Applying Statistical Learning Theory to Deep Learning:

What we understand, what we need understand, and what we need to re-think

Or: *what's on the x axis?*

Nati Srebro (TTIC)

Based on joint work with many co-authors, including
Behnam Neyshabur (TTIC→Google) , Blake Woodworth (TTIC→INRIA), Suriya Gunasekar (TTIC→MSR),
Jason Lee (Princeton), and Daniel Soudry (Technion)

And recent work on Benign Overfitting with
Lijia Zhou (UChicago), Frederic Koehler (Stanford) and Danica Sutherland (TTIC→UBC)

Plan

- Recap from Yesterday: Classic view of Supervised Learning, Estimation vs Approximation Error, Capacity/Uniform Convergence, ***Inductive Bias***
- How does Deep Learning fit into classical view?
- Explicit vs Implicit Inductive Bias
- Benign Overfitting

- **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 $S \sim \mathcal{D}^m$):

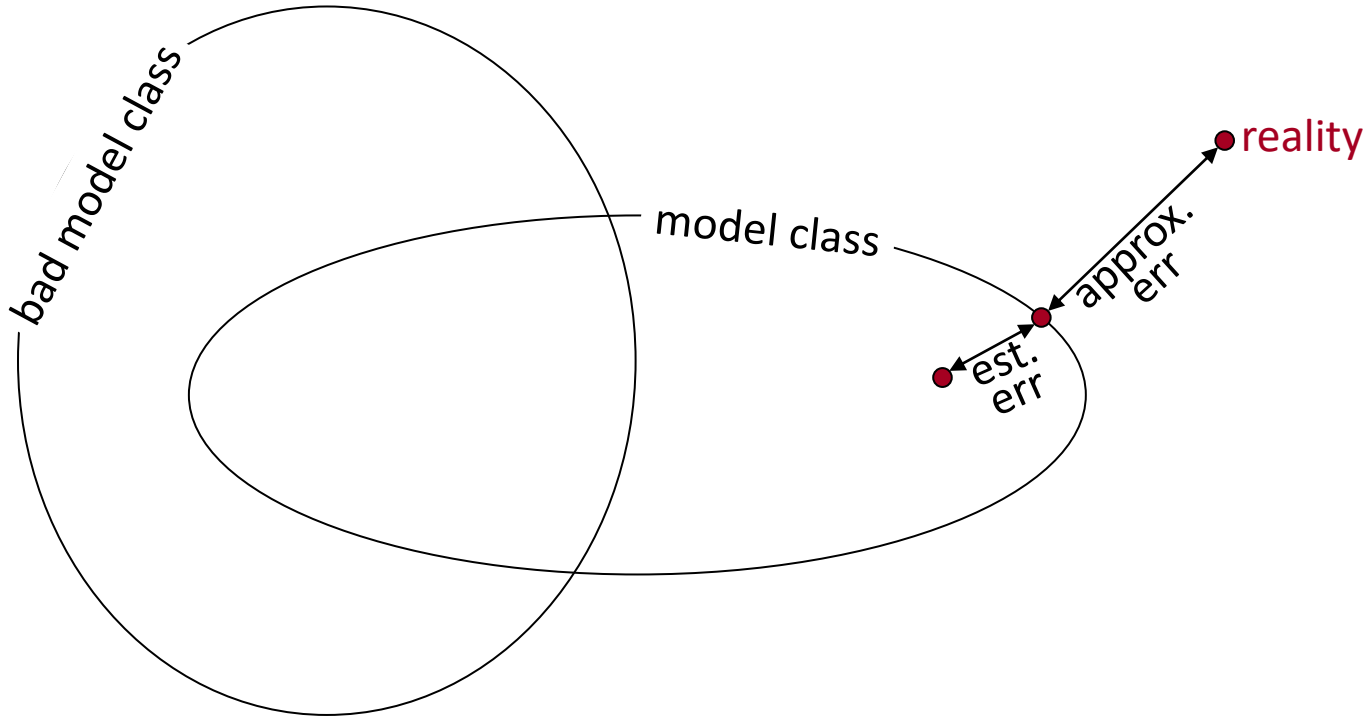
$$L\left(ERM_{\mathcal{H}}(S)\right) \leq L(h^*) + \mathcal{R}_m(\mathcal{H}) \approx L(h^*) + \sqrt{\frac{O\left(capacity(\mathcal{H})\right)}{m}}$$

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

- 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: $VCdim(\mathcal{H}) \leq \log|\mathcal{H}| \leq \#bits = \#params \cdot \#bits/param$
  - 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

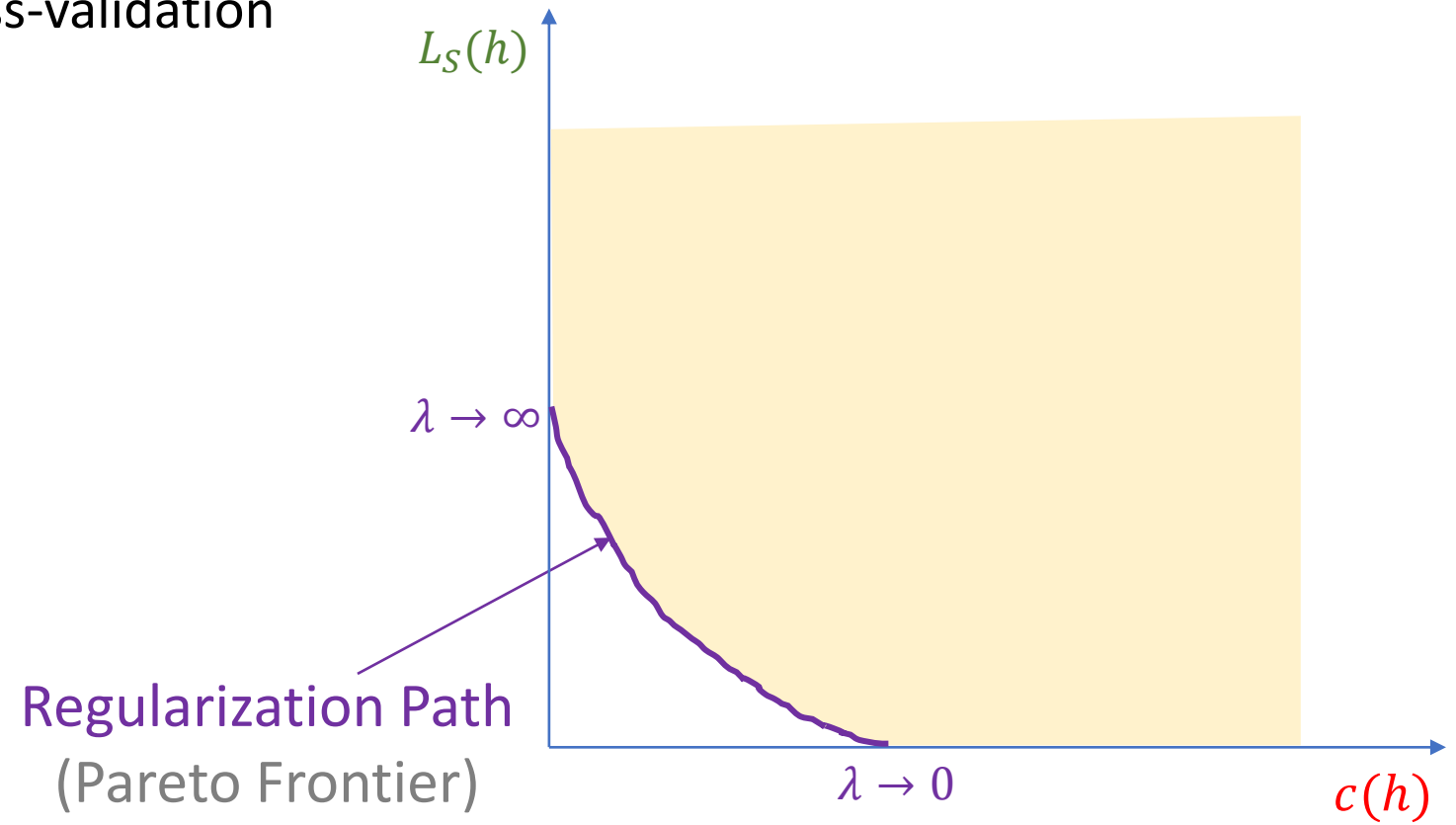
Complexity Measure as Inductive Bias

- **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. $\arg\min L(h) + \boldsymbol{\lambda}\, c(h)$ or $\arg\min L(h)$ s.t. $c(h) \leq \boldsymbol{B}$

  and choose λ or B using cross-validation

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



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: $SRM_{\mathcal{H}}(S) = \arg\min \; L(h) \;,\; c(h)$

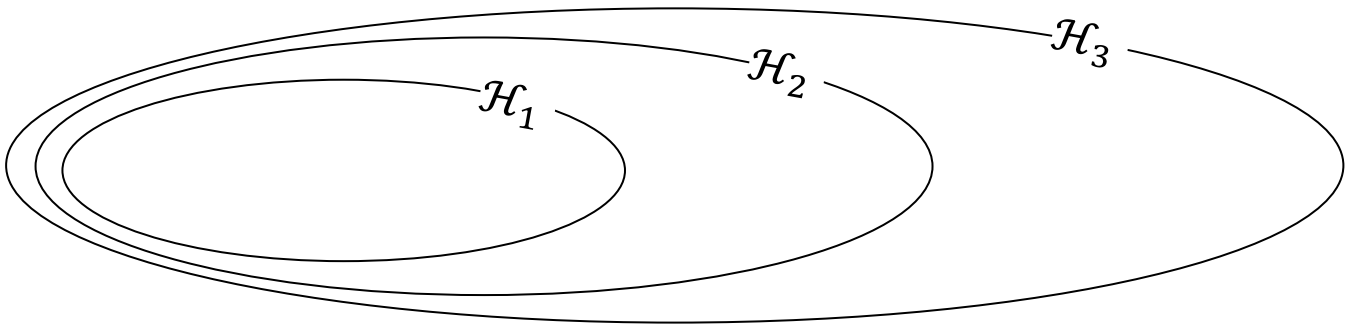
  e.g. $\arg\min L(h) + \lambda\, c(h)$ or $\arg\min L(h)$ s.t. $c(h) \leq B$

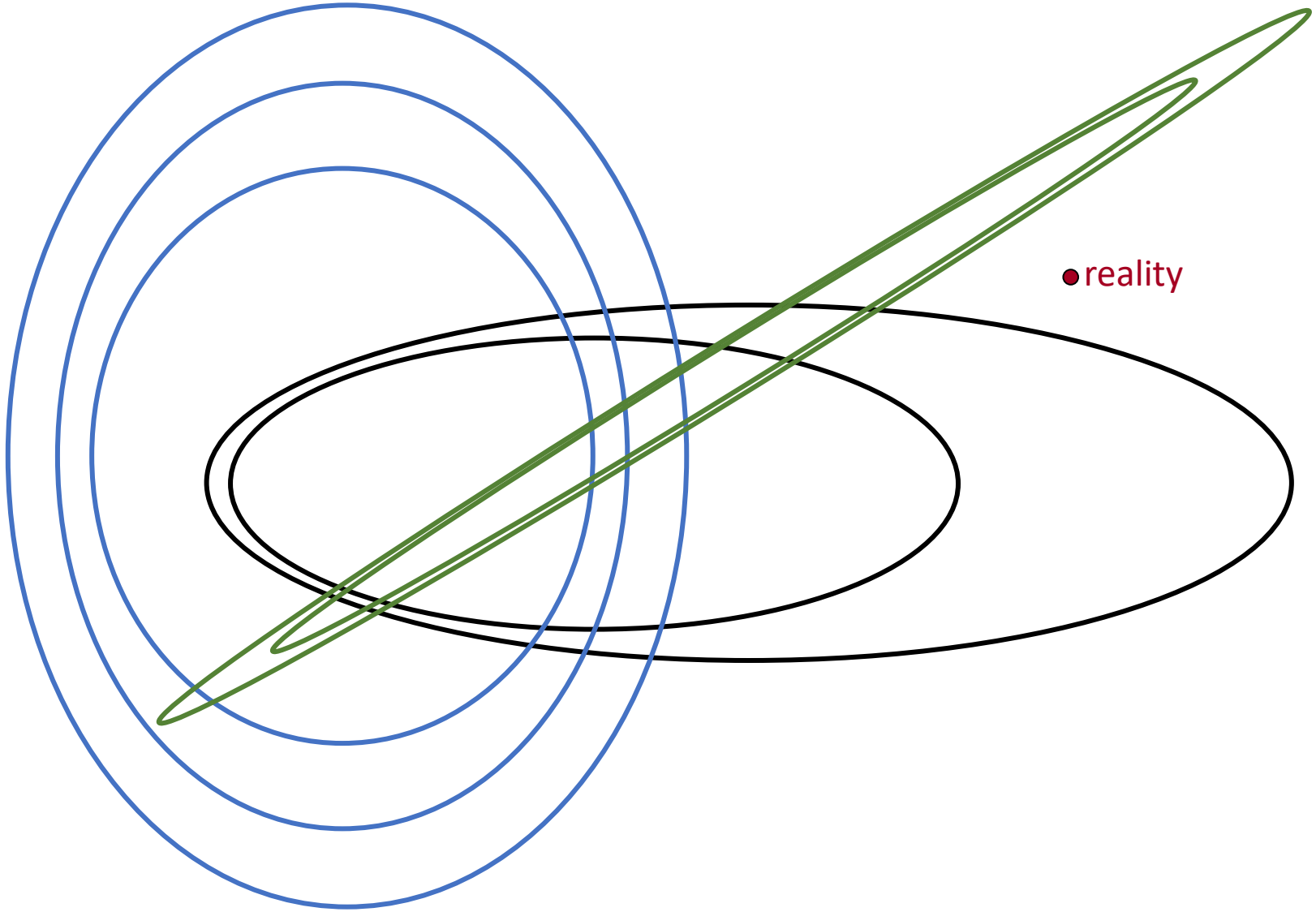
  and choose λ or B using cross-validation
- Guarantee:

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

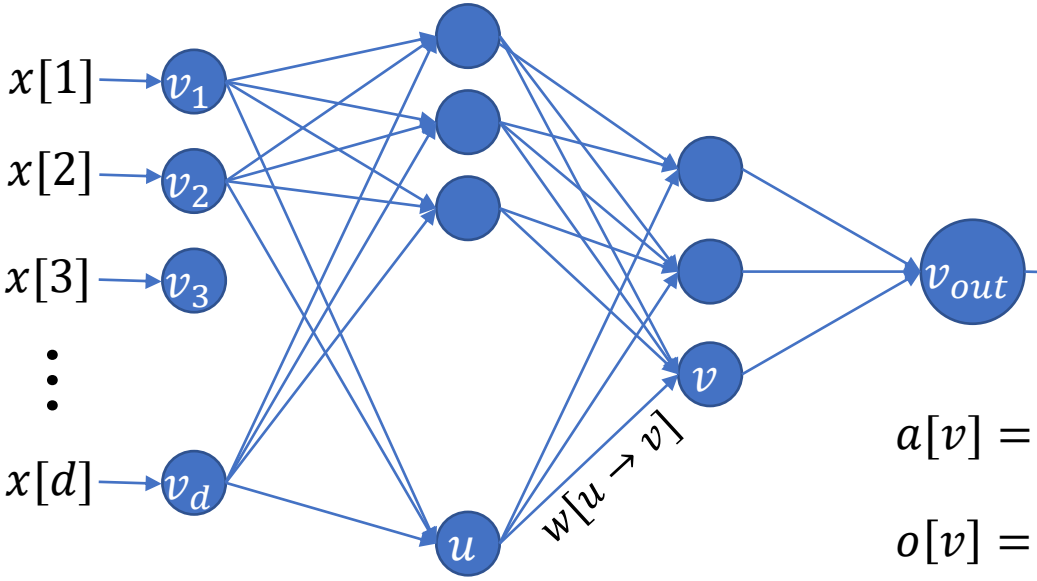
$\mathcal{H}_B = \{h | c(h) \leq B\}$

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




reality

Feed-Forward Neural Networks (The Multilayer Perceptron)



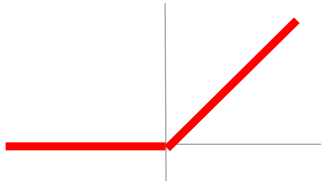
$h_{G(V,E),\sigma,w}(x)$

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

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

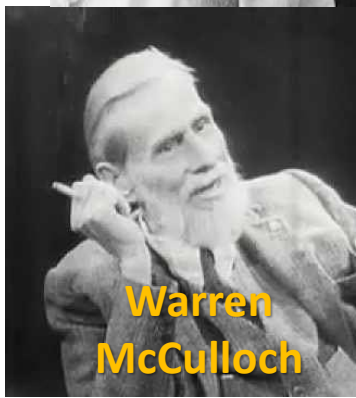
Architecture:

- 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" σ: $\mathbb{R} \to \mathbb{R}$. E.g. $\sigma_{RELU}(z) = [z]_+$



Parameters:

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



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

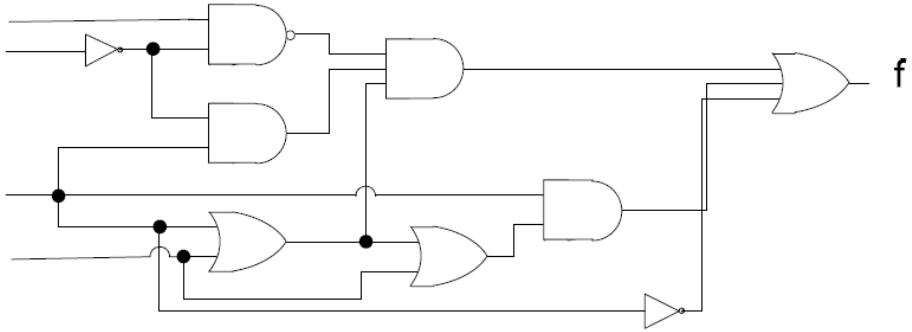
- Expressive Power / Approximation

What can Feed-Forward Networks Represent?

- ANDs (using a single unit)
- ORs (using a single unit)
- XORs (parities) (using $|E| = d^2$ with depth 2, or $|E| = O(d)$ with depth $\log(d)$)
- NOT (using a single weight)

- Any function over $\mathcal{X} = \{\pm 1\}^d$

Learning Circuits as Neural Networks



$CIRCUIT_n[depth, size]$ = functions $f: \{\pm 1\}^n \to \{0,1\}$ that can be implemented with logical circuits with at most $size$ unlimited-fan-in AND, OR and NOT gates, and longest path from input to output at most $depth$

$(AC^i \approx CIRCUIT[O(\log^i n), poly(n)])$

Learning a circuit (ie learning with the class $CIRCUIT$): learning the *architecture*

Claim: $CIRCUIT_n[depth, size] \subseteq \mathcal{H}_{G_{n,L=depth,k=size}, sign}$

Fully connected layer graph, with L(=*depth*) layers and k(=size) nodes in each layers.

- Weights are ± 1 if connected in the circuit (with/without a NOT gate in between), 0 otherwise;
- Bias terms are *fanin-1* for AND, *1-fanin* for OR

What can Feed-Forward Networks Represent?

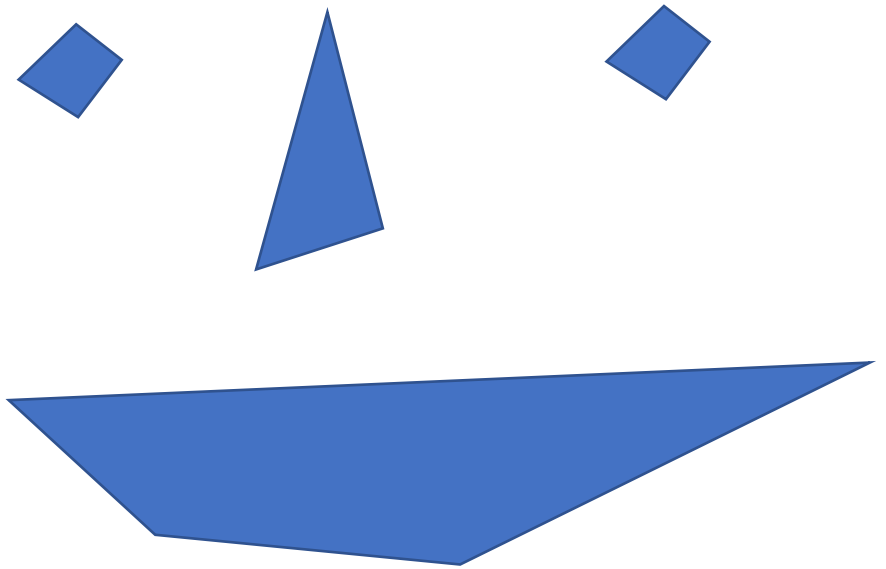
- Any function over $\mathcal{X} = \{\pm 1\}^d$
  - As a circuit
  - E.g. using DNF (OR of ANDS), with a single hidden layer of ANDs, output output unit implementing OR
  - $|V| = 2^d, |E| = d2^d$
  - Like representing the truth table directly…

- Universal Representation Theorem: Any continuous functions $f: [0,1]^d \rightarrow \mathbb{R}$ can be approximated to to within ϵ (for 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

What can SMALL Networks Represent?

