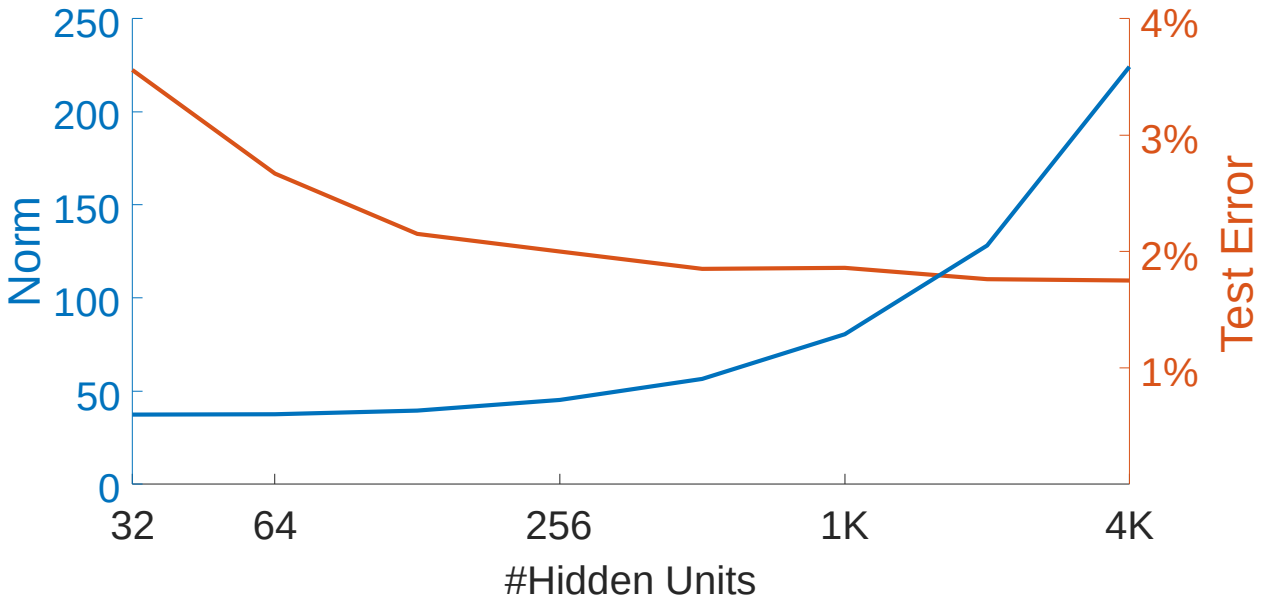
$$W_k = \underset{rank(W)\leq k}{\arg\min} \|W\|_{tr} \text{ s.t. } L(W) = 0$$

- $rank(W_k)$ increases with k
  ➔ more complex in terms of rank
- $\|W_k\|_{tr}$ decreases with k
  ➔ simpler in terms of norm

- If norm is better "inductive bias", W_k generalizes better as $k \nearrow$

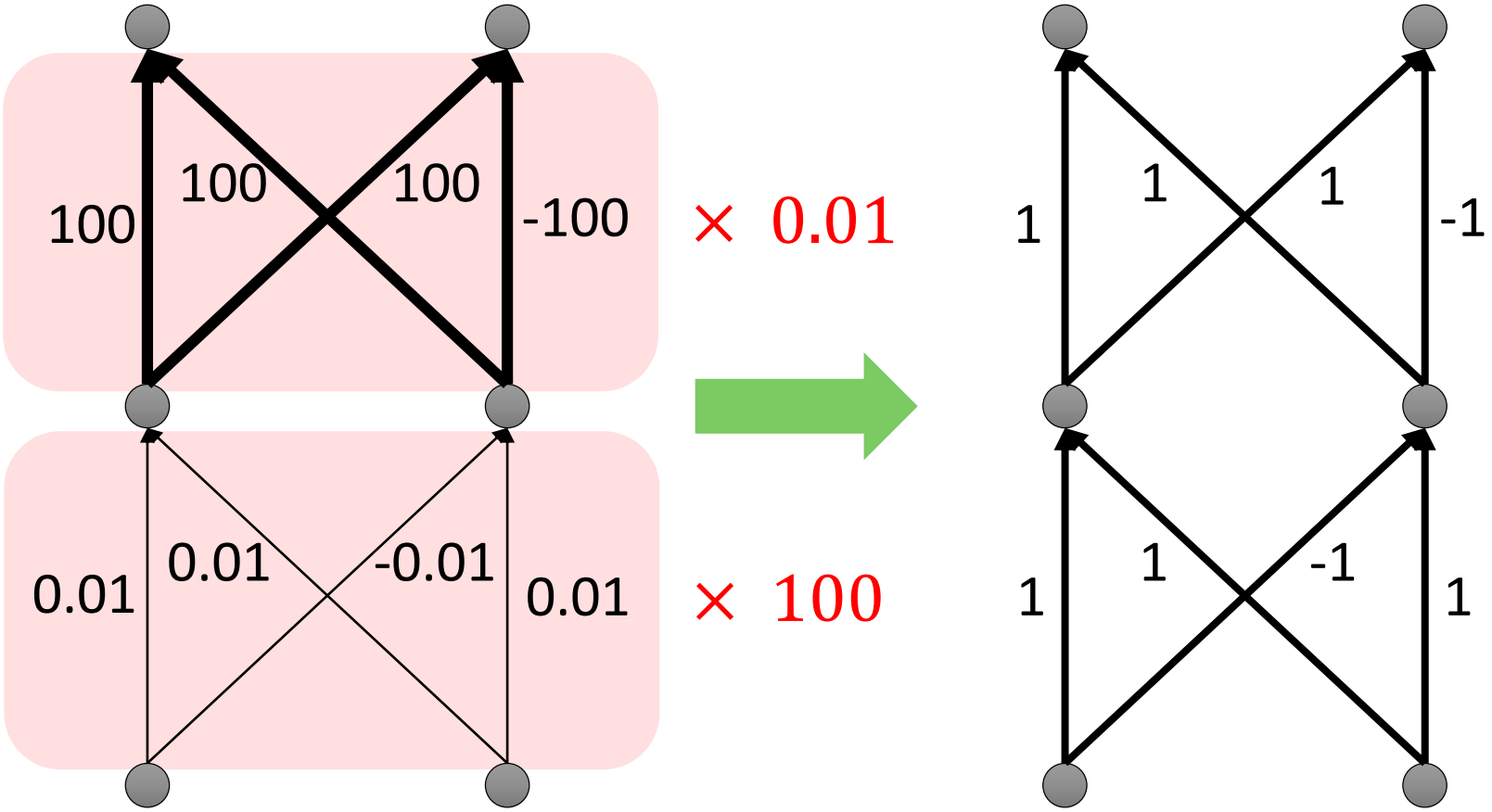
- In practice, for many tasks (including NetFlix)
  $$W_k = \arg\min_W L(W) + \lambda\|W\|_{tr} \quad s.t.\ rank(W) = k$$
  Test error of W_k monotonically decreases as $k \nearrow$