- Intersection of halfspaces
  - Using single hidden layer (the halfspaces; output unit does AND)

- Union of intersection of halfspaces
  - Using two hidden layers (halfspaces→OR→AND)



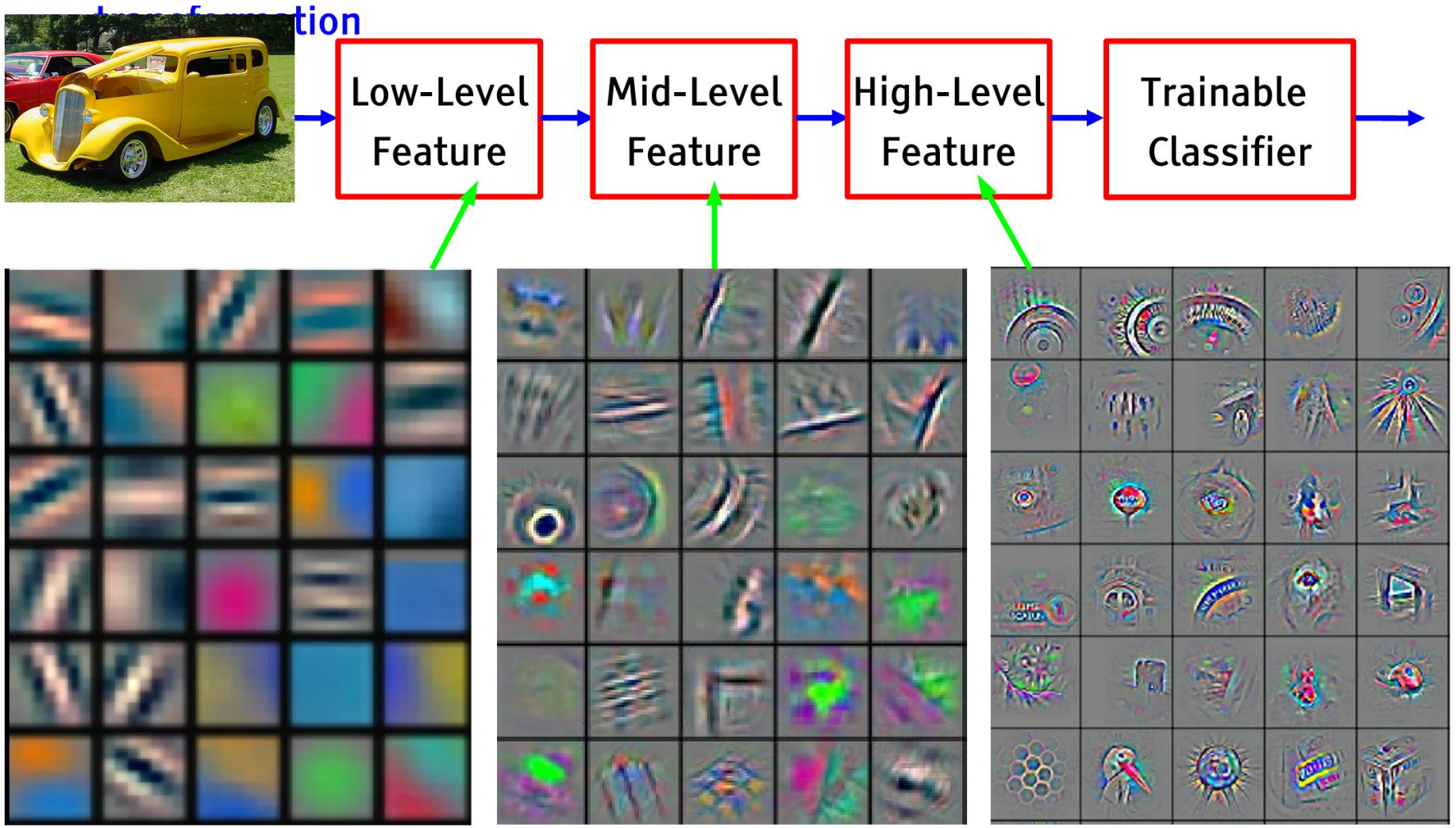
What can SMALL Networks Represent?

- Intersection of halfspaces
  - Using single hidden layer (the halfspaces; output unit does AND)

- Union of intersection of halfspaces
  - Using two hidden layers (halfspaces→OR→AND)

- Feature learning:
  Linear predictors over (small number of) features,
  in turn represented as linear predictors over more basic features,
  that in turn are also represented as linear predictors

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

- Expressive Power / Approximation
  - Any continuous function with huge network
  - Lots of interesting things naturally with small networks
  - Realities captures by hierarchies of progressively complex features
  - **Any time T computable function with network of size $\widetilde{\boldsymbol{O}}(\boldsymbol{T})$**

    Using a depth-T network, since anything computable in time T is also computable using a logical circuit 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 $\mathbb{E}[L(A(S))] \leq 0.4$)

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+Search 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
  - $\widetilde{\boldsymbol{O}}(|\boldsymbol{E}|)$ (number of edges, i.e. number of weights)
    (with threshold σ, or with RELU and finite precision; RELU with inf precision: $\widetilde{\Theta}(|E| \cdot depth)$)

- 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 $\widetilde{\boldsymbol{O}}(\boldsymbol{T})$**

- 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 (or short runtime) Programs

- Universal
- Captures anything we want with reasonable sample complexity
- ERM is NP-hard
- Provably hard to learn even improperly, with any rule (subject to crypto)
- Hard to optimize in practice
  Short programs: Incomputable.
  Even if we limit to bounded-time:
  - No practical local search
  - Highly non-continuous, disconnected discrete space
  - Not much success

Deep Networks

- Universal
- Captures anything we want with reasonable sample complexity
- ERM is NP-hard
- Provably hard to learn even improperly, with any rule (subject to crypto)
- Often easy to optimize
  - **Continuous**
  - Amenable to local search with Grad Descent, or SGD
  - **Lots of empirical success**

Feed Forward Neural Networks

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

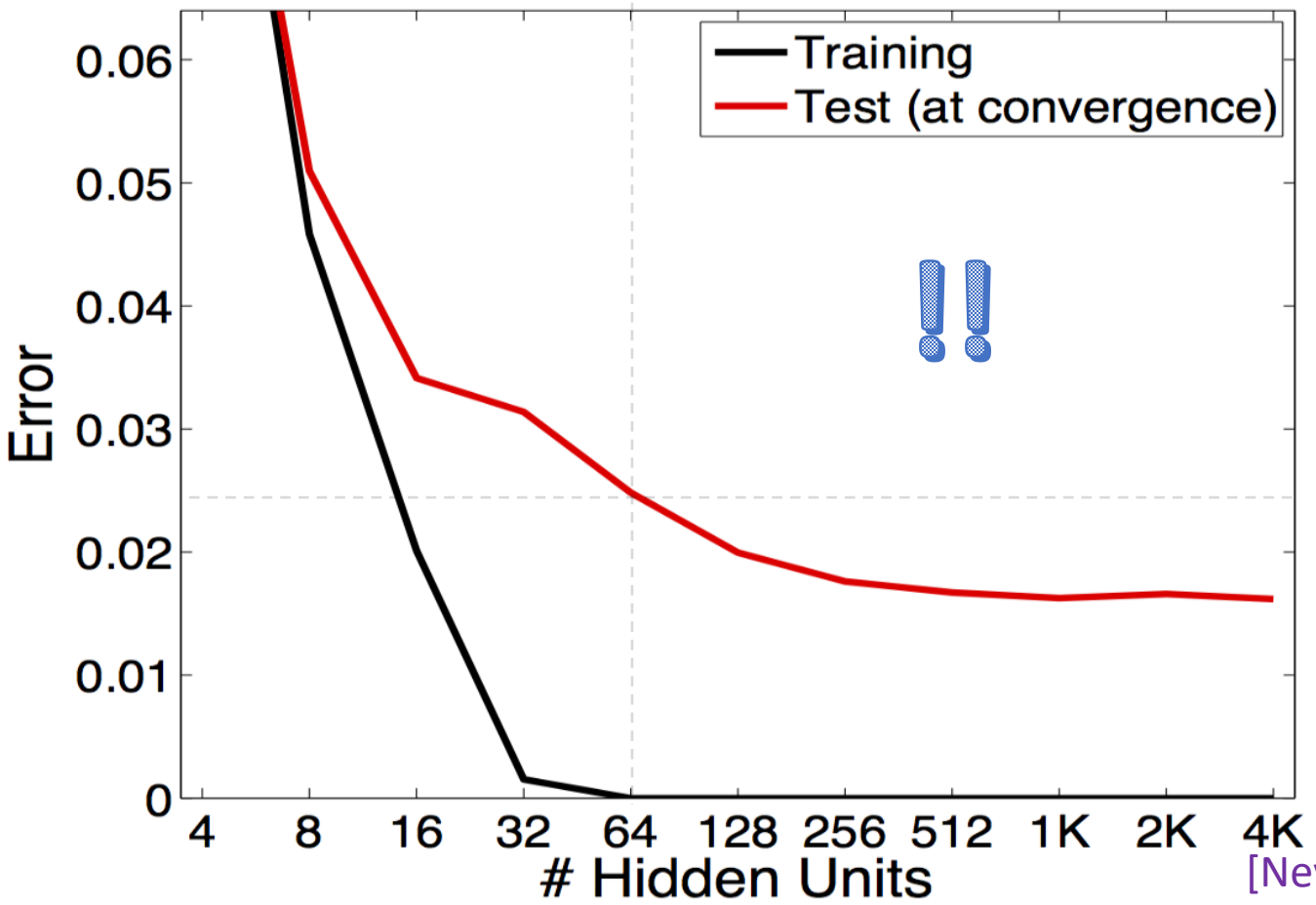
$$\mathcal{H}_{G(V,E),\sigma} = \{\, f_w(x) = output\ of\ net\ with\ weights\ w \,\}$$

- 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: $\widetilde{\Theta}(|E| \cdot depth)$)

- 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 $\widetilde{O}(T)$**

- 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"



[Neyshabur Tomioka S ICLR'15]

model	# params	random crop	weight decay	train accuracy	test accuracy
Inception	1,649,402	yes	yes	100.0	89.05
		yes	no	100.0	89.31
		no	yes	100.0	86.03
		no	no	100.0	85.75
(fitting random labels)		no	no	100.0	9.78
Inception w/o BatchNorm	1,649,402	no	yes	100.0	83.00
		no	no	100.0	82.00
(fitting random labels)		no	no	100.0	10.12
Alexnet	1,387,786	yes	yes	99.90	81.22
		yes	no	99.82	79.66
		no	yes	100.0	77.36
		no	no	100.0	76.07
(fitting random labels)		no	no	99.82	9.86
MLP 3x512	1,735,178	no	yes	100.0	53.35
		no	no	100.0	52.39
(fitting random labels)		no	no	100.0	10.48
MLP 1x512	1,209,866	no	yes	99.80	50.39
		no	no	100.0	50.51
(fitting random labels)		no	no	99.34	10.61

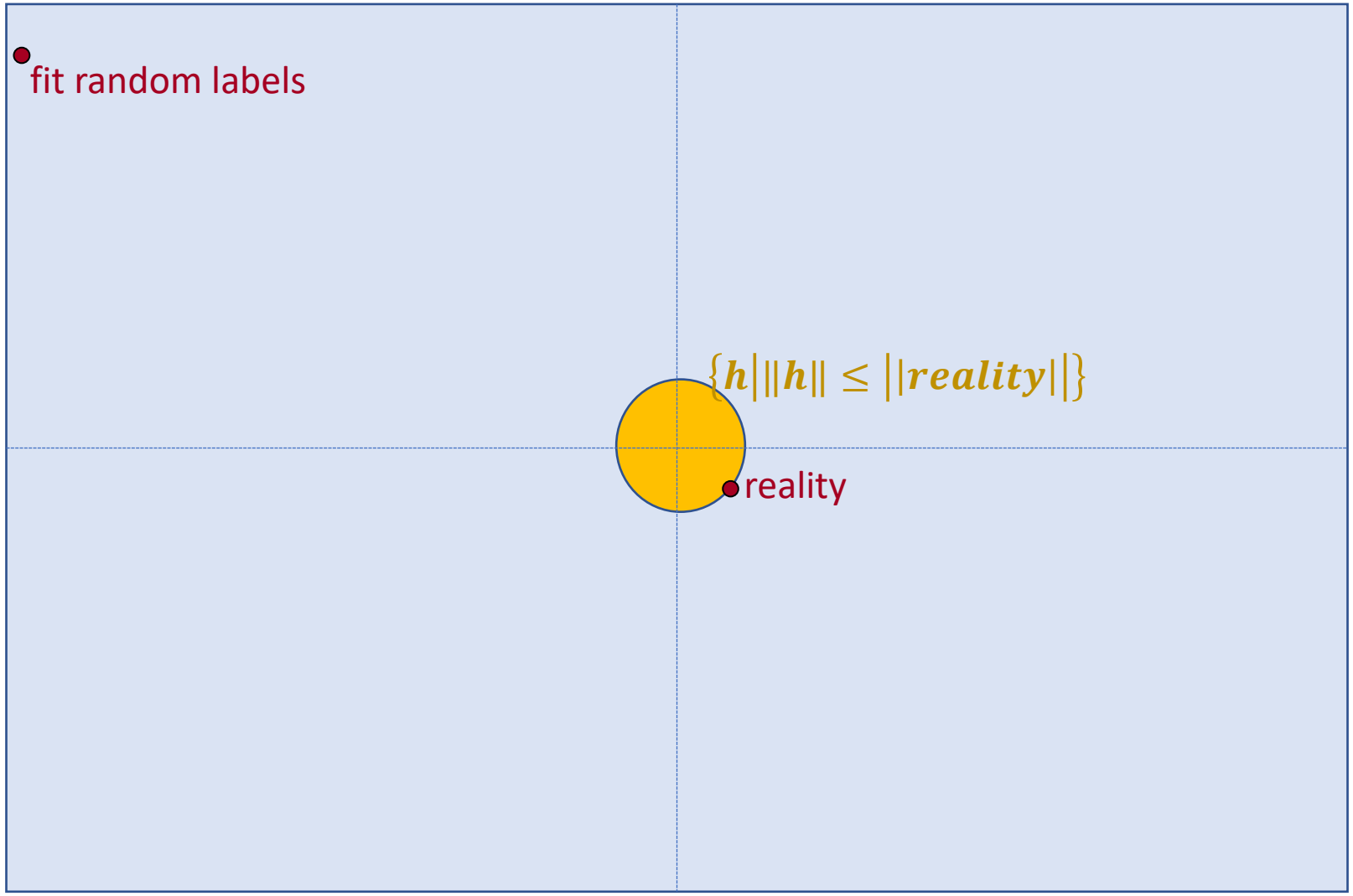
UNDERSTANDING DEEP LEARNING REQUIRES RE-THINKING GENERALIZATION

Zhang, Bengio, Hardt, Recht, Vinyals 2017

Learning with a Rich Function Class

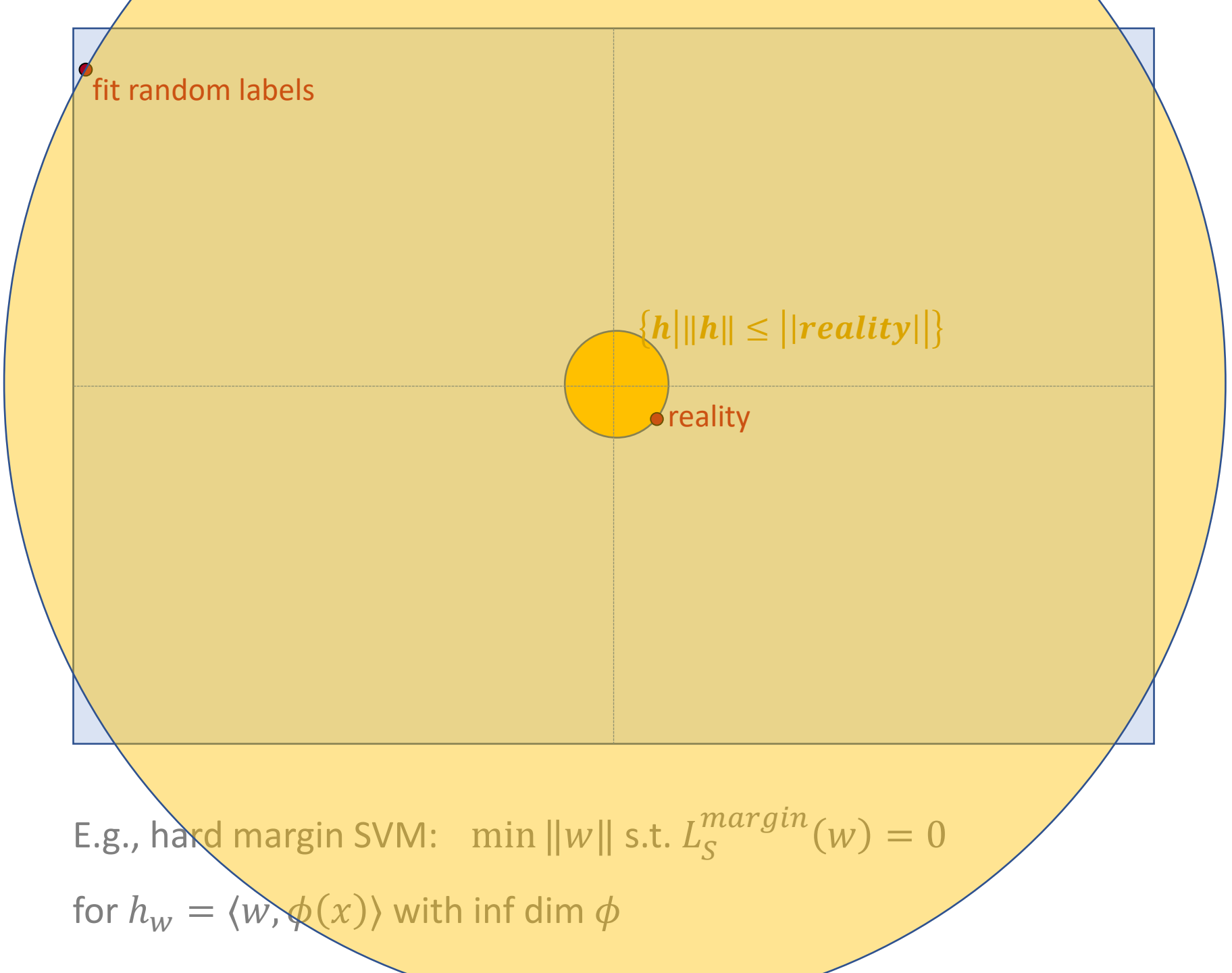
- Learning rule $A(S)$ s.t.
  - For any data set, even with random labels, can fit data: $L_S\big(A(S)\big) = 0$
  - For "real" data $S \sim \mathcal{D}^m$ sampled from reasonable reality $\mathcal{D}$, we can generalize: $L_{\mathcal{D}}\big(A(S)\big)$ is low
- Examples:
  - 1-Nearest Neighbor: if realizable by some continuous h^* (ie $L(h^*) = 0$), then consistent: $L_{\mathcal{D}}(1NN(S)) \xrightarrow{|S|\to\infty} 0$
  - Hard Margin SVM with Gaussian Kernel (or other universal kernel) or more generally min norm consistent solution: $\arg\min \|h\|_K \quad s.t.\ L_S(h) = 0$

$$\equiv \arg\min_{\langle w,\phi(x_i)\rangle = y_i} \|w\|_2$$



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

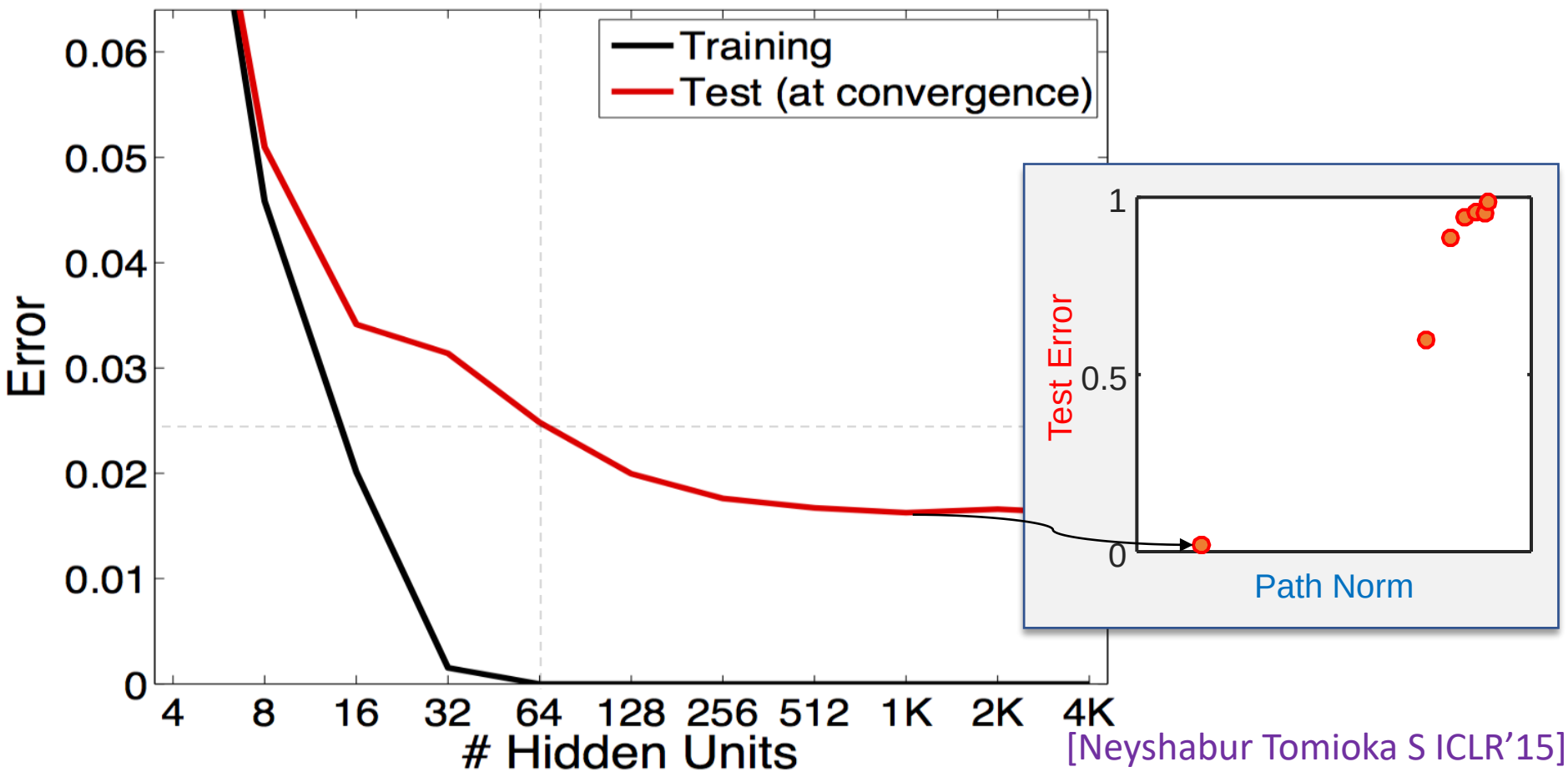
Learning with a Rich Function Class

- Learning rule $A(S)$ s.t.
  - For any data set, even with random labels, can fit data: $L_S\big(A(S)\big) = 0$
  - For "real" data $S \sim \mathcal{D}^m$ sampled from reasonable reality $\mathcal{D}$, we can generalize: $L_{\mathcal{D}}\big(A(S)\big)$ is low
- Examples:
  - 1-Nearest Neighbor: if realizable by some continuous h^* (ie $L(h^*) = 0$), then consistent: $L_{\mathcal{D}}(1NN(S)) \xrightarrow{|S|\to\infty} 0$
  - Hard Margin SVM with Gaussian Kernel (or other universal kernel) or more generally min norm consistent solution: $\arg\min \|h\|_K \;\; s.t. \; L_S(h) = 0$

    $$\equiv \arg\min_{\langle w,\phi(x_i)\rangle = y_i} \|w\|_2$$

    - Can always get $L_S(h) = 0$
    - If $\exists h^*, L_{\mathcal{D}}(h^*) = 0$, generalizes with sample complexity $|S| = O\big(\|h\|_K^2\big)$
  - MDL: $\arg\min|prog| \;\; s.t. \;\; L_S\,(prog) = 0$