Is improved Generalization explained by Decrease in Norm?



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

$\|W\|_2$ doesn't capture complexity



$\|W\|_2 \approx 200$

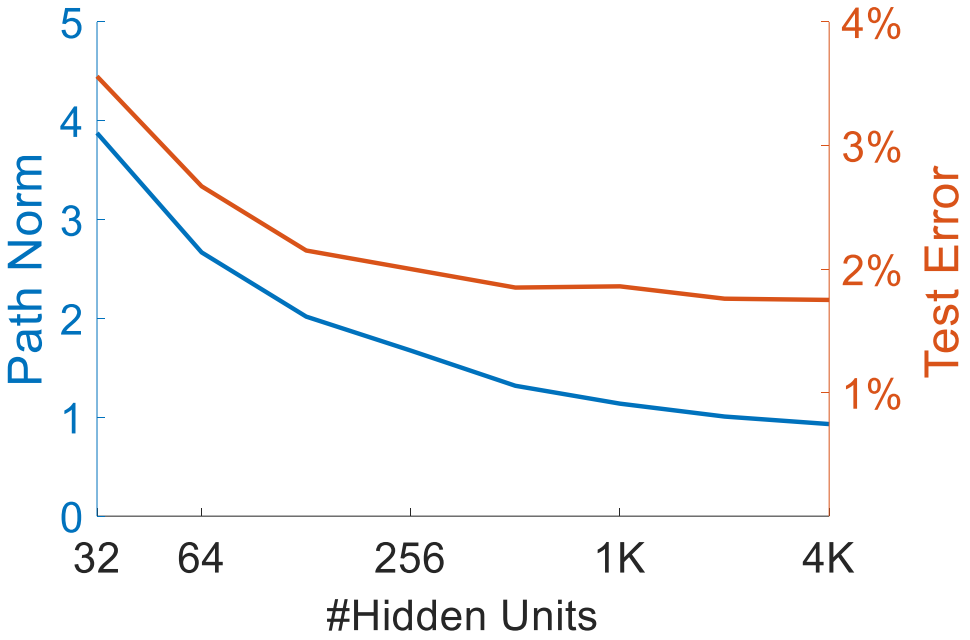
$\|W\|_2 = 2\sqrt{2}$

The Path-Norm

$$\phi(W) = \sqrt{\sum_{\text{path}} \prod_{e \in \text{path}} w(e)^2}$$

With ReLU activations:

- Invariant to "weight balancing" ➔ depends more directly on f_W
- Bounding the path-norm provides capacity control and ensures generalization, independent of #units (and even if #units unbounded)