    $L\big(MDL(S)\big) \le O\left(\frac{|prog^*|}{|S|}\right)$ if realizable by $prog^*$

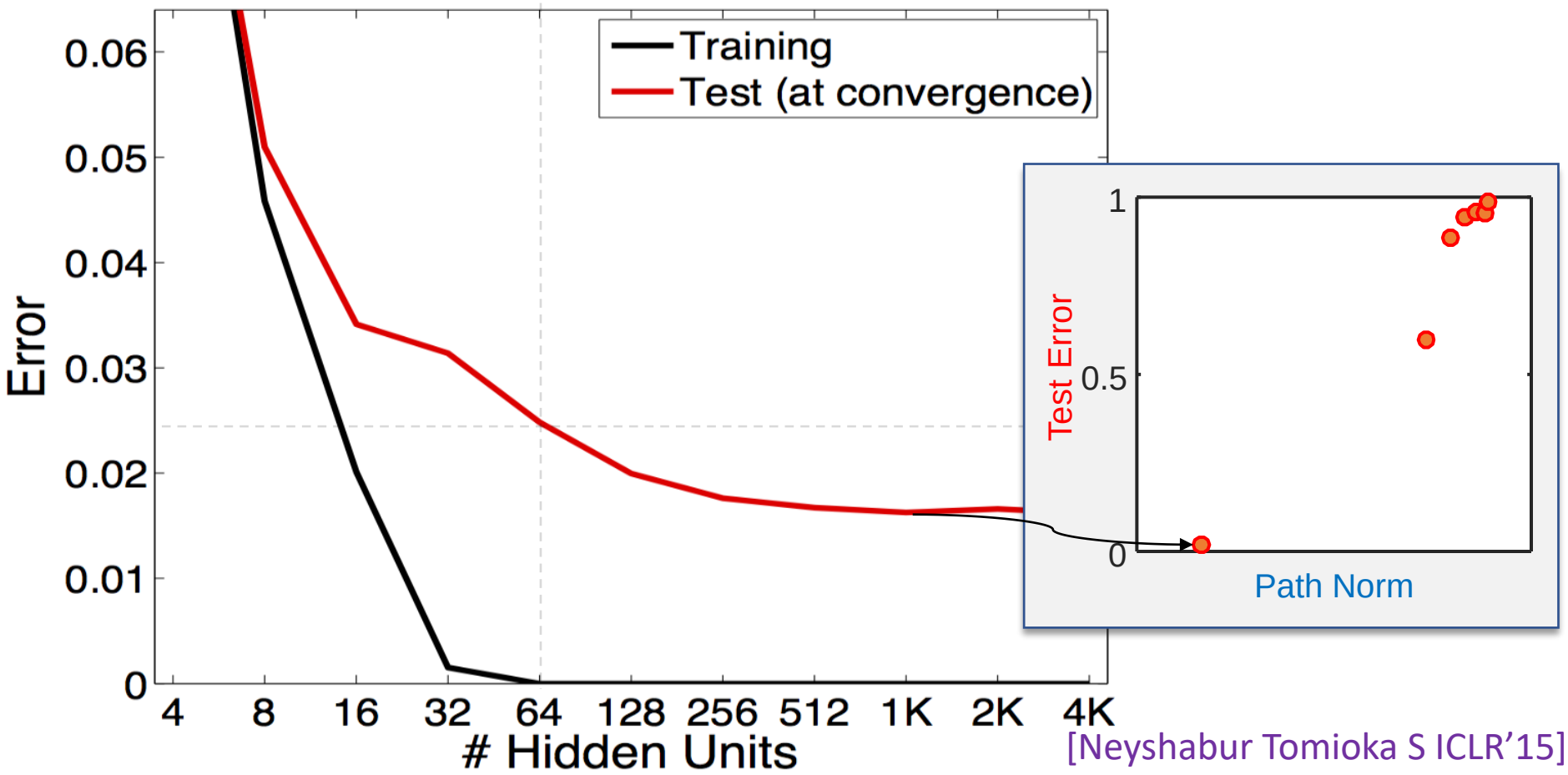


[Neyshabur Tomioka S ICLR'15]

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?

Peter L. Bartlett

model	# params	random crop	weight decay	train accuracy	test accuracy
Inception	1,649,402	yes	yes	100.0	89.05
		yes	no	100.0	89.31
		no	yes	100.0	86.03
		no	no	100.0	85.75
(fitting random labels)		no	no	100.0	9.78
Inception w/o BatchNorm	1,649,402	no	yes	100.0	83.00
		no	no	100.0	82.00
(fitting random labels)		no	no	100.0	10.12
Alexnet	1,387,786	yes	yes	99.90	81.22
		yes	no	99.82	79.66
		no	yes	100.0	77.36
		no	no	100.0	76.07
(fitting random labels)		no	no	99.82	9.86
MLP 3x512	1,735,178	no	yes	100.0	53.35
		no	no	100.0	52.39
(fitting random labels)		no	no	100.0	10.48
MLP 1x512	1,209,866	no	yes	99.80	50.39
		no	no	100.0	50.51
(fitting random labels)		no	no	99.34	10.61

UNDERSTANDING DEEP LEARNING REQUIRES RE-THINKING GENERALIZATION

Zhang, Bengio, Hardt, Recht, Vinyals 2017

Where is the regularization?

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

$$X \in \mathbb{R}^{m \times d}, y \in \mathbb{R}^m, m \ll d$$

- Claim: **Gradient Descent** (or SGD), initialized at $w_0 = 0$, converges to min norm solution

$$\min_{Xw=y} \|w\|_2$$

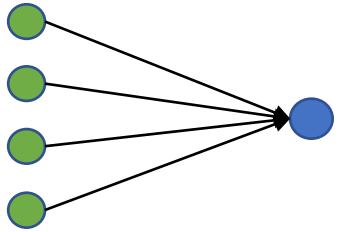
➢ Proof: iterates always spanned by rows of X

- **Coordinate Descent**, initialized at $w_0 = 0$, related to, but not quite

$$\min_{Xw=y} \|w\|_1 \quad \textbf{(Lasso)}$$

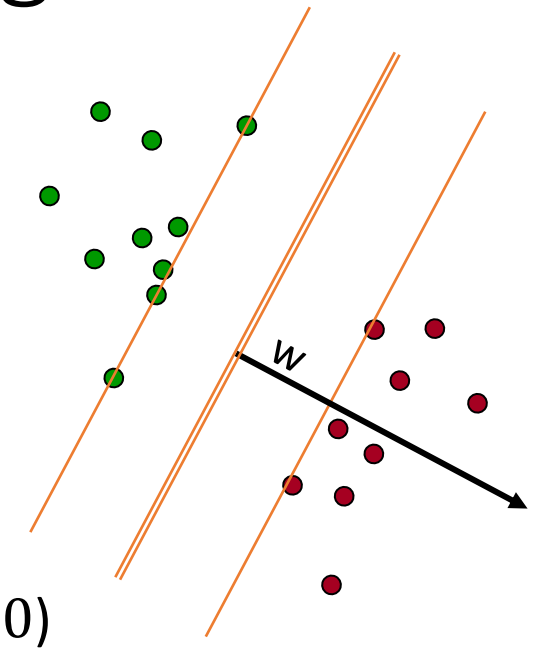
(with stepsize↘ 0 and particular tie-breaking ≈ **LARS**)

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?

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

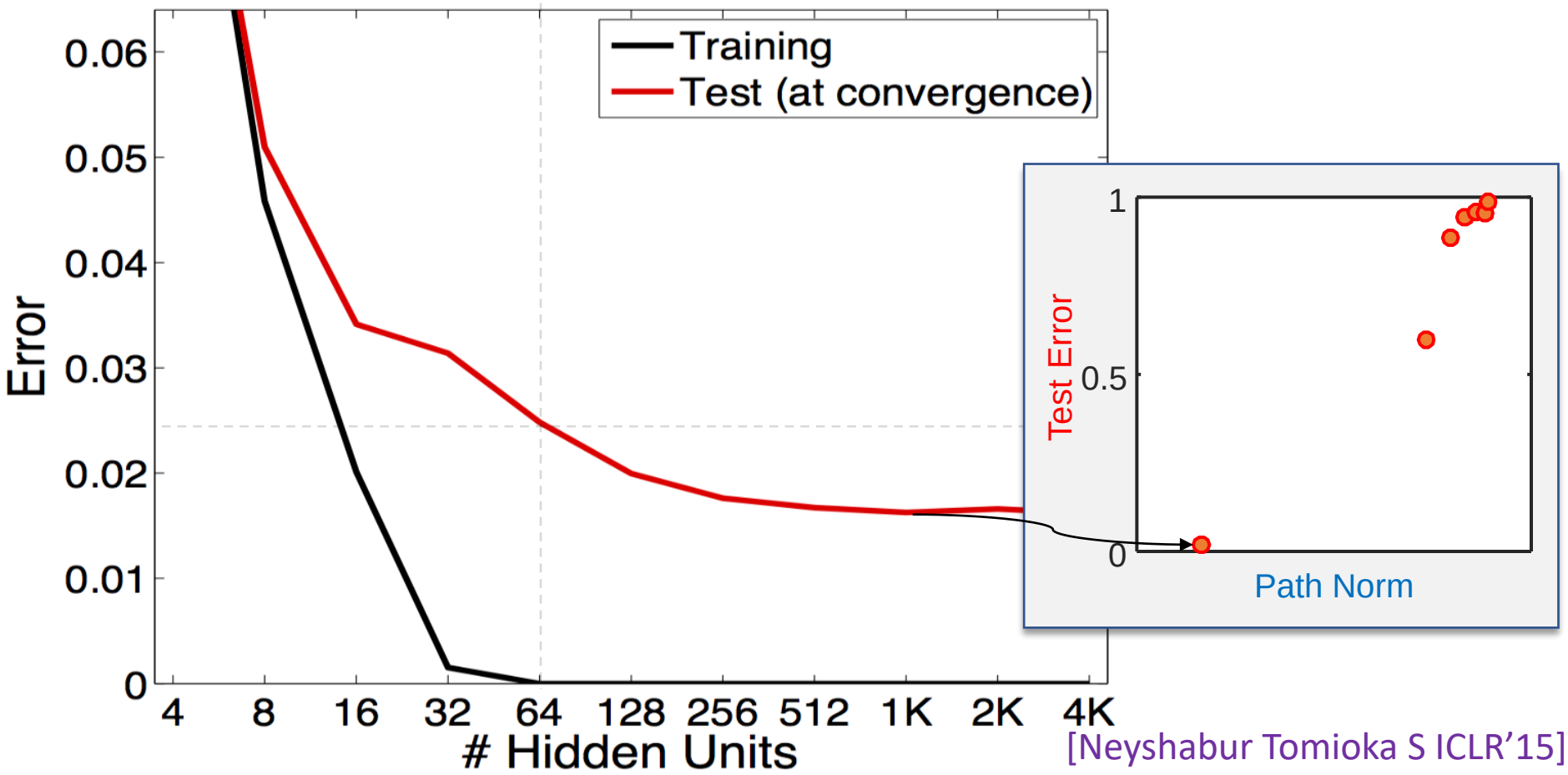
  - $\inf \mathcal{L}(w) = 0$, but minima unattainable
  - GD diverges to infinity: $w(t) \to \infty$, $\mathcal{L}\big(w(t)\big) \to 0$

- **In what direction?** What does $\frac{w(t)}{\|w(t)\|}$ converge to?
- **Theorem**: $\frac{w(t)}{\|w(t)\|_2} \to \frac{\hat{w}}{\|\hat{w}\|_2}$ $\quad \hat{w} = \arg\min\|w\|_2 \ s.t. \forall_i y_i\langle w, x_i\rangle \geq 1$

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

Implicit Bias in Logistic Regression

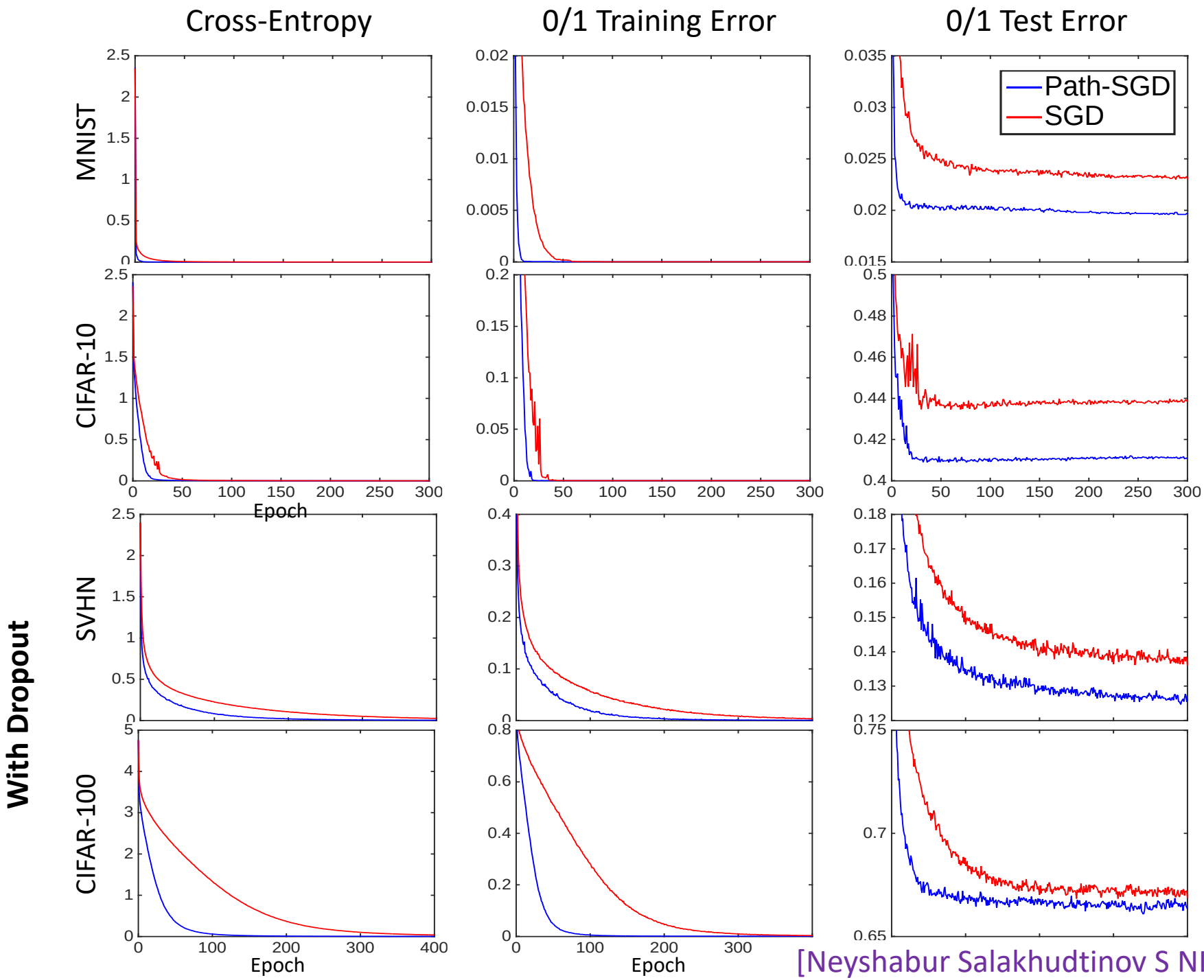
- Single linear unit, logistic loss
  ➔ hard margin SVM solution (regardless of init)
- Multi-class problems with softmax loss
  ➔ multiclass SVM solution (regardless of init)
- Steepest Descent w.r.t. $\|w\|$
  ➔ $\arg\min\|w\| \;\; s.t. \forall_i y_i \langle w, x_i \rangle \geq 1$ (regardless of init)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



[Neyshabur Tomioka S ICLR'15]

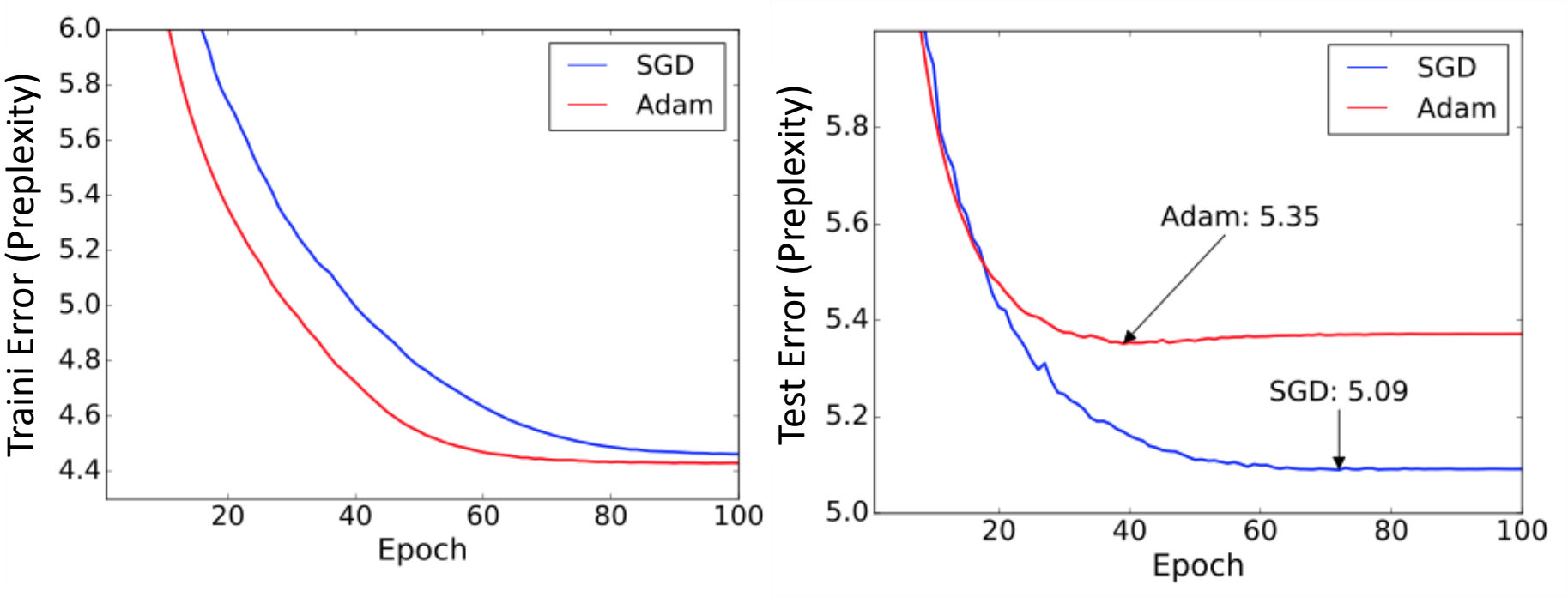
Peter L. Bartlett

- 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

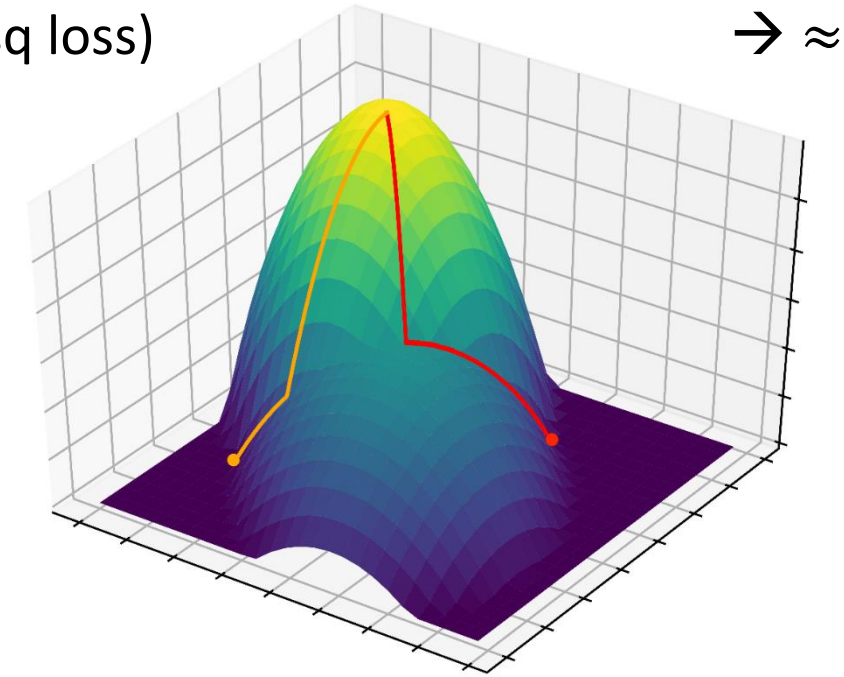
➔ Different generalization properties

Grad Descent

→ $\min_{Xw=y} \|w\|_2$ (sq loss)

Coordinate Descent

→ $\approx \min_{Xw=y} \|w\|_1$ (sq loss)



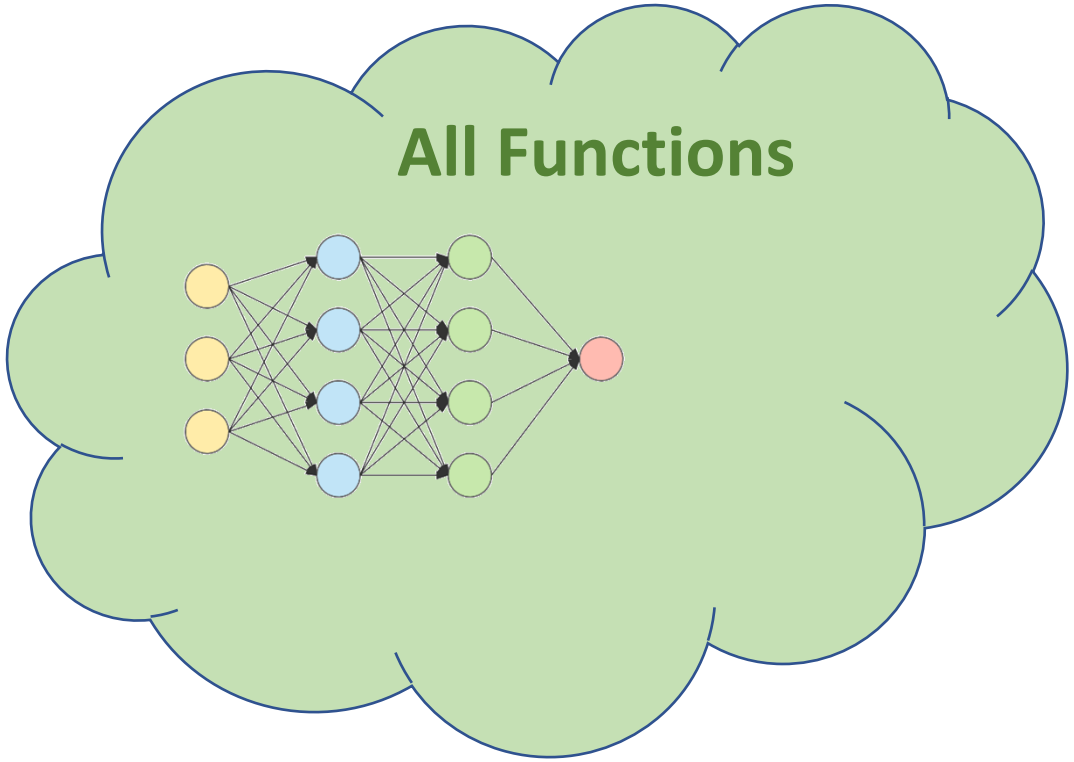
Need to understand optimization alg. not just as reaching ***some*** (global) optimum, but as reaching a ***specific*** optimum

Different optimization algorithm

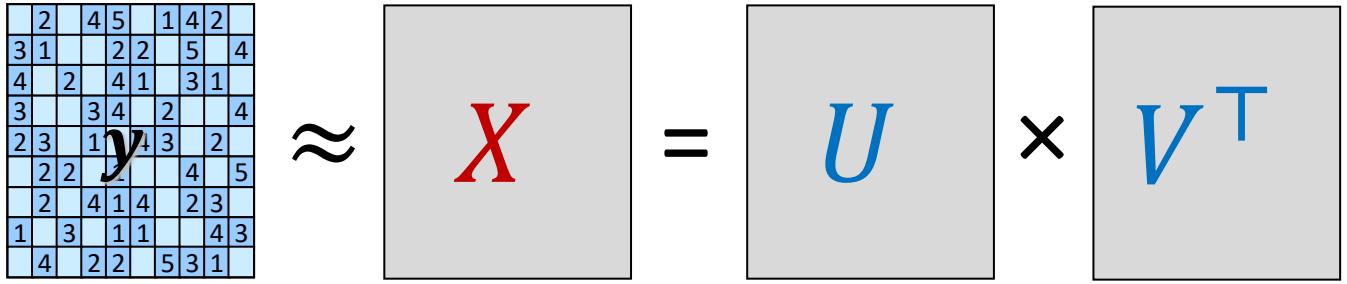
➔ Different bias in optimum reached

➔ Different Inductive bias

➔ Different generalization properties

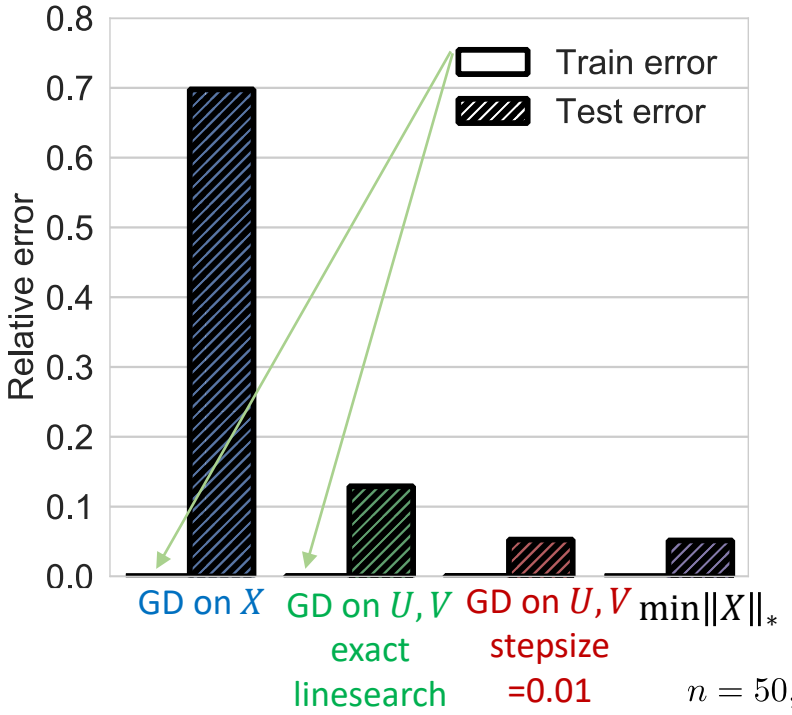


Need to understand optimization alg. not just as reaching ***some*** (global) optimum, but as reaching a ***specific*** optimum



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

- 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



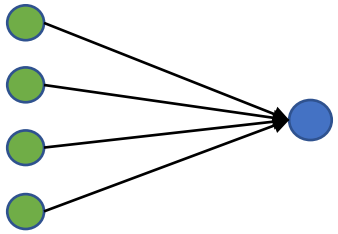
Grad Descent on $U, V \overset{???}{\rightarrow}$ **min $\|X\|_*$ solution**
(with inf. small stepsize and initialization)
→ good generalization if Y (aprox) low rank
[Gunasekar Woodworth Bhojanapalli Neyshabur S 2017]

When $y = \langle A_i, W^* \rangle$, W^* low rank, 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
$y = \mathcal{A}(X^*) + \mathcal{N}(0, 10^{-3}),\ y_{\text{test}} = \mathcal{A}_{\text{test}}(X^*) + \mathcal{N}(0, 10^{-3})$

Single Overparametrized Linear Unit



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

→ **Hard Margin SVM predictor**

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

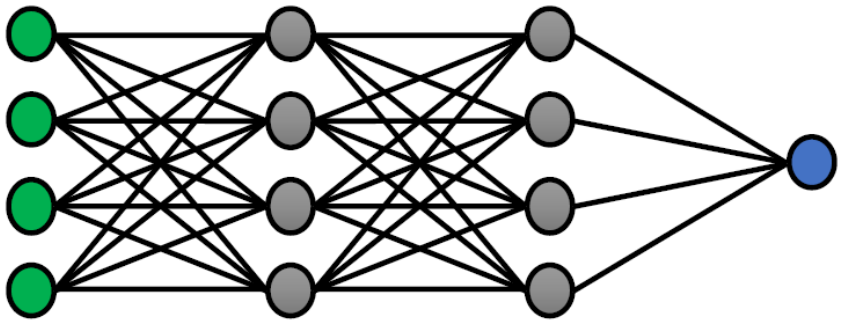
Even More Overparameterization: Deep Linear Networks

Network implements a linear mapping:

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

Training: same opt. problem as logistic regression:

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

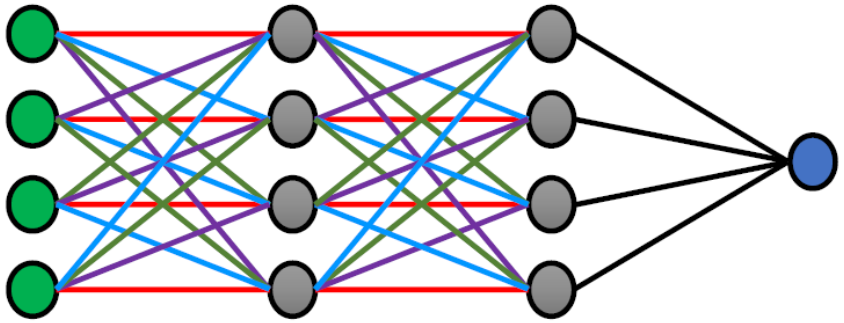


Train w with SGD

→ **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

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

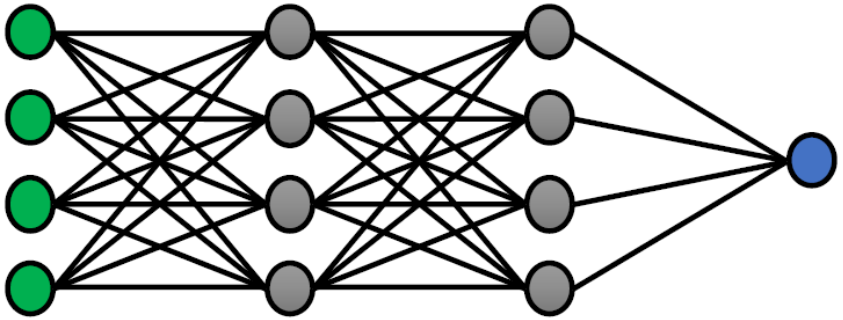
Discrete Fourier Transform

With multiple conv layers

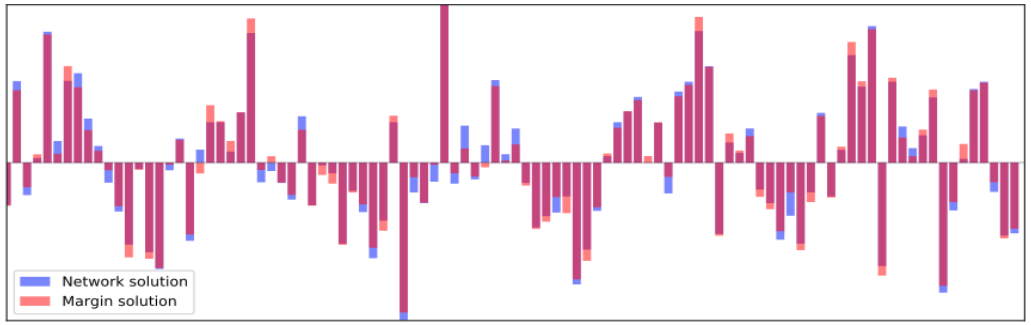
→ critical point of $\mathbf{min} \|\boldsymbol{DFT}(\boldsymbol{\beta})\|_{2/L} \; s.t. \forall_i y_i \langle \beta, x_i \rangle \geq 1$

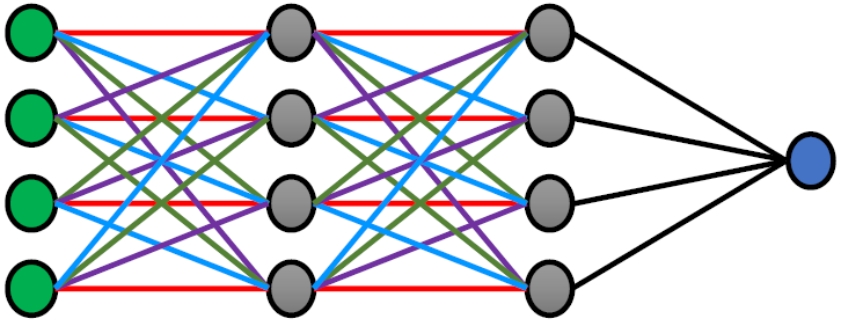
for $\ell(z) = \exp(-z)$, almost all linearly separable data sets and initializations $w(0)$ and any bounded stepsizes s.t. $\mathcal{L} \to 0$, and $\Delta w(t)$ converge in direction

[Gunasekar Lee Soudry S 2018]

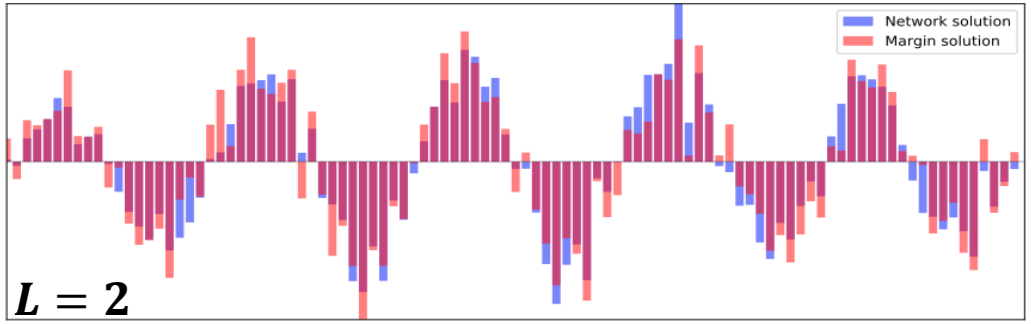


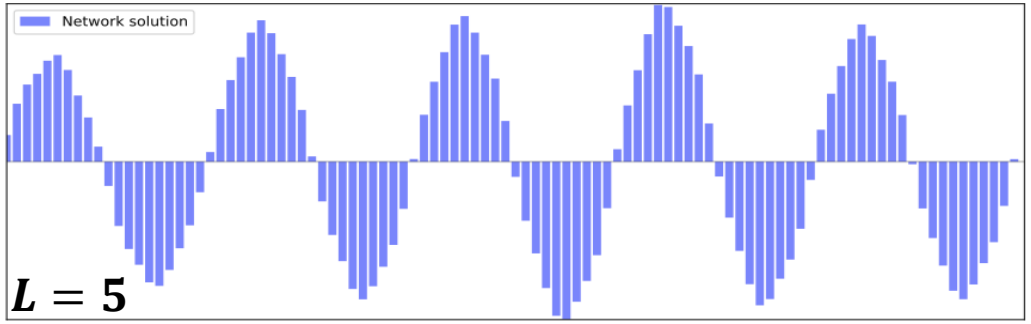
$$\min \|\boldsymbol{\beta}\|_2 \; s.t. \; \forall_i y_i \langle \beta, x_i \rangle \geq 1$$

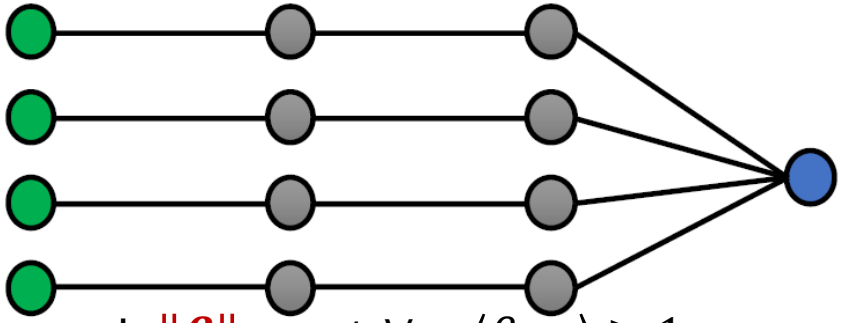




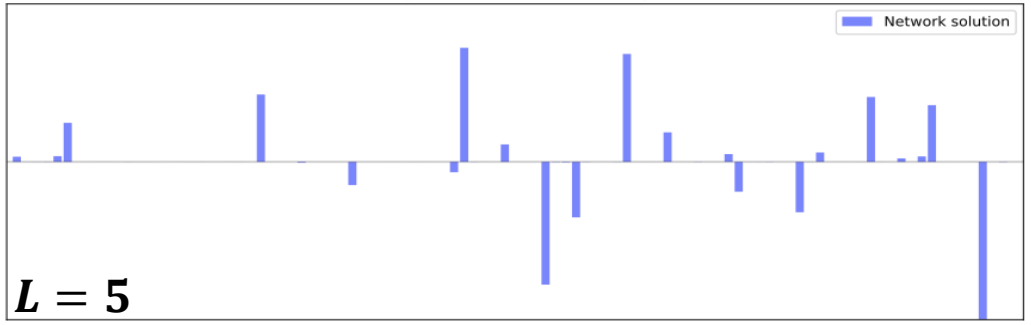
$$\min \|\boldsymbol{DFT}(\boldsymbol{\beta})\|_{2/L} \; s.t. \; \forall_i y_i \langle \beta, x_i \rangle \geq 1$$







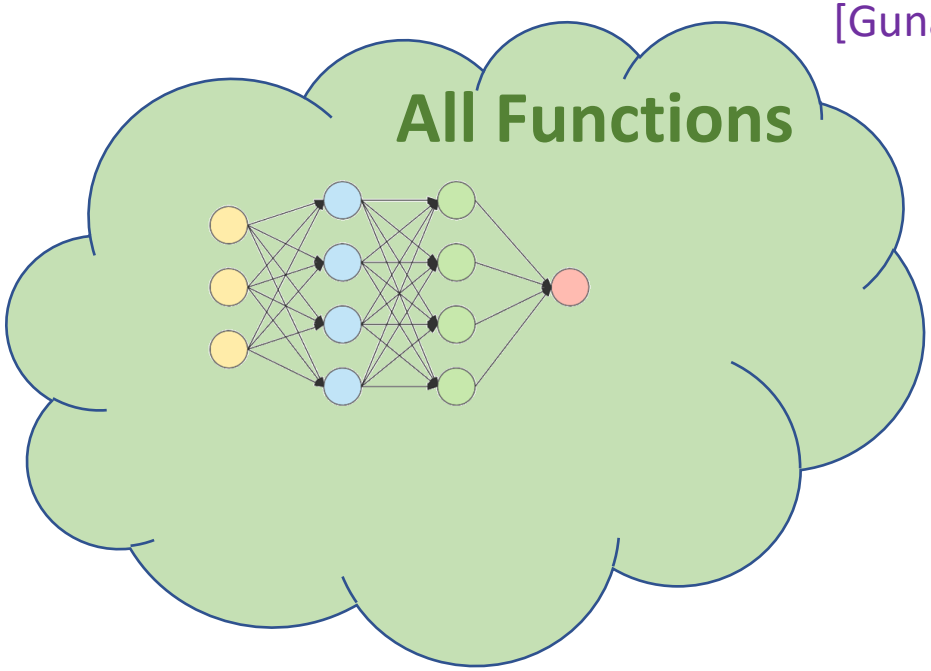
$$\min \|\boldsymbol{\beta}\|_{2/L} \; s.t. \; \forall_i y_i \langle \beta, x_i \rangle \geq 1$$