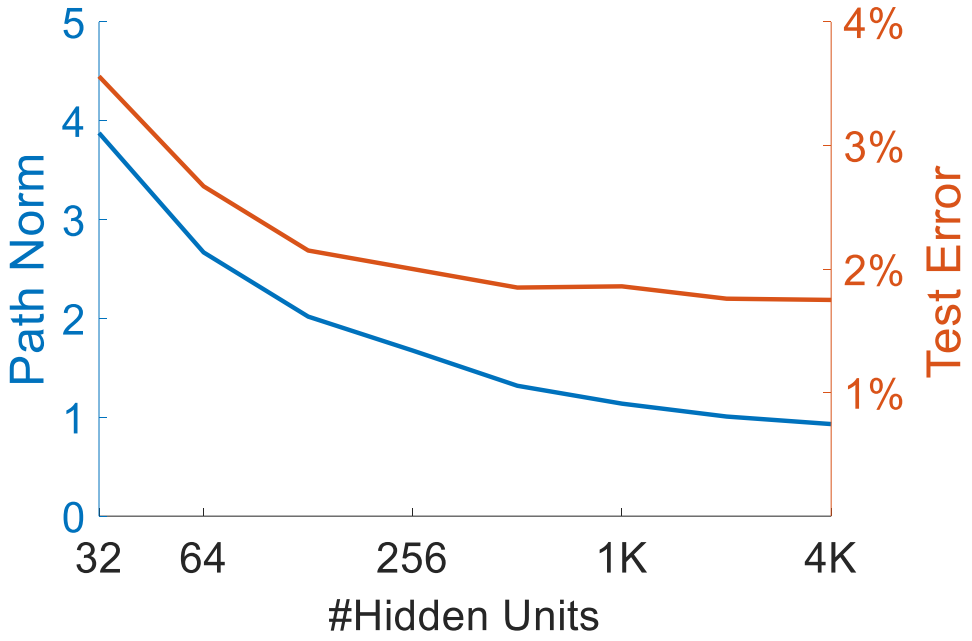


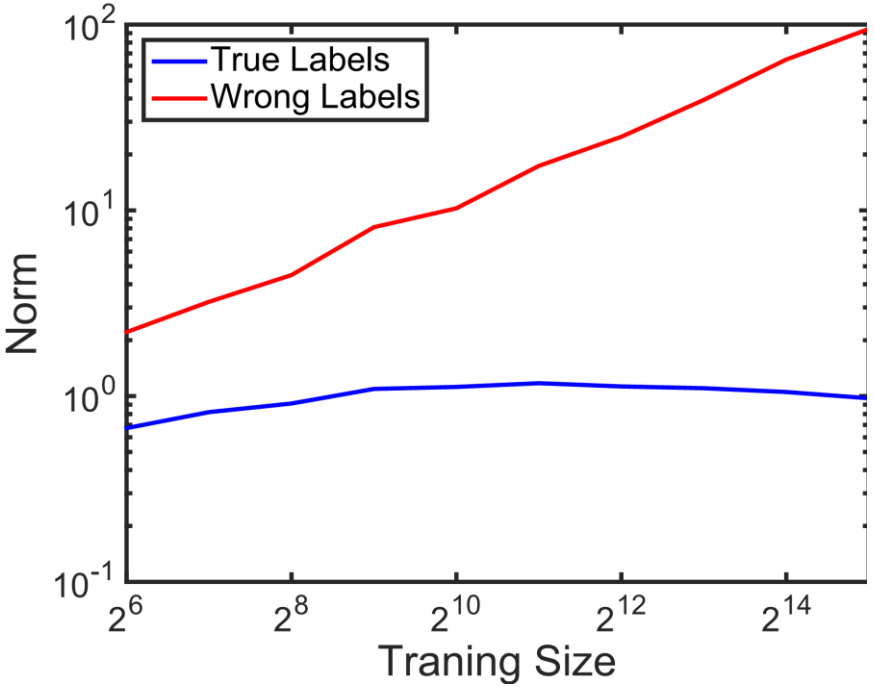
The Path-Norm

$$\phi(W) = \sqrt{\sum_{\text{path}} \prod_{e \in \text{path}} w(e)^2}$$

With ReLU activations:

- Invariant to "weight balancing" ➔ depends more directly on f_W
- Bounding the path-norm provides capacity control and ensures generalization, independent of #units (and even if #units unbounded)



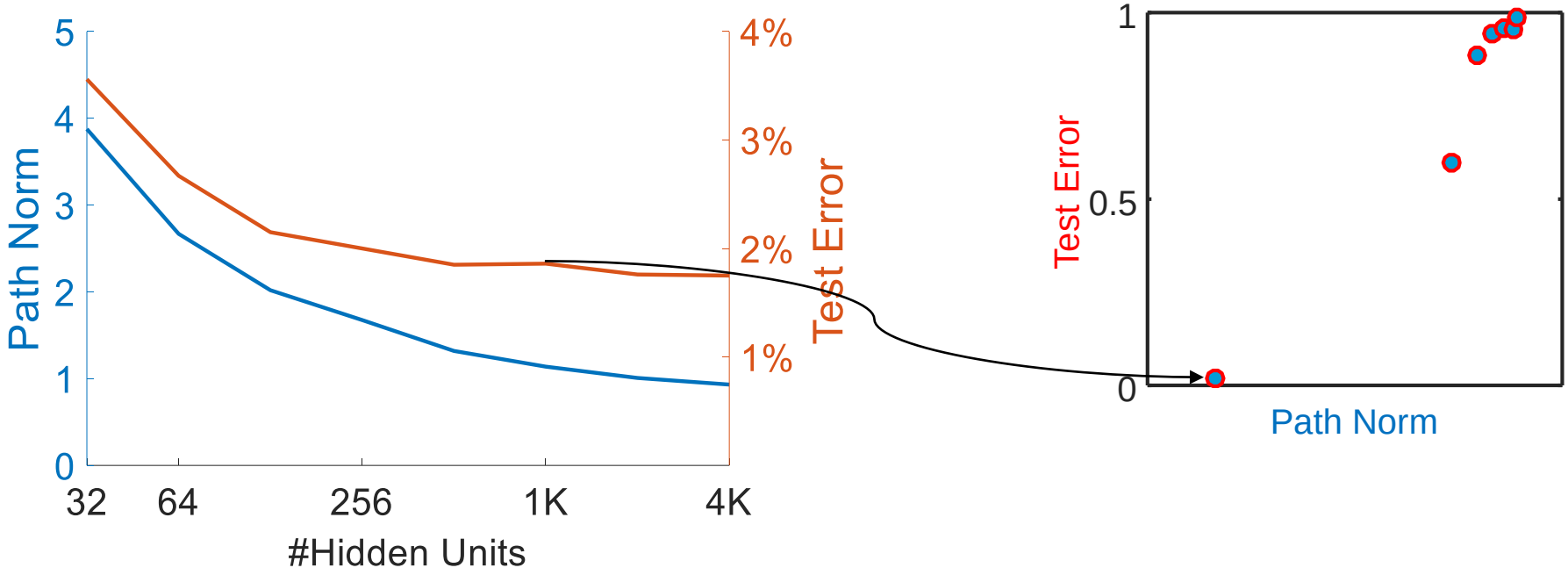


The Path-Norm

$$\phi(W) = \sqrt{\sum_{\text{path}} \prod_{e \in \text{path}} w(e)^2}$$

With ReLU activations:

- Invariant to "weight balancing" ➔ depends more directly on f_W
- Bounding the path-norm provides capacity control and ensures generalization, independent of #units (and even if #units unbounded)



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, one-pass Mirror Descent provides generalization w.r.t. any* inductive bias
  - Inductive Bias ⇔ choice of potential for Mirror Descent
- We are getting implicit regularization, **without early stopping**

- In underdetermined problem (lots of global optima), optimization is biasing us toward specific global optimum.

- What's the bias introduced by the optimization?
- Can we get better bias by changing optimization?

Optimization is Tied to Choice of Geometry

Steepest descent w.r.t. a geometry:

$$w^{(t+1)} = \arg\min_{w} \eta \left\langle \nabla \mathrm{L}\left(\mathrm{w}^{(t)}\right), w \right\rangle + \delta\left(w^{(t+1)}, w\right)$$

✓improve the objective as much as possible

✓only a small change in the model.