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

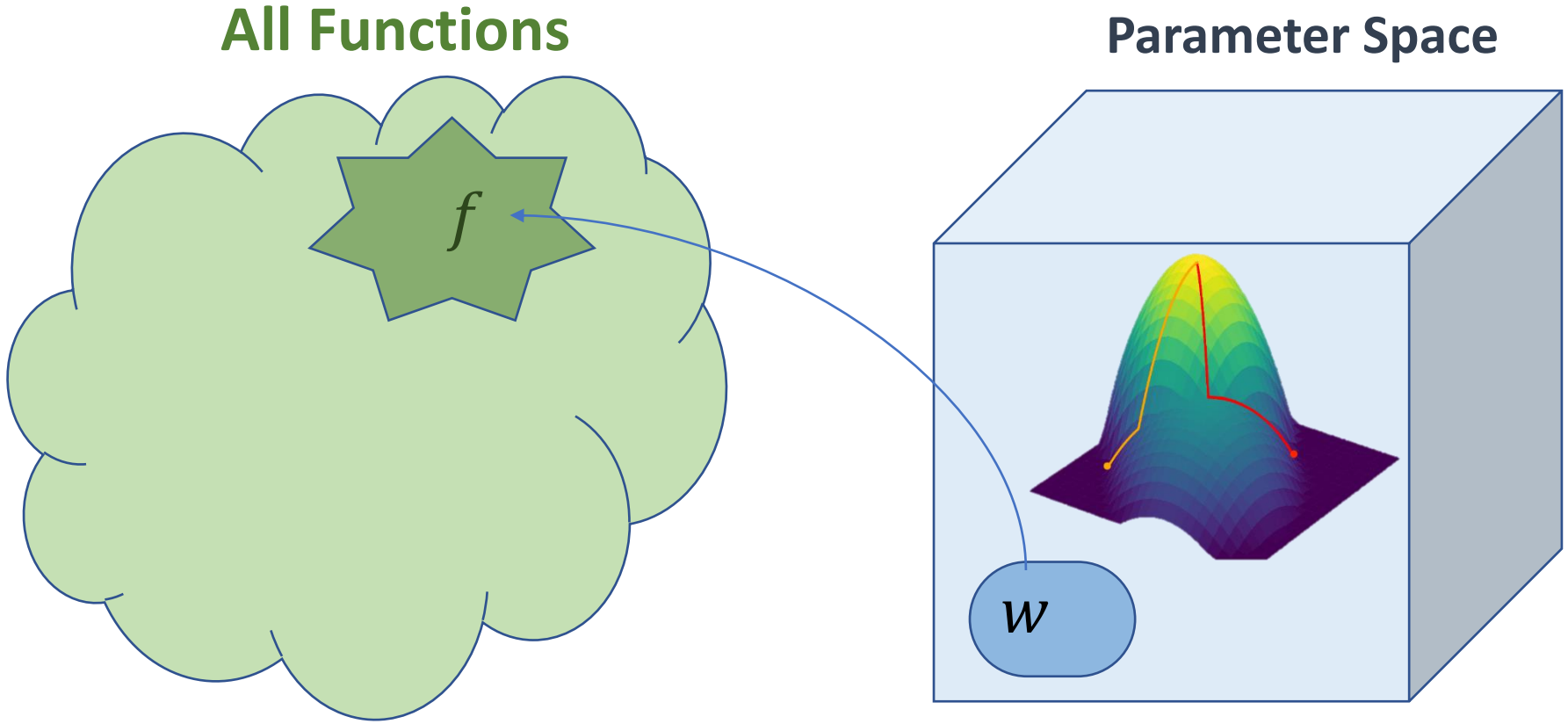
[Gunasekar Lee Soudry S 2018b]

- **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)

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

(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:

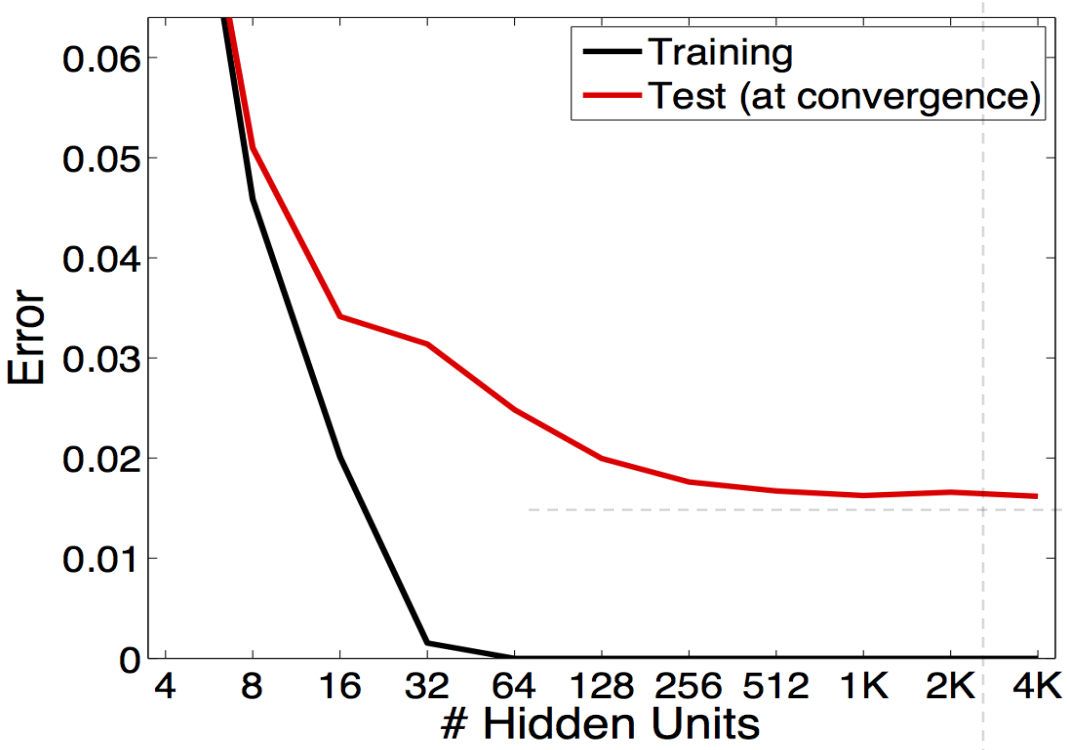
- Choice of parameterization (**architecture**)
- Geometry of local search in parameter space
- Optimization choices: Initialization, Batch Size, Step Size, etc

Understanding Learning via Local Search in Highly Underdetermined Models

The "complexity measure" approach:

- **Identify $c(h)$ such that optimization implicitly seek low $c(h)$ solution**
  $\arg\min c(h)\ s.t.\ L(h) = 0$, or at least approximately
  - How do different optimization choices affect $c(h)$?
    - Initialization scale **[Woodworth Gunasekar Lee Moroshko Sevarese Golan Soudry S 2019]**
    - Stepsize **[Nacson Ravichandran S Soudry 2022]**
    - Early stopping / Optimization accuracy **[Moroshko Gunasekar Woodworth Lee S Soudry 2020]**
    - Stochasticity: - Batchsize **[Pesme Pillaud-Vivien Flammarion 2021]**
      - Label noise **[HaoChen, Wei, Lee, Ma 2020][Blanc, Gupta, Valiant, Valiant 2020]**
  - How does architecture choice effect $c(h)$?
- **Understand generalization properties ensures by low $c(h)$**
- **Understand why in reality $\exists h^*$ with low $c(h)$ and low $L(h)$**

- In general, optimization bias not captured by distribution-independent $c(h)$
  - Distribution-specific characterization of implicit bias
  - Or: direct analysis of generalization properties



What fits our understanding:

- Can get generalization even if can fit random labels [we're controlling some other complexity measure]
- Can get implicit regularization (seek small "norm") from optimization algorithm, even if not explicit
- **Generalization becomes better as size increases**

$y = \langle w^*, \phi_\infty(x) \rangle$ ($\|\phi_\infty(x)\|$ bounded)

$\phi_d(x)$ = random projection of $\phi_\infty(x)$

e.g. $\langle \phi_\infty(x), \phi_\infty(x') \rangle = e^{-\|x-x'\|^2}$

and $\phi_d(x)[i] = \frac{1}{\sqrt{d}} \cos(\langle \omega_i, x \rangle + \theta_i)$

$A(S) = \arg\min \|w\| \ \ s.t. L_S(x \mapsto \langle w, \phi_d(x) \rangle) = 0$

i.e. $\forall_{(x_i,y_i) \in S} y_i = \langle w, \phi_d(x_i) \rangle$

A similar example:

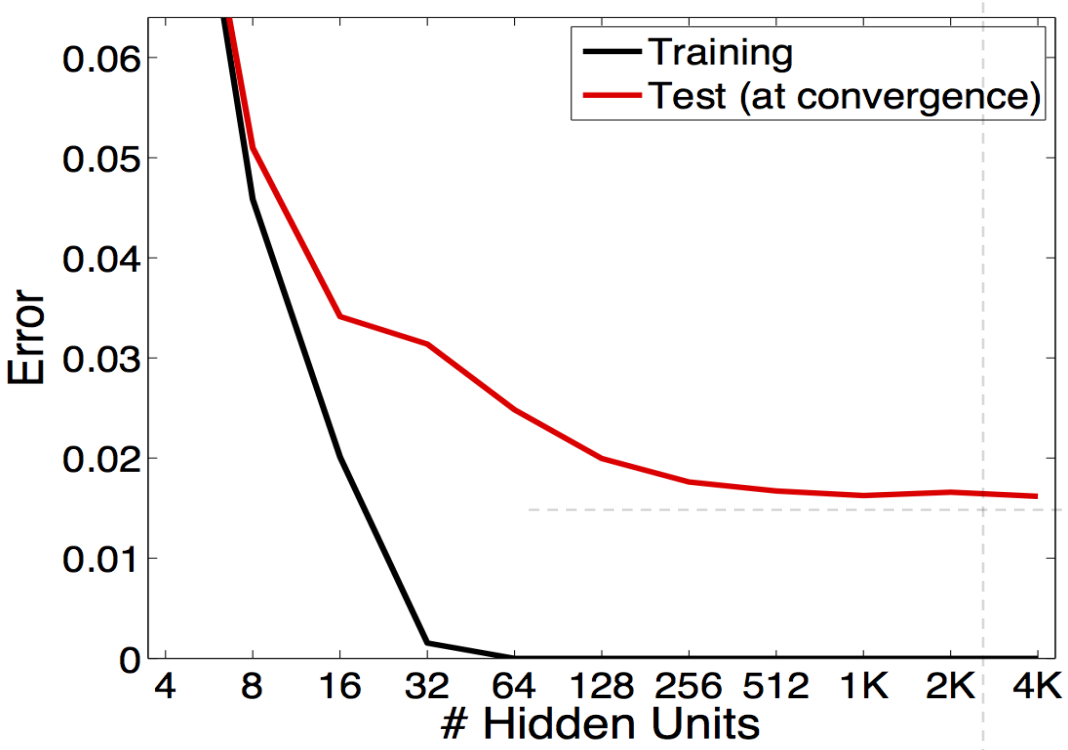
Matrix completion using a rank-*d* factorization:

$L(X) = \|X - A\|_2^2$, $\hat{L}$ based on nk observed entries

$X = UV^\top$, $U, V \in \mathbb{R}^{n \times d}$ ➔ $rank(X) \leq d$

If $d < k$: $\arg\min \hat{L}(X)$ s.t. $rank(X) \leq d$

If $d > k$: $\arg\min \|X\|_*$ $s.t.$ $\hat{L}(X) = 0, rank(X) \leq d$



What fits our understanding:

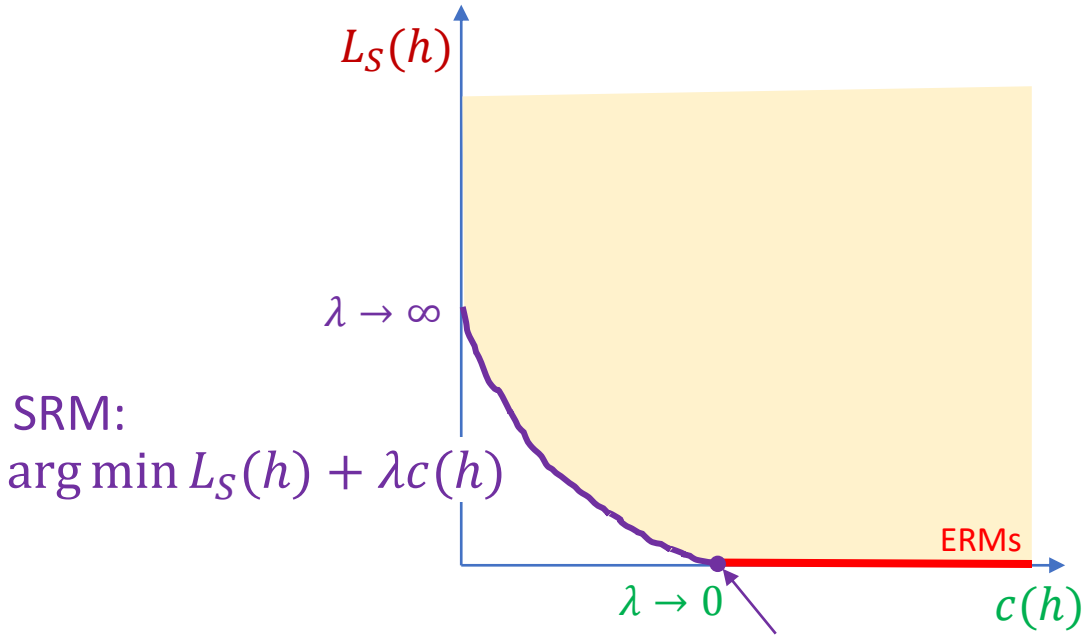
- Can get generalization even if can fit random labels [we're controlling some other complexity measure]
- Can get implicit regularization (seek small "norm") from optimization algorithm, even if not explicit
- **Generalization becomes better as size increases**

What doesn't fit:

- Even when the approximation error>0 (with noise), we get good generalization with $L_S(h) = 0$

Table 1: The training and test accuracy (in percentage) of various models on the CIFAR10 dataset.

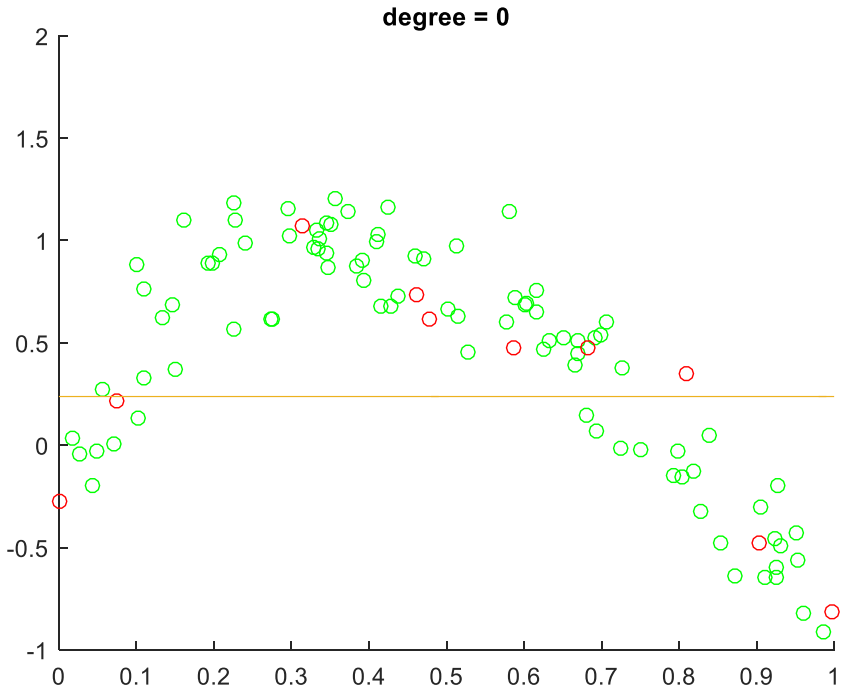
model	# params	random crop	weight decay	train accuracy	test accuracy
Inception	1,649,402	yes	yes	100.0	89.05
		yes	no	100.0	89.31
		no	yes	100.0	86.03
		no	no	100.0	85.75
MLP 3x512	1,735,178	no	yes	100.0	53.35
		no	no	100.0	52.39
(fitting random labels)		no	no	100.0	10.48



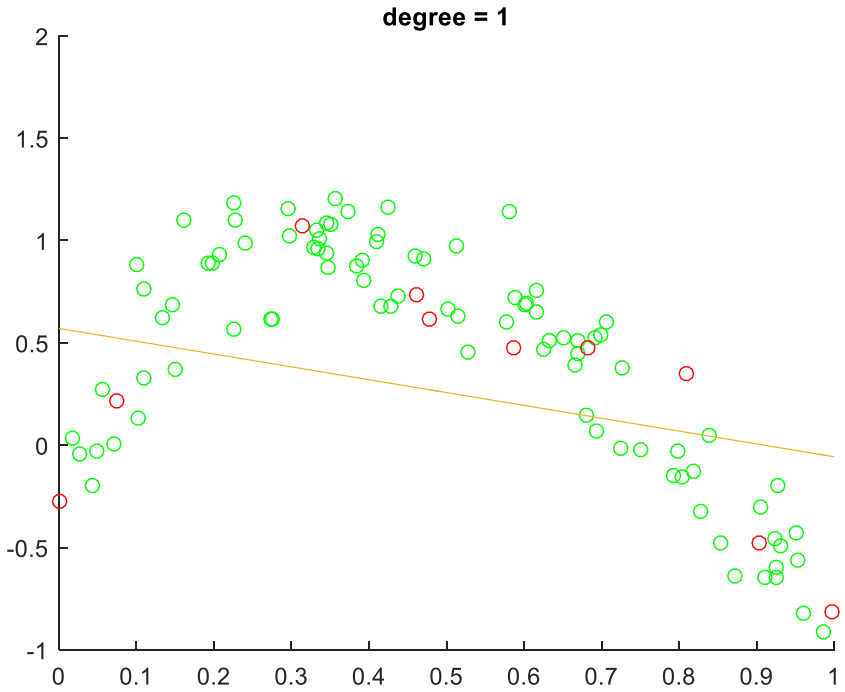
SRM:
$\arg\min L_S(h) + \lambda c(h)$

MDL:
$\arg\min c(h)\ \ s.t. L_S(h) = 0$

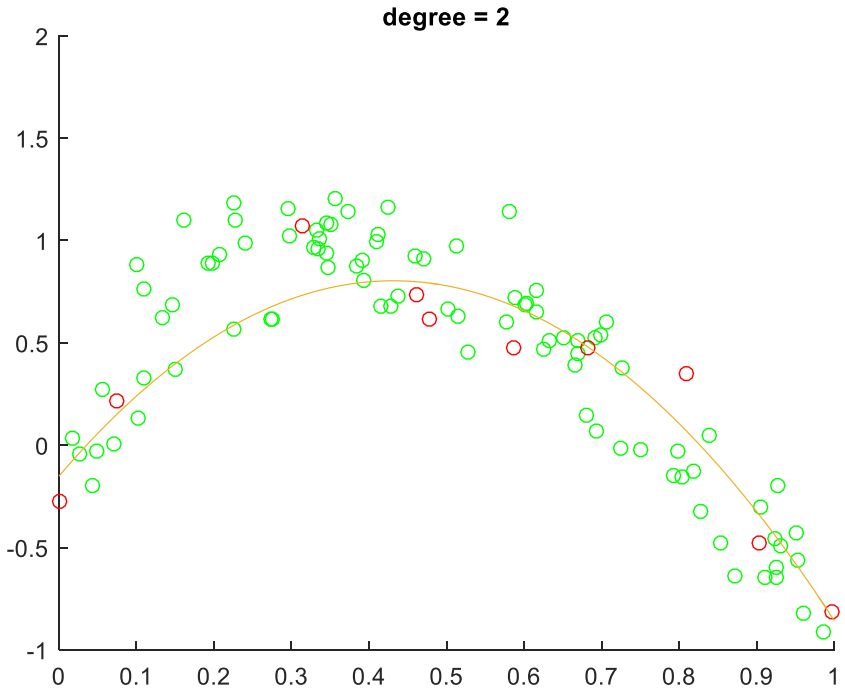
Intro to Machine Learning, Lecture 2



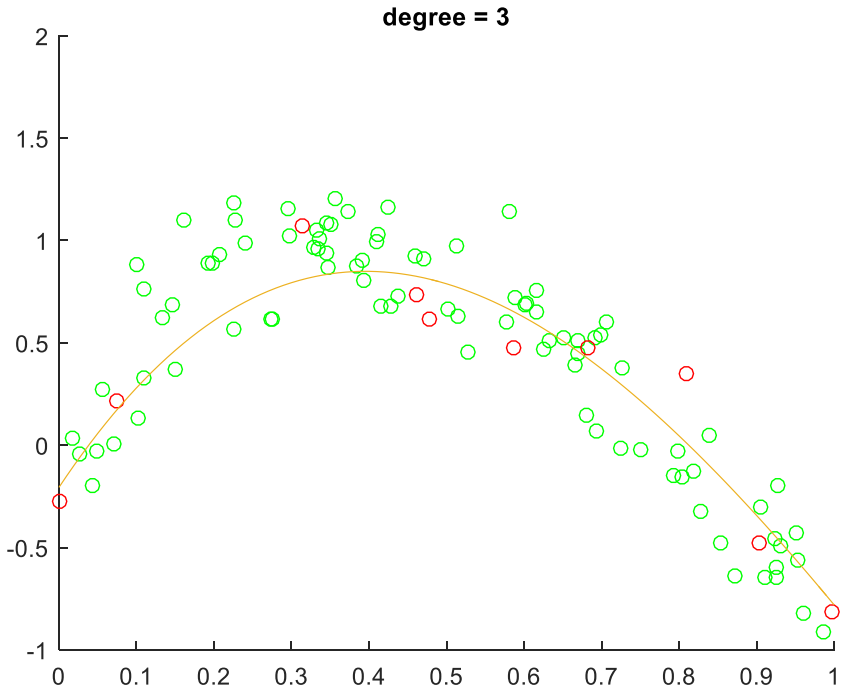
Intro to Machine Learning, Lecture 2



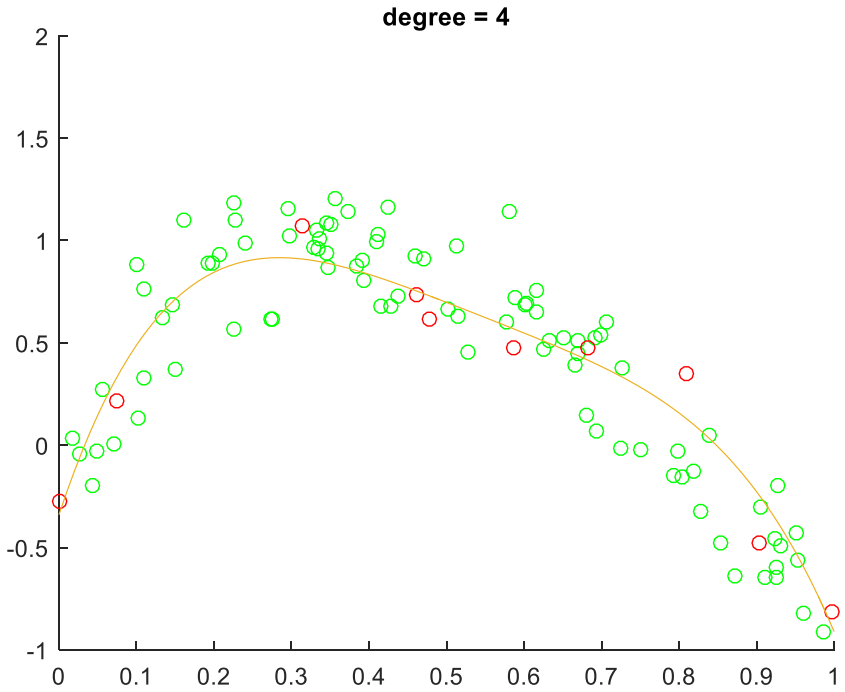
Intro to Machine Learning, Lecture 2



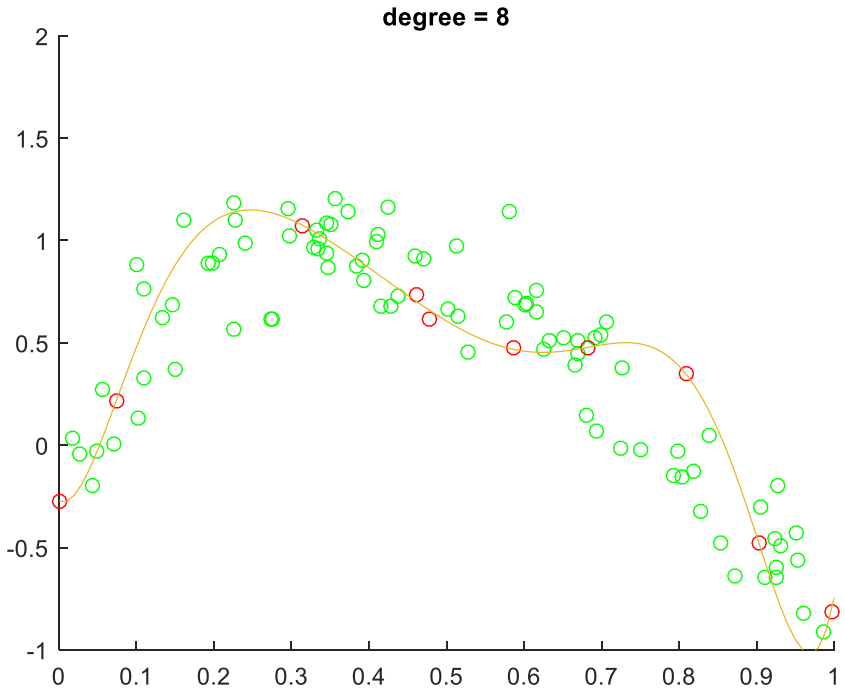
Intro to Machine Learning, Lecture 2



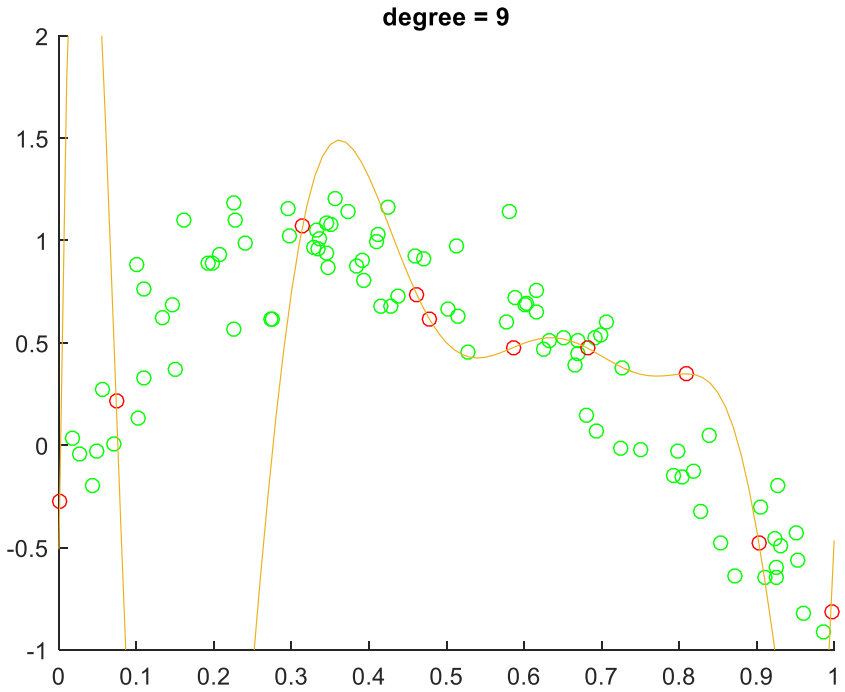
Intro to Machine Learning, Lecture 2

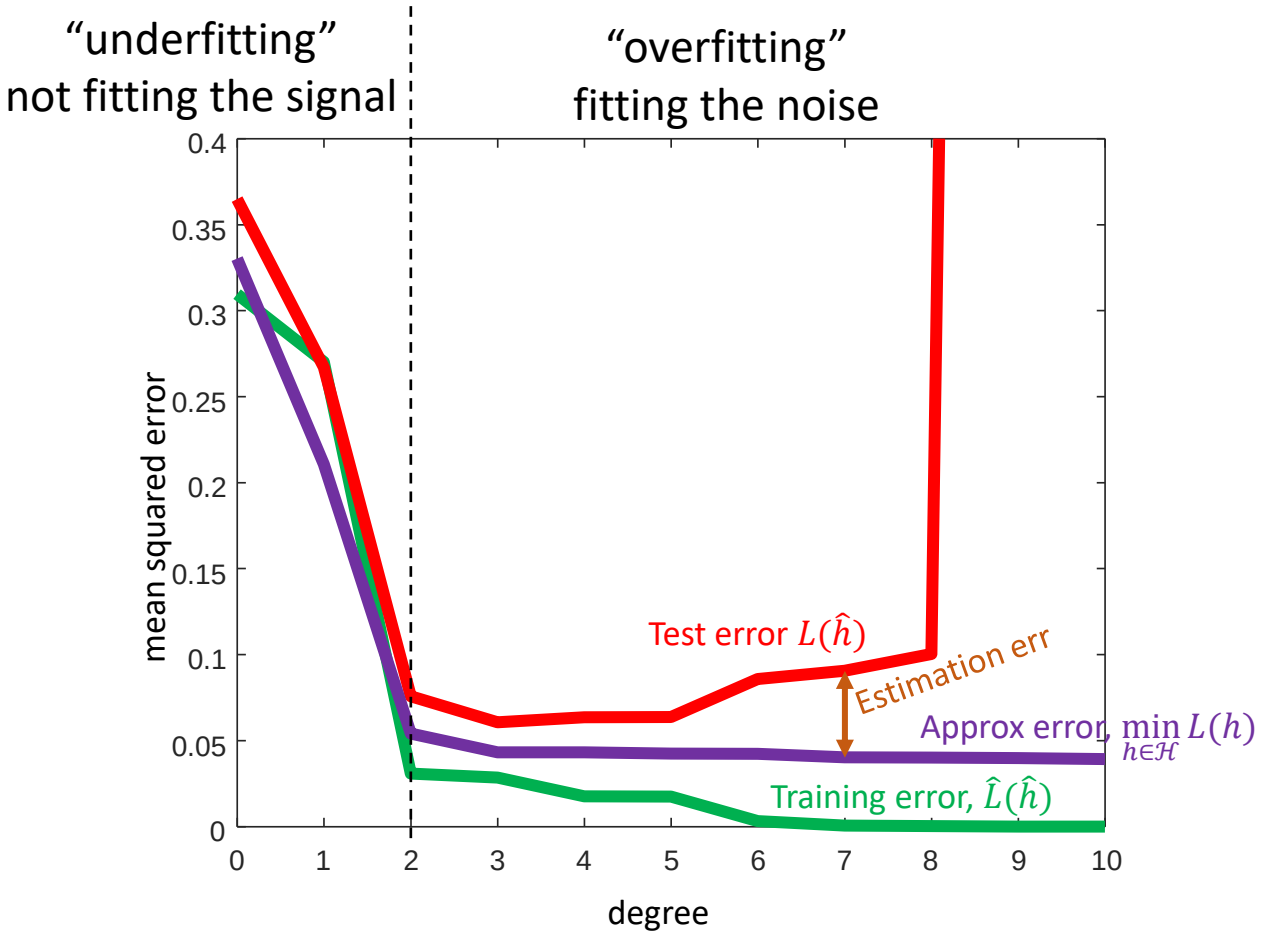


Intro to Machine Learning, Lecture 2



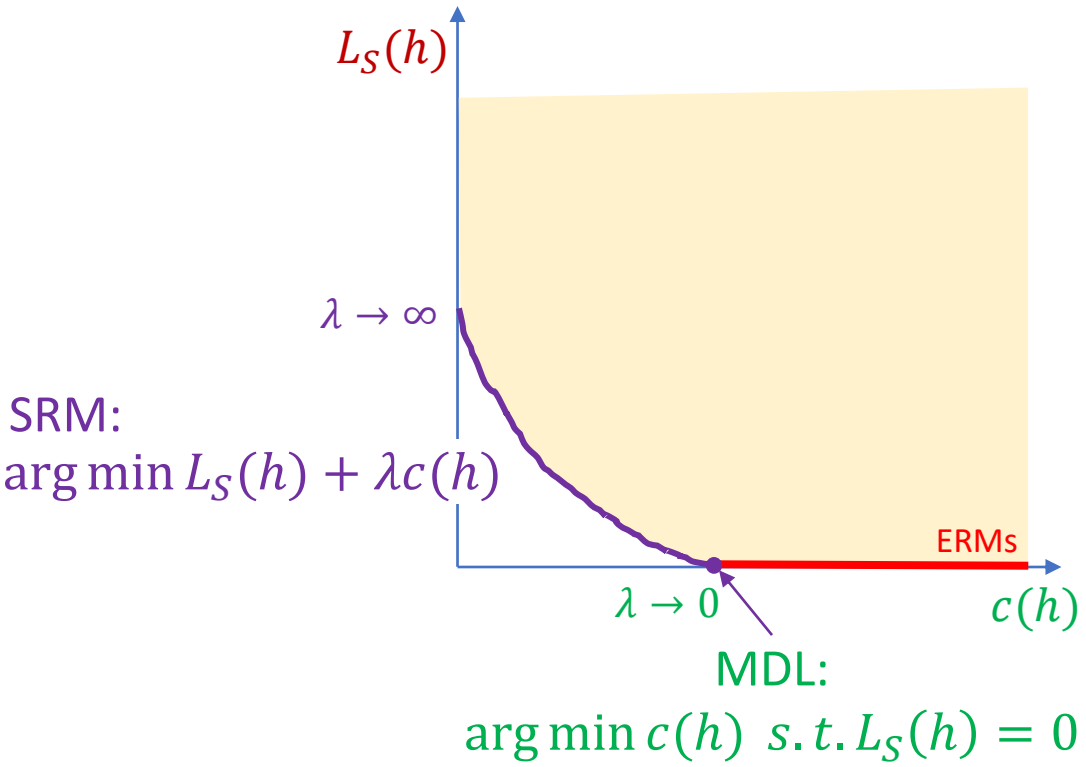
Intro to Machine Learning, Lecture 2

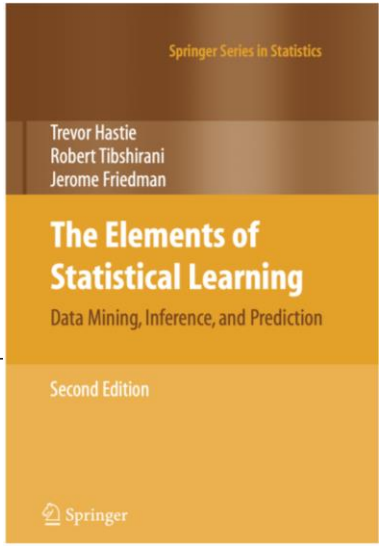


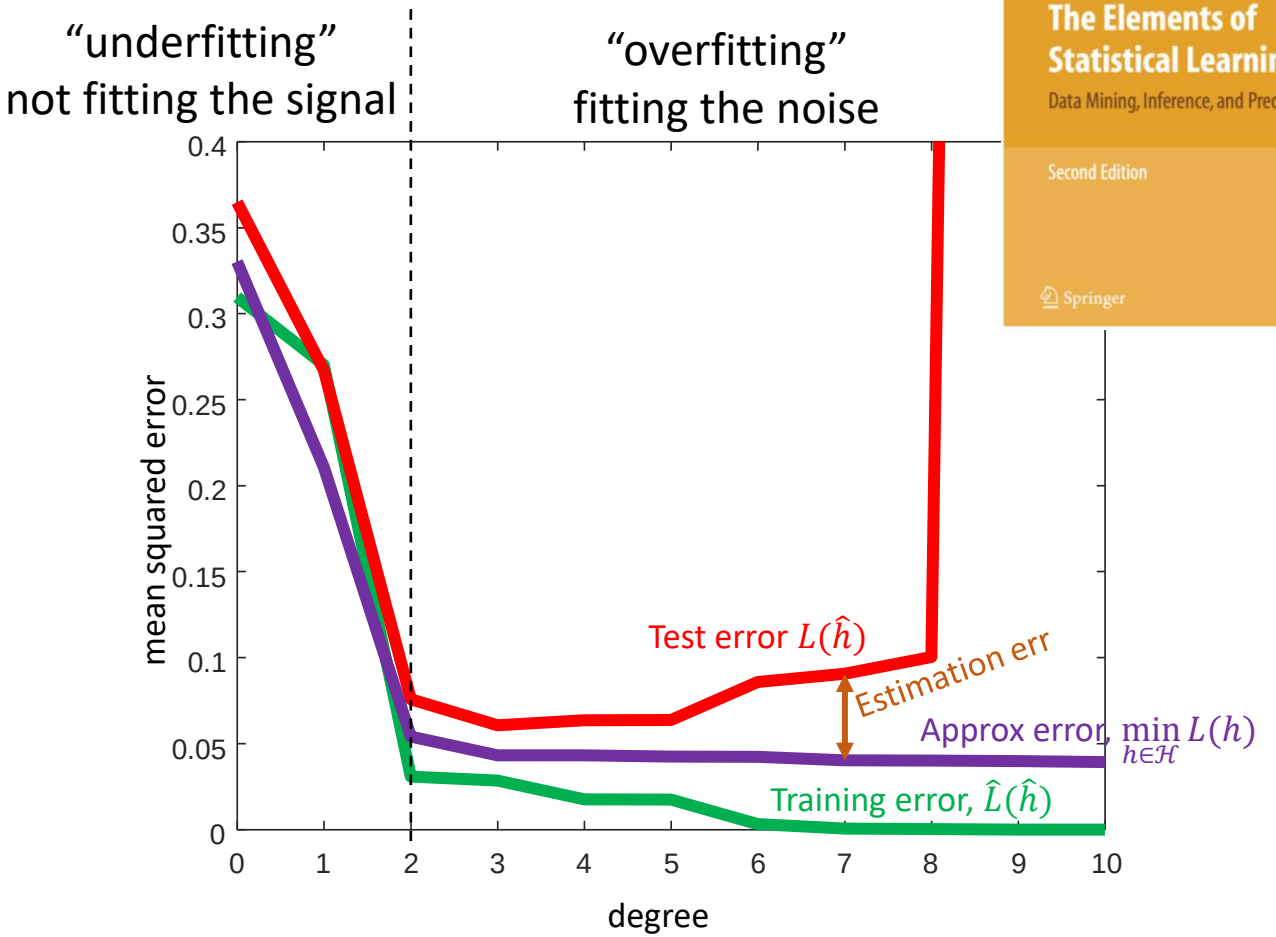


$$L(\hat{h}) \leq \underbrace{\inf_{h\in\mathcal{H}} L(h)}_{\text{approximation error}} + \underbrace{\sqrt{\frac{\log|\mathcal{H}| + 2\log 2/\delta}{n}}}_{\text{estimation error}}$$

“a model with zero training error is overfitting […] and will typically generalize poorly”



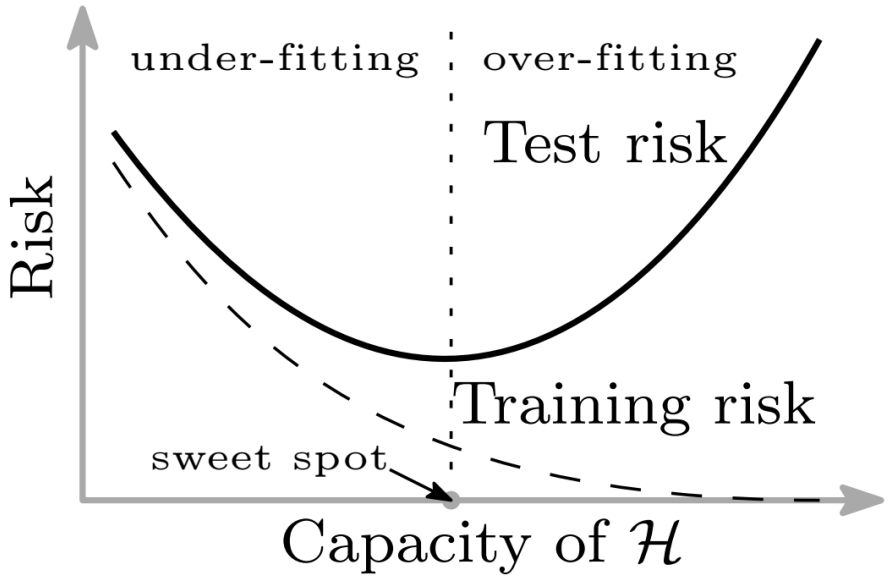


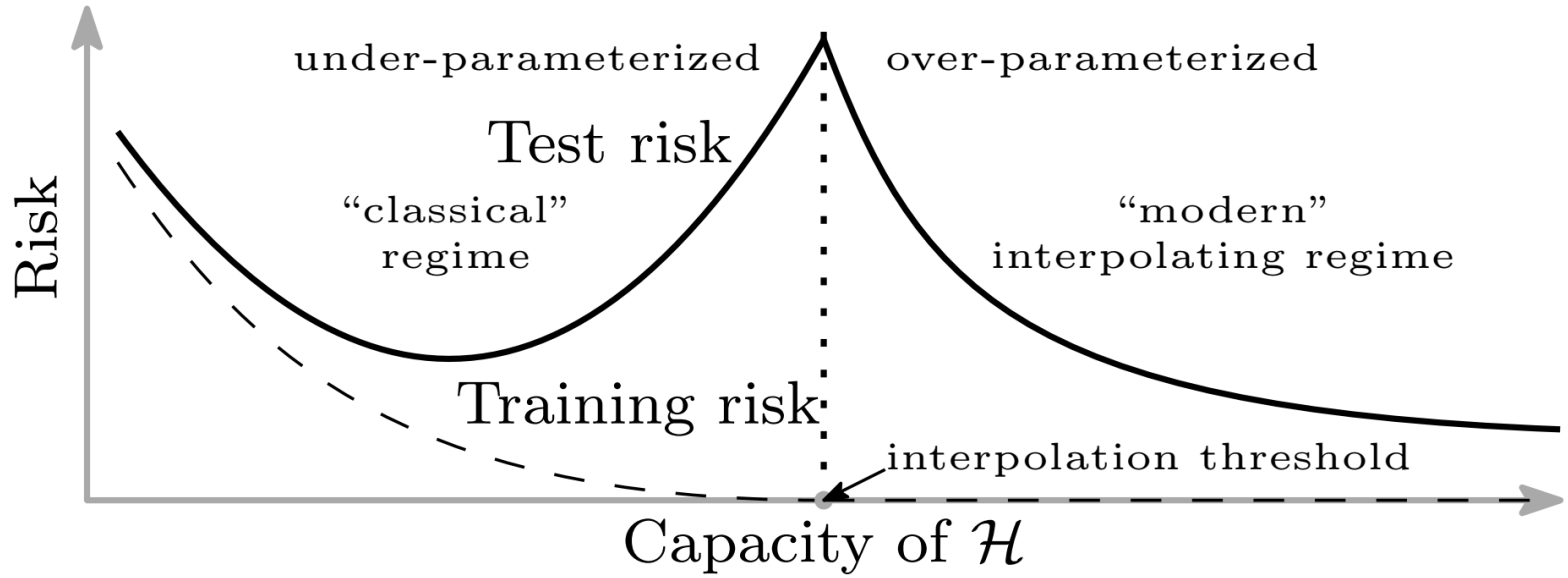


$$L(\hat{h}) \le \underbrace{\inf_{h\in\mathcal{H}} L(h)}_{\text{approximation error}} + \underbrace{\sqrt{\frac{\log|\mathcal{H}| + 2\log^{2}/_{\delta}}{n}}}_{\text{estimation error}}$$

Reconciling modern machine-learning practice and the classical bias–variance trade-off