Examples:

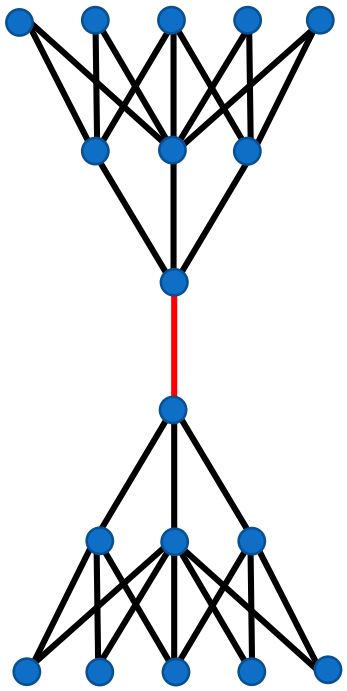
- Gradient Descent: Steepest descent w.r.t ℓ_2
- Coordinate Descent: Steepest descent w.r.t. ℓ_1

What's the geometry appropriate for deep networks?

Better Geometry

- Can we devise a better geometry?
- More directly dependent on the **functions computed by the network**, not the vector of weights
- **Invariant to rescaling / reparametrization**
- Captures a **natural notion of complexity** for deep networks

Path-SGD: (Approximate) Steepest Descent on $\phi(W)$



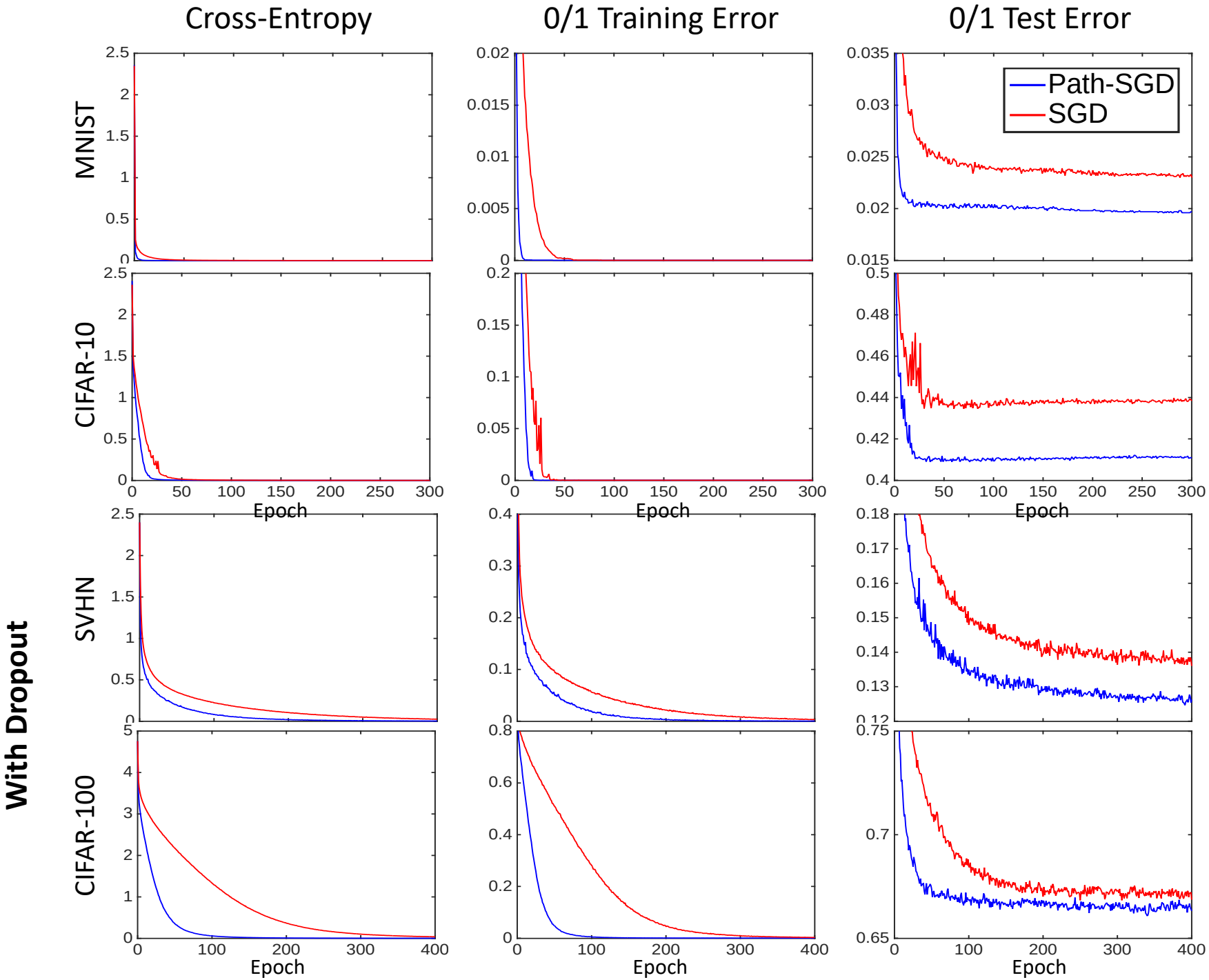
$$\phi(W) = \sqrt{\sum_{\text{path}} \prod_{e \in \text{path}} w(e)^2}$$

$$w_e^{(t+1)} = w_e^{(t)} - \frac{\eta}{\kappa_e(w^{(t)})} \frac{\partial L}{\partial w(e)}\left(w^{(t)}\right)$$