Mikhail Belkin[a,b,1], Daniel Hsu[c], Siyuan Ma[a], and Soumik Mandal[a]





$L(w) = \mathbb{E}[(\langle w, \phi_d(x)\rangle - y)^2]$ $\quad \hat{L}(w) = \frac{1}{n}\sum_i(\langle w, \phi_d(x_i)\rangle - y_i)^2$

$\phi_d(x) \in \mathbb{R}^d$

~~$\hat{w} = \arg\min \hat{L}(w)$~~

$\hat{w}$ = GD on $\hat{L}(w)$

```
>> w = PhiX \ y
```

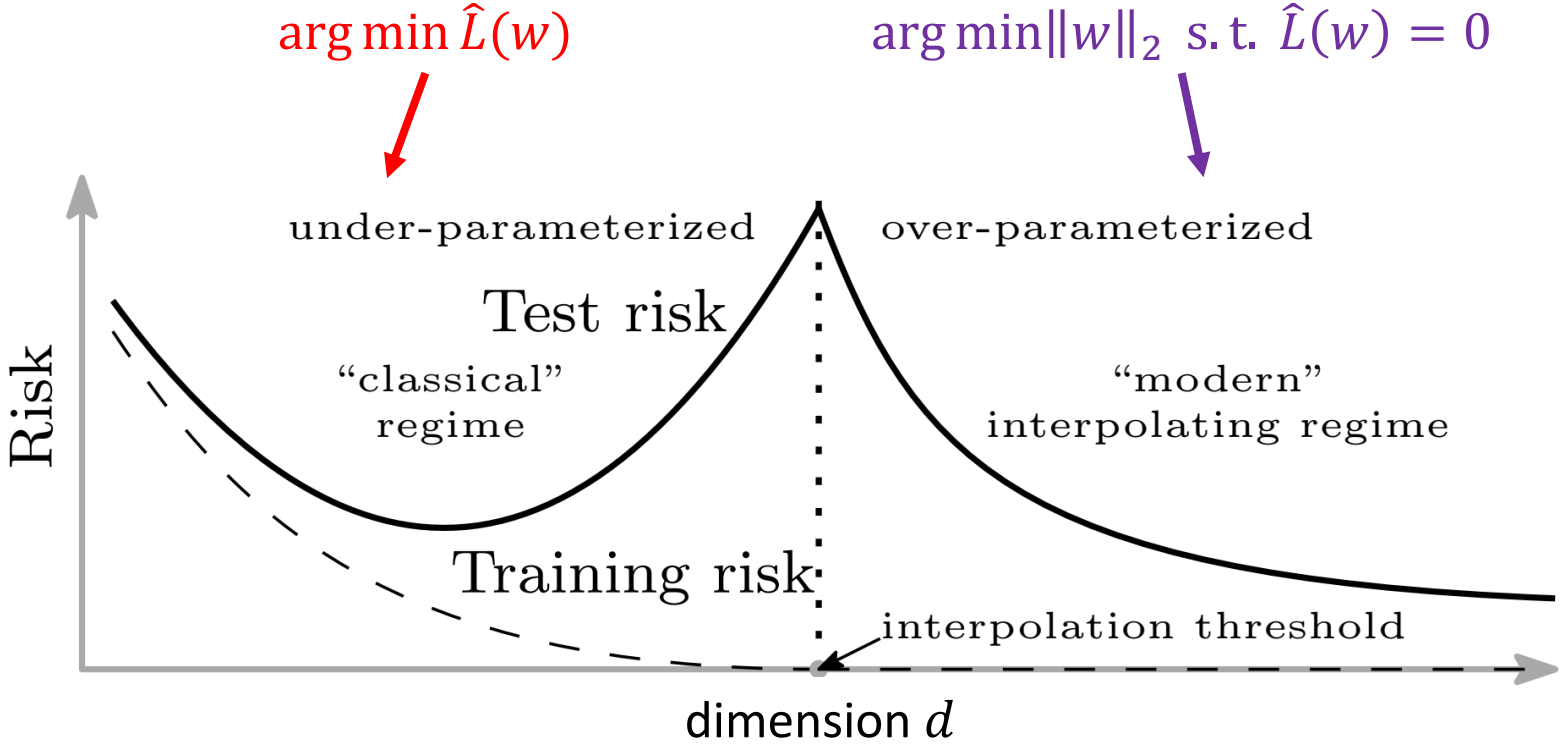
```
>>> w = np.linalg.lstsq(PhiX,y)[0]
```

$x = \langle w^*, \phi_\infty(x)\rangle \quad (\|\phi_\infty(x)\| \le 1)$

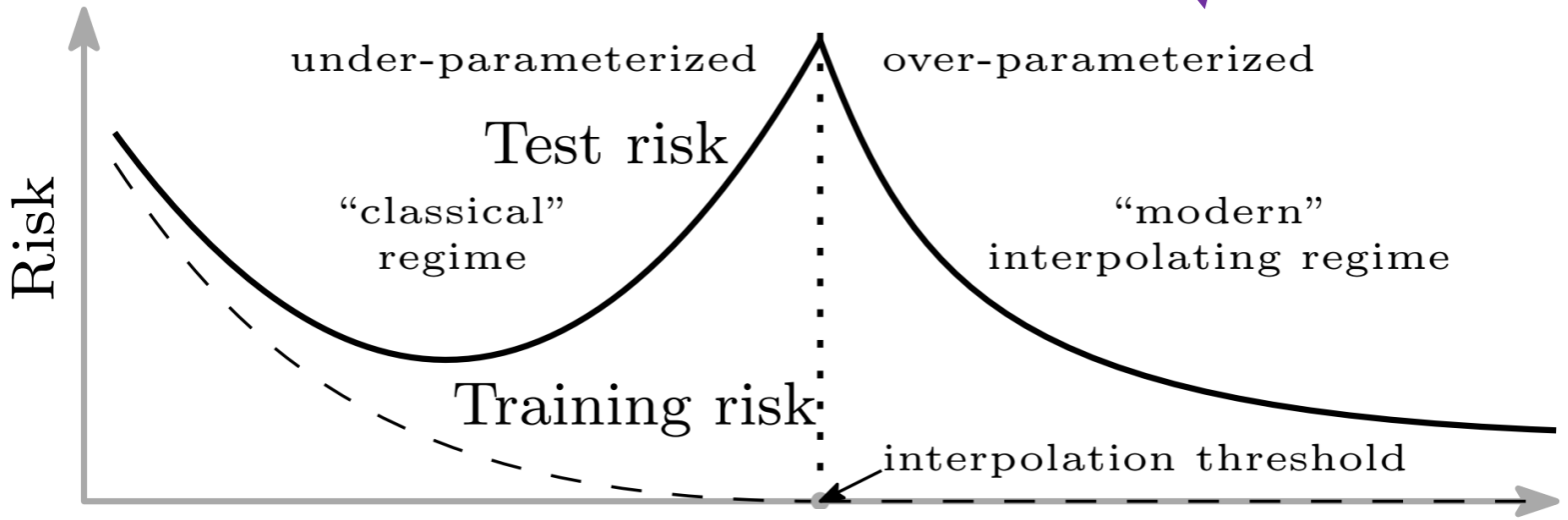
$\phi_d(x)$ = random projection of $\phi_\infty(x)$

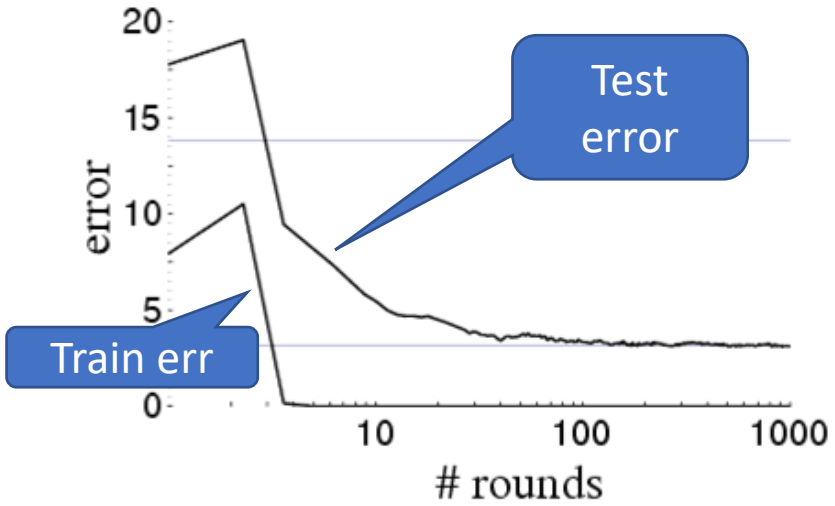
e.g. $\langle \phi_\infty(x), \phi_\infty(x')\rangle = e^{-\|x-x'\|^2}$

and $\phi_d(x)[i] = \frac{1}{\sqrt{d}}\cos(\langle \omega_i, x\rangle + \theta_i)$

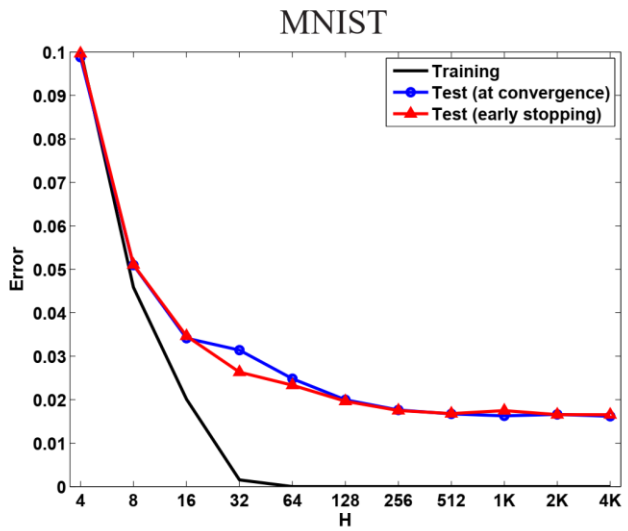


$\arg\min \|w\|_2 \text{ s.t. } \hat{L}(w) = 0$





[Bartlett et al "Boosting the Margin" 1998]



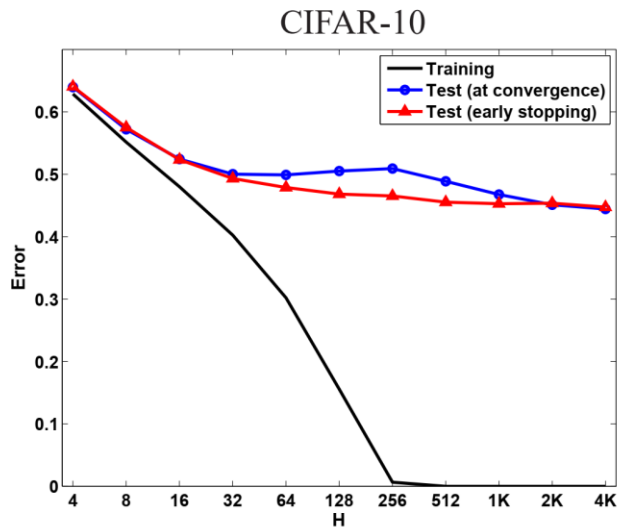
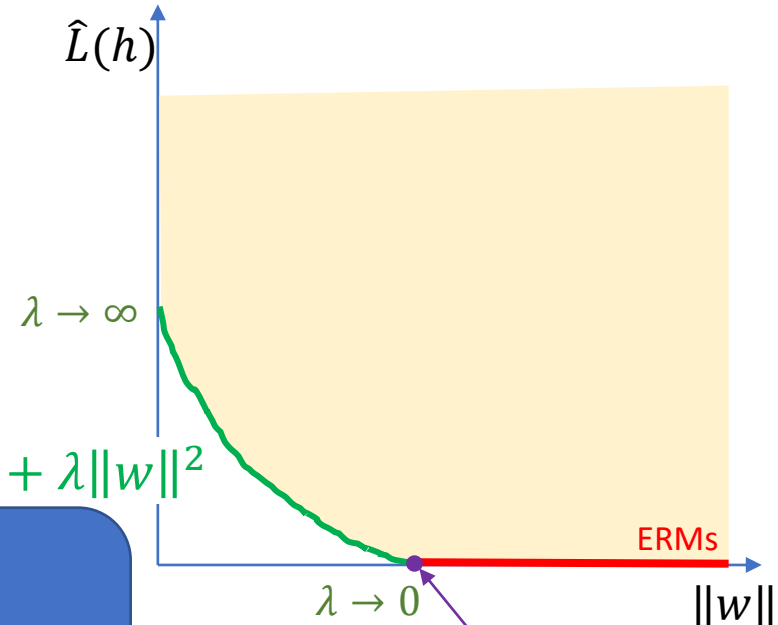


Table 1: The training and test accuracy (in percentage) of various models on the CIFAR10 dataset.

model	# params	random crop	weight decay	train accuracy	test accuracy
Inception	1,649,402	yes	yes	100.0	89.05
		yes	no	100.0	89.31
		no	yes	100.0	86.03
		no	no	100.0	85.75

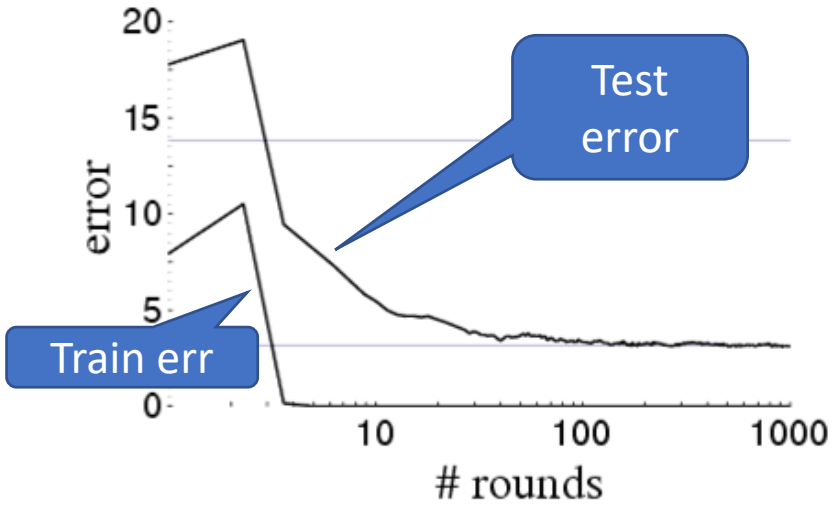
[Zhang et al "Rethinking generalization" ICLR 2017]



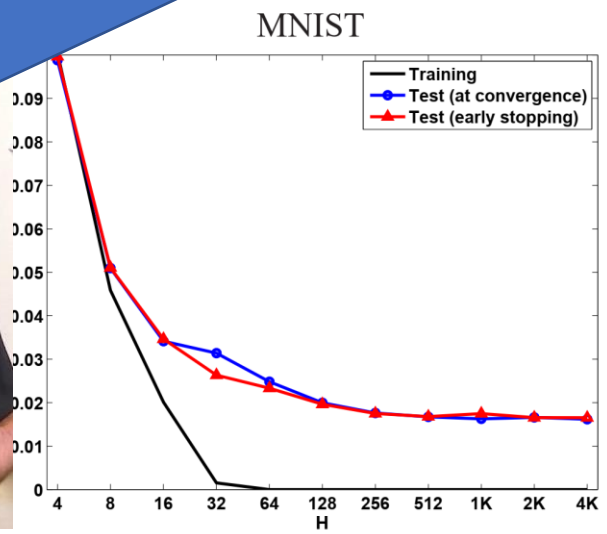
SRM:
$\arg\min \hat{L}(w) + \lambda\|w\|^2$

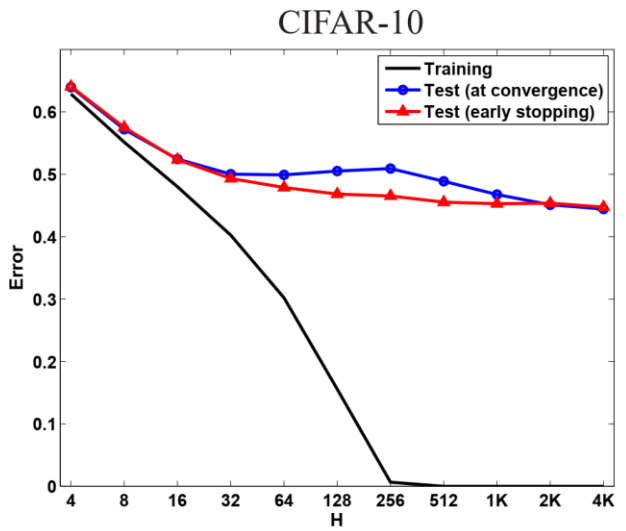
MDL:
$\arg\min \|w\| \ s.t. \hat{L}(w) = 0$

We can learn with MDL ($\hat{L}(w) = 0$, "**interpolation learning**") in many settings where $L(w^*) \gg 0$, eg noisy settings where $y = h_{w^*}(x) + noise$.
Often, **overfitting (fitting the noise) is benign**, and not as harmful as theory tells us.
-Misha Belkin, 2018



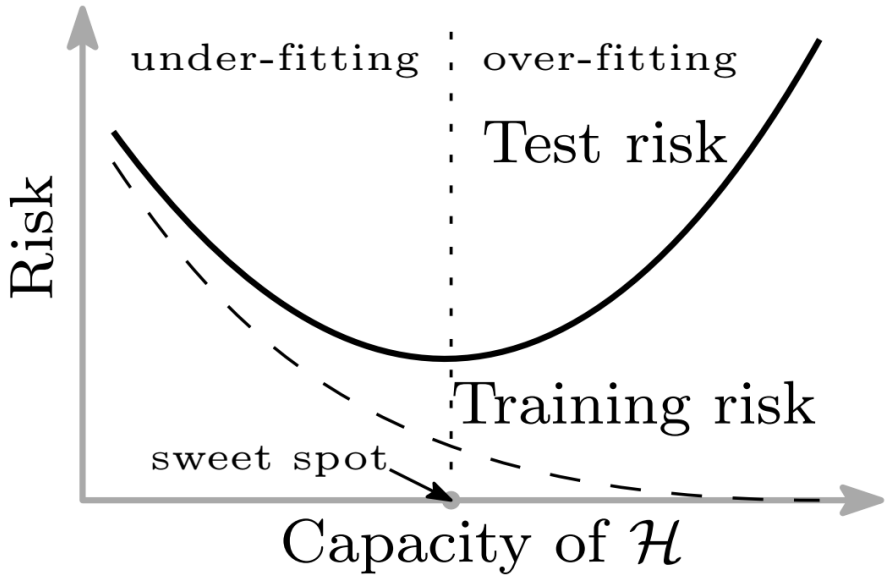
[Bartlett et al "Boosting the Margin" 1998]

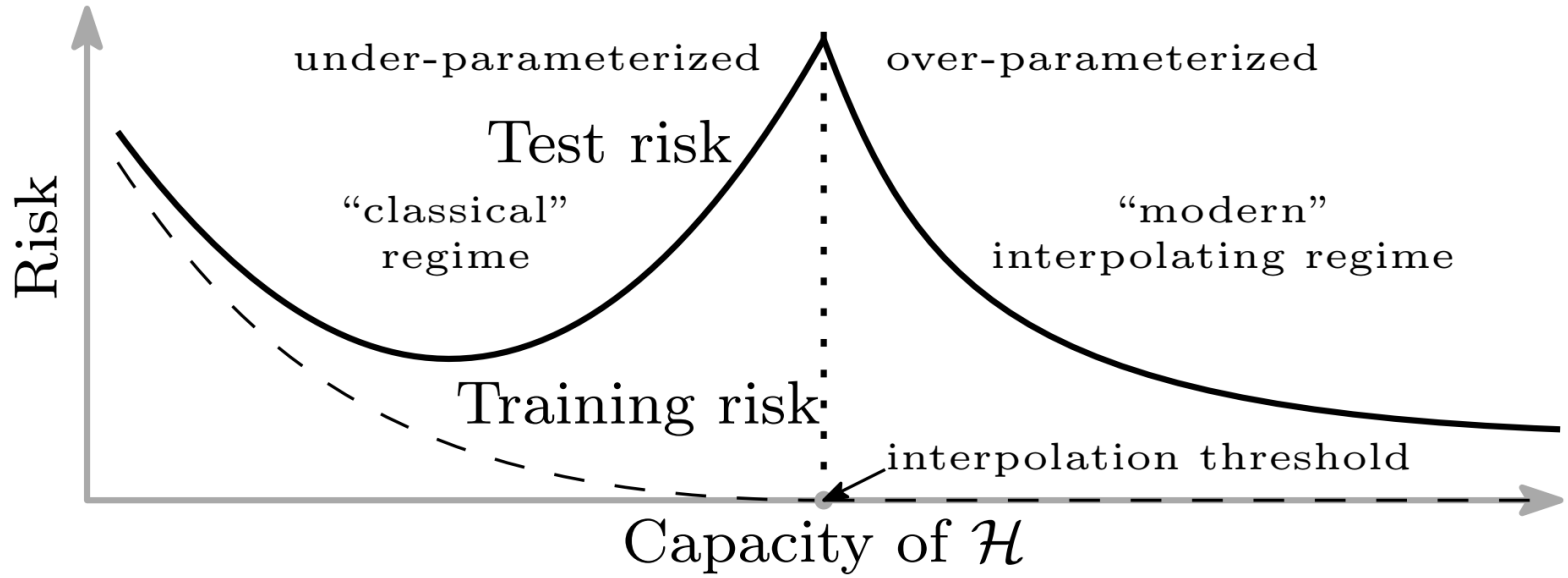




Reconciling modern machine-learning practice and the classical bias–variance trade-off