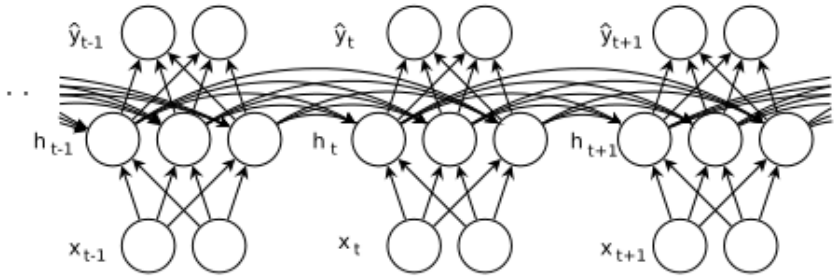
$$\kappa_e(w) = \sum_{\substack{\text{path} \\ \text{via } e}} \prod_{e' \in \text{path}} w(e')^2$$

✓ **As fast as a forward-backward step on a single data point ☺**

[Neyshabur Salakhudtinov S NIPS'15]

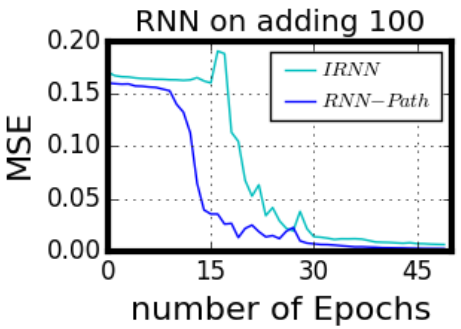

Cross-Entropy
0/1 Training Error
0/1 Test Error
With Dropout
MNIST
CIFAR-10
SVHN
CIFAR-100
Path-SGD
SGD
Epoch

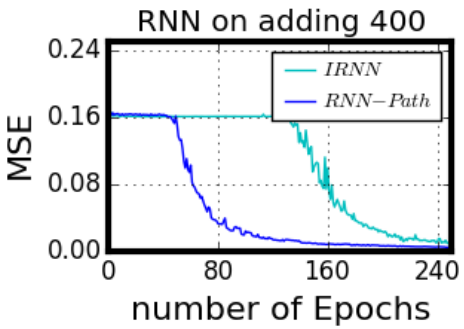
Recurrent Neural Networks

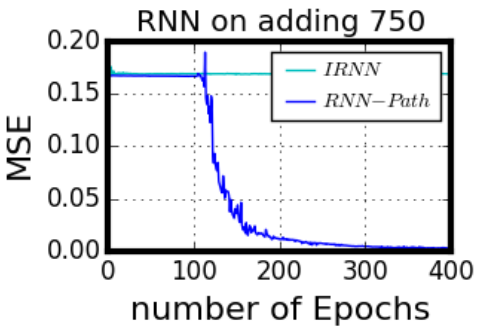


- Sequence to sequence learning architecture
  - Translations, parsing, speech
- Notoriously hard to train: large effective depth and repeated weights
  - With saturating activations (sigmoid, tanh, ramp) : gradient decay
  - With unbounded activation (ReLU): weight explosion
- Most empirical success is with LSTMs, GRUs
  - Different, more complicated architecture
  - Saturating activations, but “short circuits” gradient paths
  - Is this more complicated modeling really necessary?
- Instead: Path SGD to plain vanilla ReLU RNNs

Path-SGD on RNNs







	Penn Tree-Bank	Text8
Plain ReLU SGD	1.55	1.65
Plain ReLU Path-SGD	1.47	1.58
LSTM	1.41	1.52
%gap to LSTM bridged	57%	53%

[Neyshabur Wu Salakhudtinov S 2016]

Building a Circumstantial Case

- Generalization ability
  - Optimization biases us toward "low norm" **???**
  - "low norm" ensures generalization [Neyshabur S Tomioka COLT15]
- Expressive Power / Approximation
  - "being representable by a low-norm NN"
- Computation / Optimization
  - Magic property of reality that makes local search "work" **???**
    "being representable by a low-norm NN" ???

The Mysteries of Local Search (Gradient Descent and Relatives)

- How come local search succeeds in finding a global optimum of a non-convex function?

- How does gradient descent bias us towards low norm solutions? What norm?

Non-Convex Optimization: Low Rank Matrix Factorization

$$\min_{rank(X)\leq r} \|\mathcal{A}(X) - y\|_2^2 = \sum_{i=1}^{m} (\langle A_i, X\rangle - y_i)^2$$

$$\min_{U\in\mathbb{R}^{n\times r}} f(U) = \|\mathcal{A}(UU^T) - y\|_2^2$$

- Non-convex
  - Rank is non-convex constraint
  - $f(U)$ is non-convex objective

- Focus on noiseless measurements:
  - $y_i = \langle A_i, X^*\rangle,\ rank(X^*) \leq k$
  - Goal: recover X^*

Recovery with Convex Relaxation

$$X^*_{SDP} = \arg\min_X \|\mathcal{A}(x) - y\|_2^2 + \lambda \|X\|_{tr}$$

Definition: A satisfies (δ_r, r) isometry, if for any rank-r X:

$$(1-\delta_r)\|X\|_F^2 \leq \frac{1}{m}\|\mathcal{A}(X)\|_2^2 \leq (1+\delta_r)\|X\|_F^2$$

- Satisfied for iid Gaussian A_i with $m = \Omega\left(\frac{nr}{\delta_r^2}\right)$

- If $\delta_{2r} < 1$, then X^* is unique (and so recoverable)
- If $\delta_{4r} < 0.414$, then $X^*_{SDP} = X^*$

[Recht Fazel Parrilo 2007, Candes Recht 2008]

But: Computationally heavy; not what NN do

Global Initialization + Local Search

- Step 1: Initialize X_{init} based on SVD of measurements
- Step 2: Local search starting from $X_{\text{init}} = U_{\text{init}} U_{\text{init}}^T$
  - Alternating minimization **[Jain Netrapalli Sunghavi 2012]**
  - Gradient Descent on U_{init} **[Zang Lafferty 2015, Tu Boczar Simchowitz Soltanolkotabi Recht 2015, Chen Wainwright 2015, Bhojanapalli Kyrillidis Sanghavi 2015]**

- If $\delta_{2r} \leq O\left(\frac{1}{r}\right)$, local search after SVD converges (quickly) to X^*

But:

- SVD does heavy lifting: problem almost convex after SVD
- Not what NN do

Our Result: Local Search Sufficient

[Bhojanapalli Neyshabur S NIPS16]

$$\min_{U \in \mathbb{R}^{n \times r}} f(U) = \|\mathcal{A}(UU^T) - y\|_2^2$$

Theorem: If $\mathcal{A}$ satisfies isometry with $\delta_{2r} \leq 0.2$, then:

- All local minima are global (no spurious local minima)
- All saddle points are strict: $\lambda_{min}(\nabla^2) \leq -\frac{4}{5}\sigma_r(U^*)^2$

Corollary: Starting from random initialization, noisy gradient descent [Ge Huang Jin Yuan 2015] converges to global optimum in $poly\left(\kappa(U^*), \frac{1}{\epsilon}, n\right)$ iterations

(Extensions also to noisy and approximate low-rank)

This is what NN do!

Convex Relaxation Work ➔ Local Search Works

- (this work) Low-rank recovery with linear measurements
  - $\delta_{2r} < 0.2$, i.e. $O(nr)$ iid Gaussian measurements suffice
- [Ge Lee Ma 2016, in parallel] Low-rank Matrix Completion
  - Special type of linear measurements, not isometric
  - Need additional regularization
  - Stricter dependence on δ ➔ poly(r) same complexity
- Inspired by prior work on rank-1 problems:
  - [Bandeira Boumal Voroninski 2015] community detection
  - [Sun Qu Wright 2015] phase retrieval
- PCA: Non-convex, but no spurious local min [e.g. S Jaakkola 2003]
- "multilayer" PCA: $\min_{W_1, W_2, \ldots, W_d} \|\prod W_i X - Y\|$ [Kawaguchi 2015]
- Vector sparse problems: [Tropp 2004]

The Second Mystery of Local Search

- How can we get implicit regularization **without early stopping**?

- How does gradient descent bias us towards low norm solutions? What norm?

Warm-up: Least Squares

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

$$\min_{w \in \mathbb{R}^d} \| Ax - y \|^2$$

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

- Claim: Gradient Descent (or SGD, or conjugate gradient descent, or BFGS) converges to the least norm solution

$$\min_{Ax=y} \|x\|_2$$

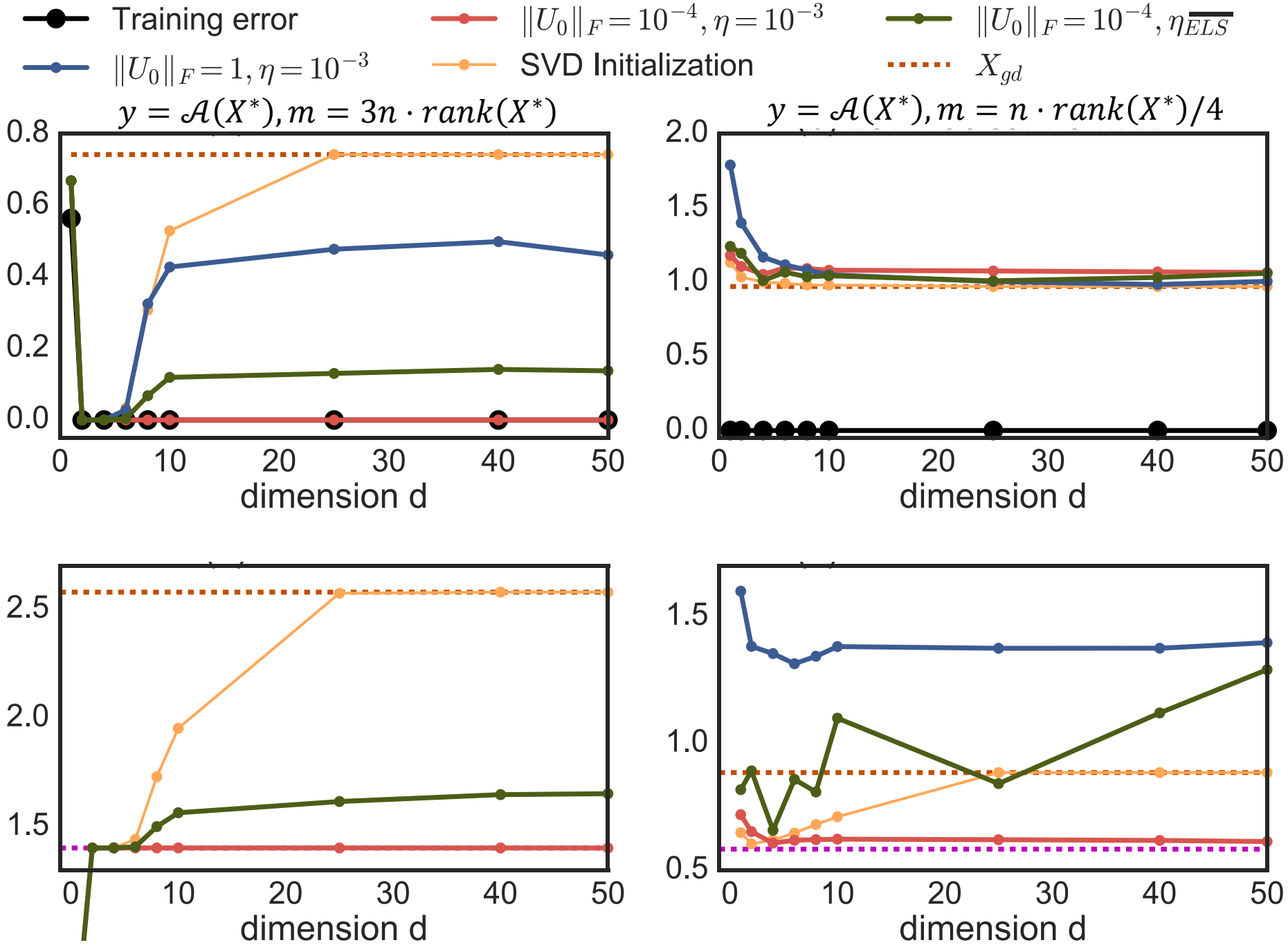
➢ Proof: iterates always spanned by rows of A

Gradient Descent on Factorization

$$\min_{U \in \mathbb{R}^{n \times n}} f(U) = \|\mathcal{A}(UU^T) - y\|_2^2$$

This time:

- Any linear operator $\mathcal{A}$ (incoherent, non-incoherent, matrix completion, etc)
- Allow high-dimensional U, i.e. unconstrained high rank $X = UU^T$
- Underdetermined problem, $m < \frac{n(n+1)}{2}$
- GD will typically converge to global min $A(UU^T) = y$
- **But there are many global min. Which one??**


Training error
$\|U_0\|_F = 10^{-4}, \eta = 10^{-3}$
$\|U_0\|_F = 10^{-4}, \eta_{\overline{ELS}}$
$\|U_0\|_F = 1, \eta = 10^{-3}$
SVD Initialization
X_{gd}
$y = \mathcal{A}(X^*), m = 3n \cdot rank(X^*)$
$y = \mathcal{A}(X^*), m = n \cdot rank(X^*)/4$
dimension d
dimension d
dimension d
dimension d

Conjecture (informal): With small enough steps and starting close enough to zero, gradient descent on U converges to minimum nuclear norm solution:

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

Gradient descent with infinitesimal stepsize:

$$\dot{U}_t = \frac{dU_t}{dt} = -\nabla_U f(U_t) = -\mathcal{A}^*(\mathcal{A}(U_t U_t^\top) - y)U_t = -\mathcal{A}^*(r_t)U_t$$

$r_t = \mathcal{A}(\ X_t\ ) - y$

Induces dynamics on $X_t = U_t U_t^\top$:

$$\dot{X}_t = \dot{U}_t U_t^\top + U_t^\top \dot{U}_t = -\mathcal{A}^*(r_t)\ X_t\ -\ X_t\ \mathcal{A}^*(r_t)$$

Dynamics independent on choice of factorization (not so with finite stepsize!)

For initial X_0 define $X_\infty(X_0) = \lim_{t\to\infty} X_t$

Conjecture: For any full rank $\tilde{X}$, if $X_{limt} = \lim_{\alpha\to 0} X_\infty(\alpha\tilde{X})$ converges to a global min with $\mathcal{A}(X_{limit}) = y$ then:

$$X_{limit} \in \arg\min_{X \succcurlyeq 0} \|X\|_* \ s.t.\ \mathcal{A}(X) = y$$

Conjecture: For any full rank $\tilde{X}$, if $X_{limt} = \lim_{\alpha \to 0} X_\infty(\alpha \tilde{X})$ converges to a global min with $\mathcal{A}(X_{limit}) = y$ then:

$$X_{limit} \in \arg\min_{X \succcurlyeq 0} \|X\|_* \ s.t. \ \mathcal{A}(X) = y$$

- Can prove conjecture when A_is commute

  ➔ **Corollary**: Consider non-negative vector least square problem

  $$\min_{x \in \mathbb{R}^n_+} \|Ax - y\|_2^2$$

  optimizing by gradient descent on $u \in \mathbb{R}^n$ with $x_i = u_i^2$. If we start at $u_0 = \alpha \mathbf{1}$, as $\alpha \to 0$, grad flow converges to min ℓ_1 norm solution: $\arg\min_x \|x\|_1 \ s.t. \ Ax = y$

- General A_i: empirical validation + hand waving

Warm Up: Gradient Descent on X

$$\min_{X \in \mathbb{R}^{n \times n}} F(X) = \|\mathcal{A}(X) - y\|_2^2$$

Claim: Starting at $X_0 = 0$, gradient descent on X converges to

$$\min_X \|X\|_F \;\; s.t. \;\; \mathcal{A}(X) = y \qquad (*)$$

Proof:

- X_t stays on $\mathcal{M} = span(A_i) = \{X = \mathcal{A}^*(s) = \sum_i s_i A_i \,|\, s \in \mathbb{R}^m\}$

  Reason: gradients $\nabla_X F(X)$ are tangent to $\mathcal{M}$
- Consider KKT of (*):

$\mathcal{A}(X) = y$ — holds at global min

$\mathcal{A}^*(v) = X$ — satisfied for all $X \in \mathcal{M}$

- Conclusion: if GD converges to global min (and it will), we optimize (*)
- Since $\mathcal{M}$ is flat: holds also with finite step size, conjugate GD, momentum

GD on U, single observation (m=1)

$$\dot{X}_t = -r_t(AX_t + X_tA)$$

- Solution:

$$X_t = e^{s_tA}X_0e^{s_tA} \qquad s_t = -\int r_t dt$$

- Consider:

$$\min_{X \succcurlyeq 0} \|X\|_* \ s.t. \langle A, X\rangle = y \qquad (*)$$

- KKT:

$$X \succcurlyeq 0 \qquad AX = y \qquad X = \nu AX \qquad \nu A \preccurlyeq I$$

- As $X_0 \to 0$, $s_\infty \to \infty$ and so only dominant eigenvectors of A survive
  - ➔ X_∞ spanned by eigen vectors of A with eigen value $\lambda_{max}(A)$
  - ➔ $X_\infty = \nu AX_\infty$ with $\nu = 1/\lambda_{max}(A)$
  - ➔ If also $AX = y$, we found an optimum to $(*)$

What we can prove: commutative A_i

$$\dot{X}_t = -\left(\mathcal{A}^*(r_t)X_t + X_t\mathcal{A}^*(r_t)\right)$$

- Solution:

$$X_t = e^{\mathcal{A}^*(s_t)}X_0 e^{\mathcal{A}^*(s_t)} \qquad s_t = -\int r_t dt \in \mathbb{R}^m$$

- Consider:

$$\min_{X \succcurlyeq 0} \|X\|_* \ s.t. \ \mathcal{A}(X) = y \qquad (*)$$

- KKT:

$$X \succcurlyeq 0 \quad \mathcal{A}(X) = y \qquad X = \mathcal{A}^*(v)X \qquad \mathcal{A}^*(v) \preccurlyeq I$$

- As $X_0 \to 0$ and $s \to \infty$ only dominant eigenvectors of $\mathcal{A}^*\left({}^{s}/_{\|s\|}\right)$ survive

  ➔ $X_\infty = \mathcal{A}^*(v)X_\infty$ satisfied with $v = s/\lambda_{max}\left(\mathcal{A}^*(s)\right)$

  ➔ If also $\mathcal{A}(X) = y$, we found an optimum to (*)

What we can prove: commutative A_i

$$\dot{X}_t = -\left(\mathcal{A}^*(r_t)X_t + X_t\mathcal{A}^*(r_t)\right)$$

Theorem: If A_i commute ($A_iA_j = A_jA_i$), then for any full rank $\tilde{X}$, if $X_{limt} = \lim_{\alpha\to 0} X_\infty(\alpha\tilde{X})$ is a global min with $\mathcal{A}(X_{limit}) = y$ then:

$$X_{limit} \in \arg\min_{X\succcurlyeq 0} \|X\|_* \ s.t.\ \mathcal{A}(X) = y$$

- Independent of "steering" r_t—just need to stay on:

$$\mathcal{M} = \left\{X = e^{\mathcal{A}^*(s)}X_0e^{\mathcal{A}^*(s)} \middle| s \in \mathbb{R}^m\right\}$$

- E.g., can minimize other loss, use weights, or sample A_i
- But finite steps, as well as (infinitesimal) momentum, will fall off $\mathcal{M}$!

Corollary: Consider non-negative vector least square problem

$$\min_{x\in\mathbb{R}^n_+} \|Ax - y\|_2^2$$

optimizing by gradient descent on $u \in \mathbb{R}^n$ with $x_i = u_i^2$. If we start at $u_0 = \alpha\mathbf{1}$, as $\alpha \to 0$, grad flow converges to min ℓ_1 norm solution: $\arg\min_x \|x\|_1 \ s.t.\ Ax = y$

The Non-Commutative Case

$$\dot{X}_t = -\left(\mathcal{A}^*(r_t)X_t + X_t\mathcal{A}^*(r_t)\right)$$

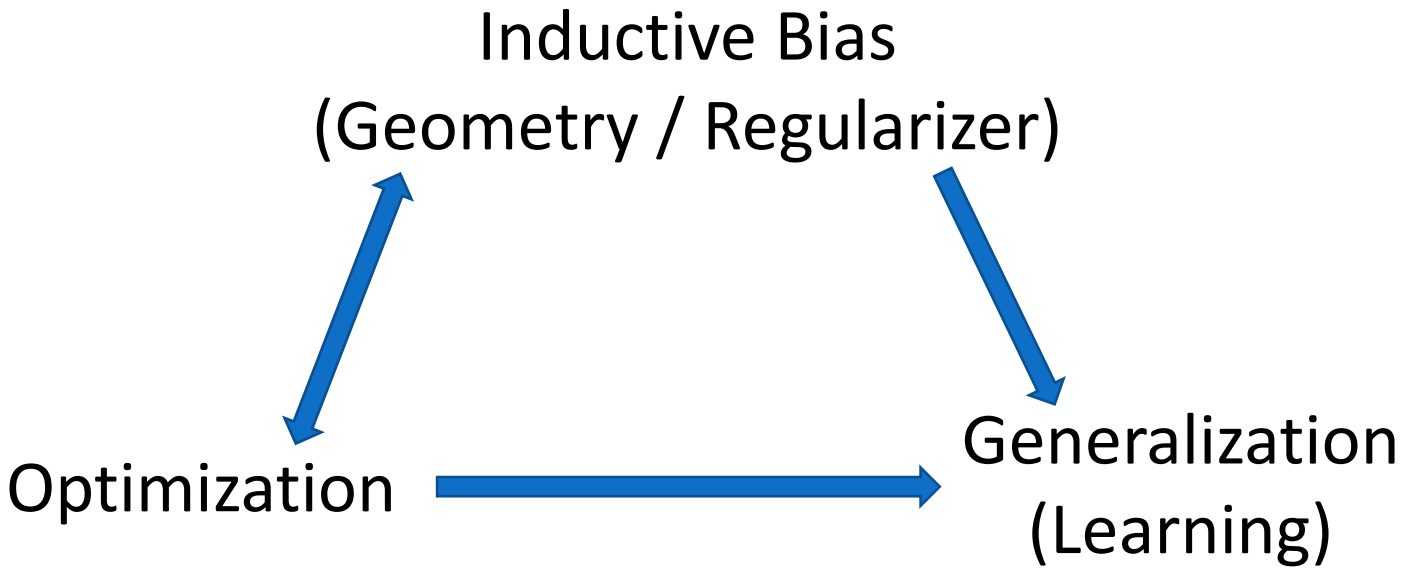
- Solution given by "time ordered exponential":

$$X_t = \left(\lim_{\epsilon\to 0}\prod_{\tau=t/\epsilon}^{0} e^{-\epsilon\mathcal{A}^*(r_\tau)}\right)X_0\left(\lim_{\epsilon\to 0}\prod_{\tau=0}^{t/\epsilon} e^{-\epsilon\mathcal{A}^*(r_\tau)}\right)$$

- With arbitrary (crazy) steering, can move in any direction and get to any psd matrix (even with $m = 2$ random measurement matrices)

- Empirically, with residual steering, or other "smooth", non-crazy steering, if we move far away form $X_0 \approx 0$, X_t does satisfy the dual condition

- Possible approach: if steering is "non-crazy" (total variations converge as integral diverges), non-commutative terms are lower order and directions not spanned by leading eigenvectors of $\mathcal{A}^*\left(\int r_t dt\right)$ vanish.


Inductive Bias
(Geometry / Regularizer)
Optimization
Generalization
(Learning)