Mikhail Belkin[a,b,1], Daniel Hsu[c], Siyuan Ma[a], and Soumik Mandal[a]





$L(w) = \mathbb{E}[(\langle w, \phi_d(x)\rangle - y)^2]$ $\quad \hat{L}(w) = \frac{1}{n}\sum_i(\langle w, \phi_d(x_i)\rangle - y_i)^2$

$\phi_d(x) \in \mathbb{R}^d$

~~$\hat{w} = \arg\min \hat{L}(w)$~~

$\hat{w}$ = GD on $\hat{L}(w)$

```
>> w = PhiX \ y
```

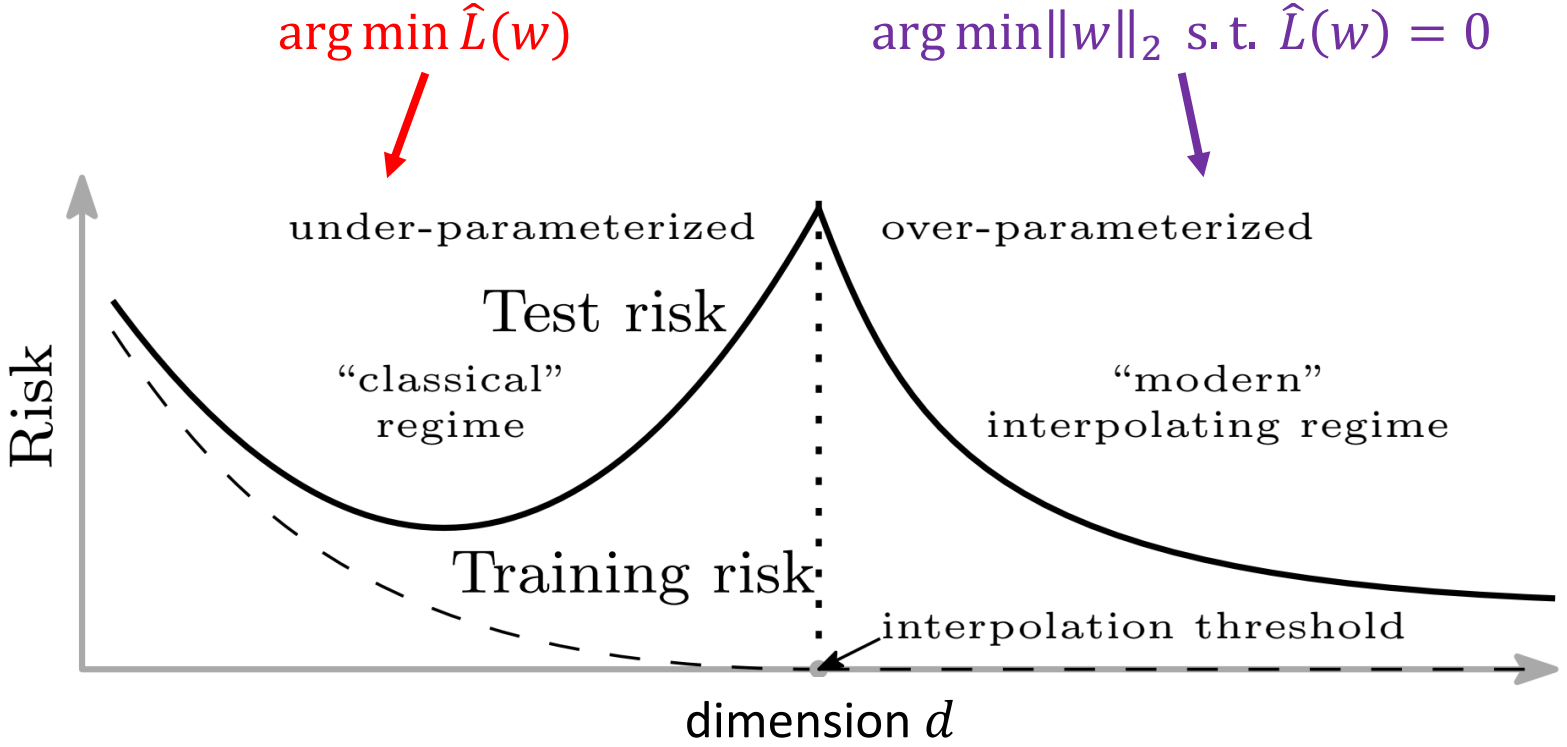
```
>>> w = np.linalg.lstsq(PhiX,y)[0]
```

$x = \langle w^*, \phi_\infty(x)\rangle \quad (\|\phi_\infty(x)\| \le 1)$

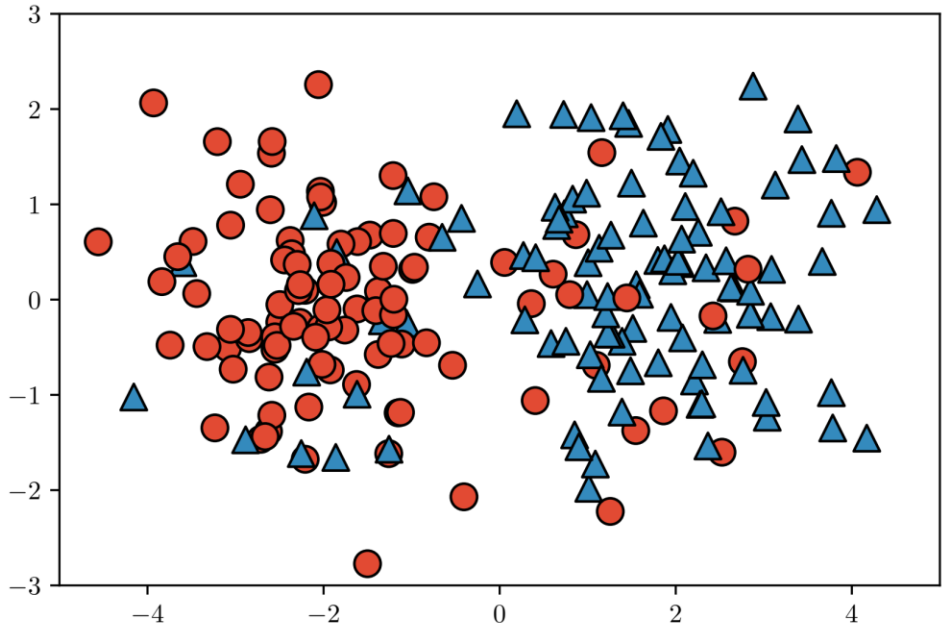
$\phi_d(x)$ = random projection of $\phi_\infty(x)$

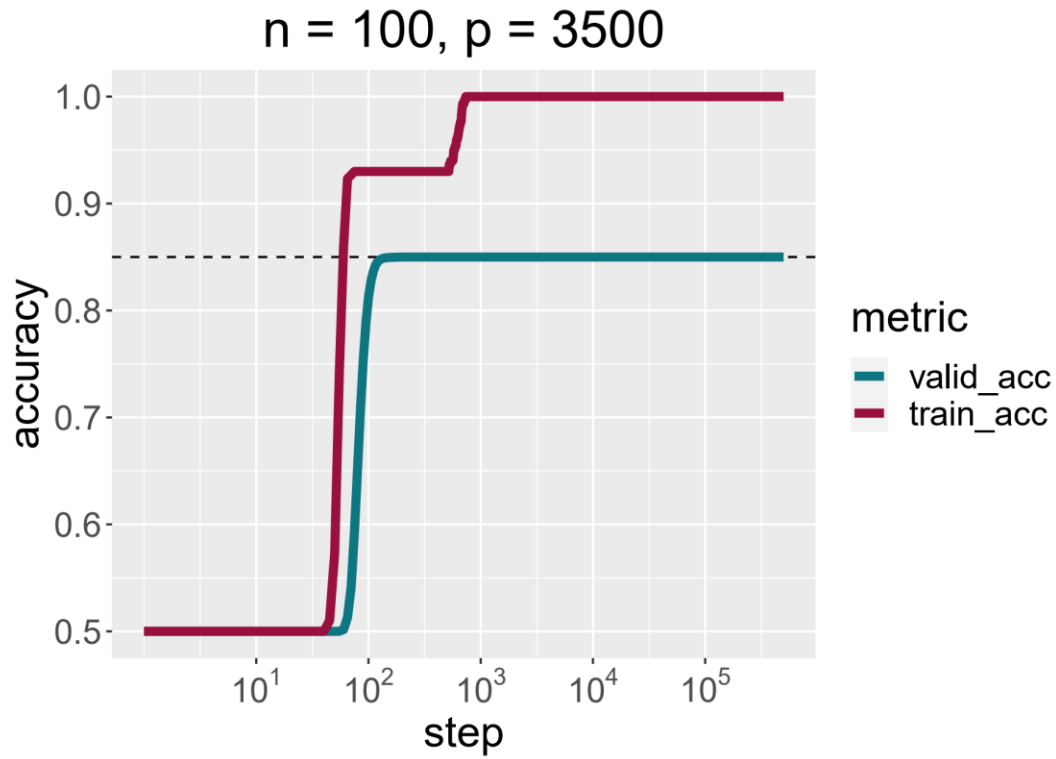
e.g. $\langle \phi_\infty(x), \phi_\infty(x')\rangle = e^{-\|x-x'\|^2}$

and $\phi_d(x)[i] = \frac{1}{\sqrt{d}}\cos(\langle \omega_i, x\rangle + \theta_i)$



Consider the following experiment: you have Gaussian mixture model data, but you have a 'noisy' distribution where every sample has 15% chance of having a random label. Then train an overparameterized two-layer network by gradient descent on this (noisy) data.

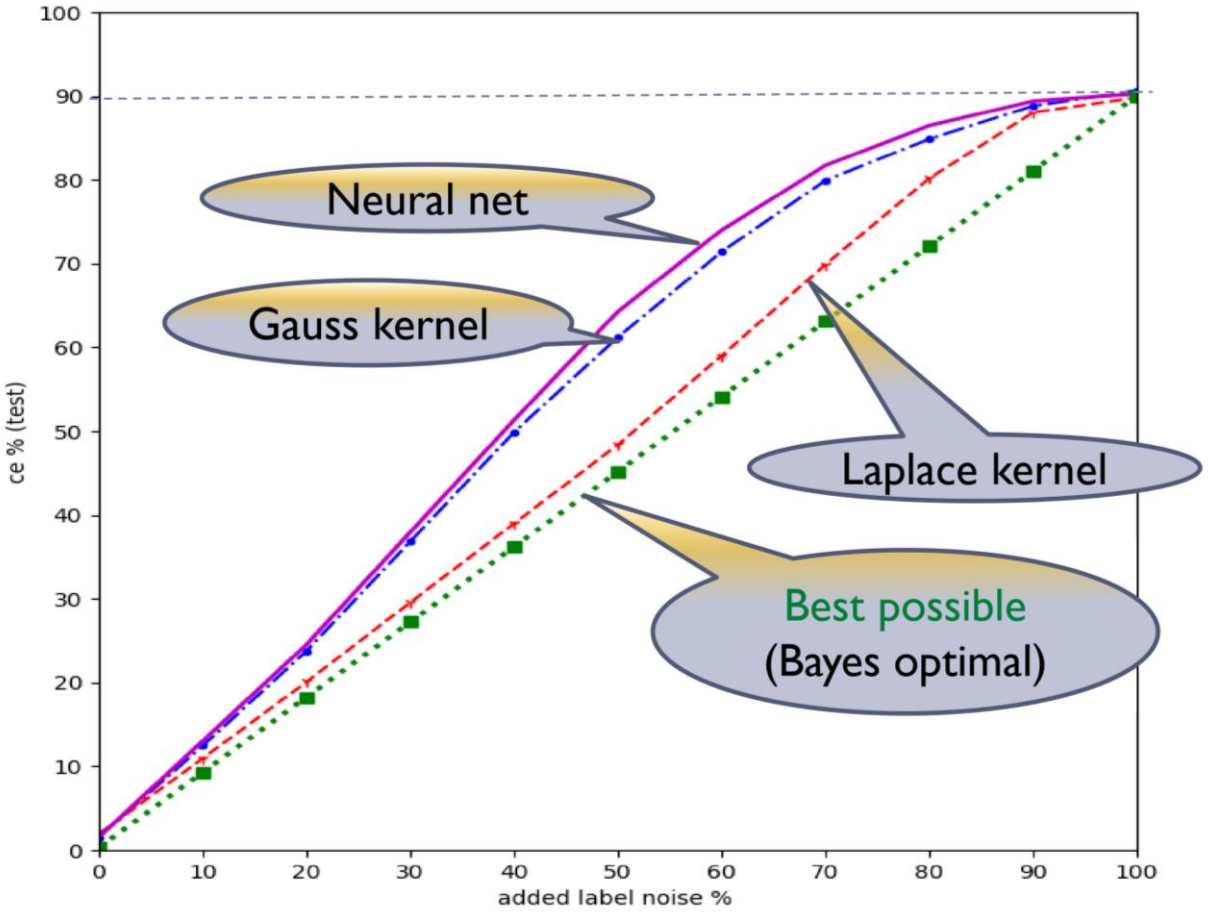




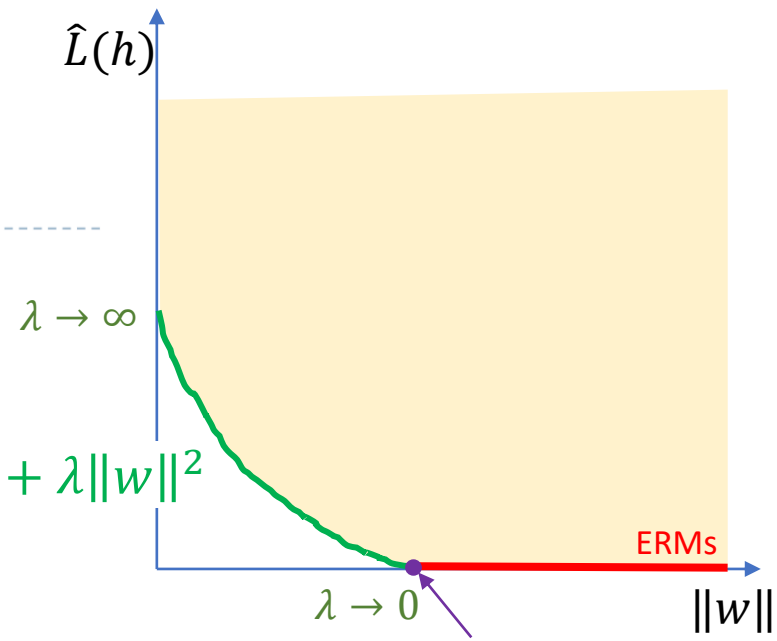
The neural network achieves 100% training accuracy and *simultaneously* optimal test accuracy (85%)—provably so [F.–Chatterji–Bartlett'22].

Interpolation does not overfit even for very noisy data

All methods (except Bayes optimal) have zero training *square* loss.



[Belkin Ma Mandal, ICML 18]

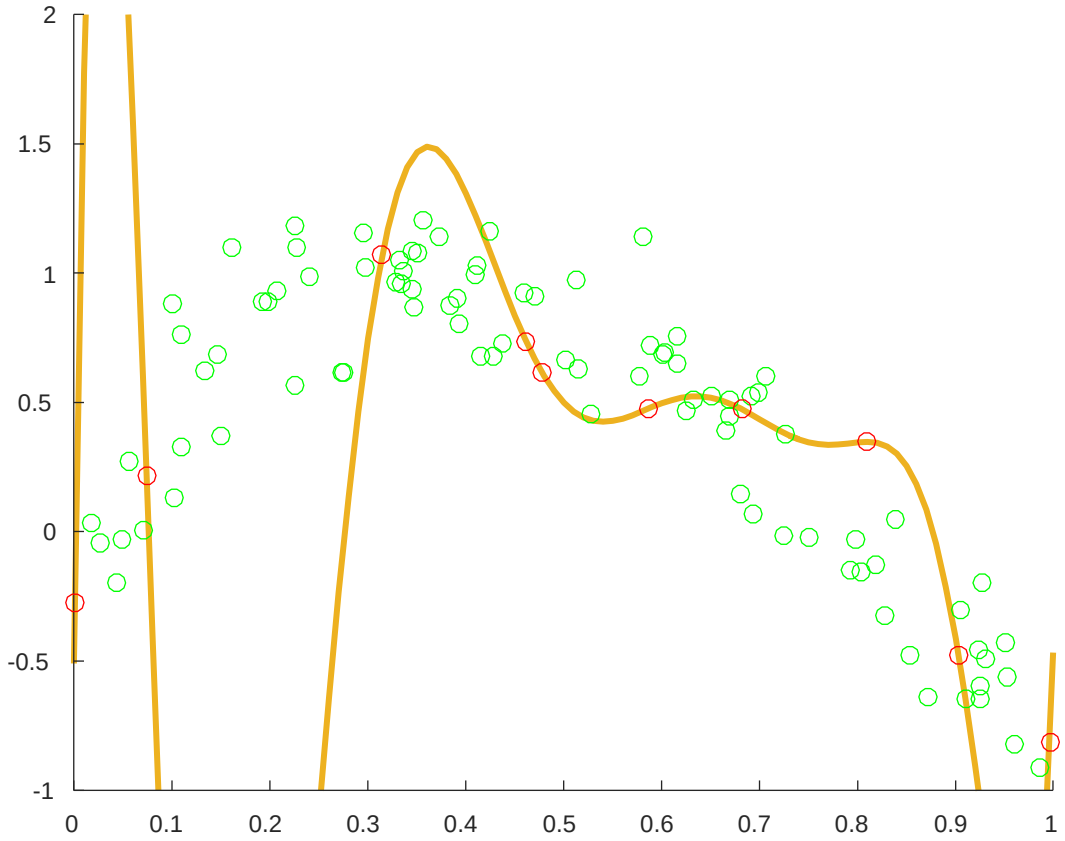


SRM:
$\arg\min \hat{L}(w) + \lambda\|w\|^2$

MDL:
$\arg\min \|w\| \ \ s.t. \hat{L}(w) = 0$

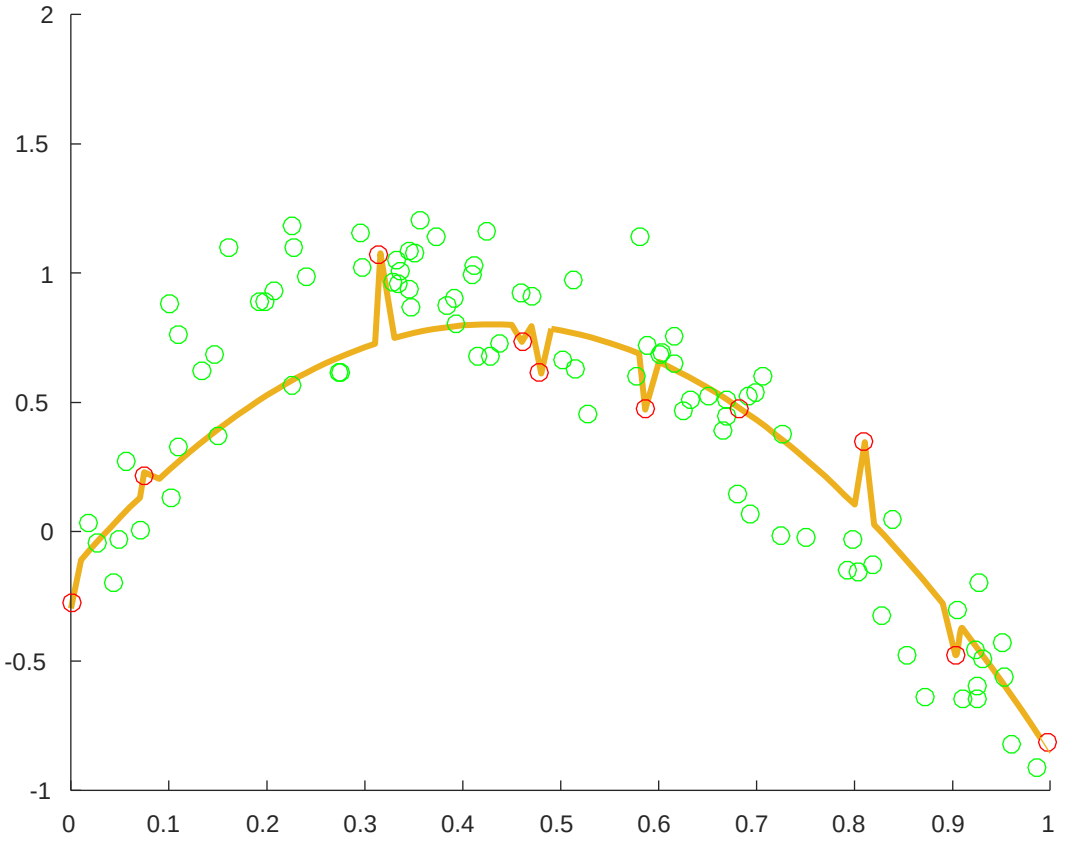
Harmful Overfitting

(fitting noise has large effect everywhere, overwhelms signal fit)



Benign Overfitting

(fitting noise has measure ≈0 effect)



$$y = \langle w^*, x \rangle + \mathcal{N}(0, \sigma^2)$$

$$w^*, x \in \mathbb{R}^d \qquad d = d_S + d_J$$

$$w^* = \left(w_S^* \ , 0_{d_J}\right)$$

d_s	$d_J \to \infty$
w_S^*	0
signal x_s	junk x_j

$$x = \left(x_S, x_J\right) \quad x_S \sim \mathcal{N}\left(0, I_{d_S}\right) \quad x_J \sim \mathcal{N}\left(0, \eta I_{d_J}\right)$$

$$d_J \to \infty, \ \ \eta = \frac{\lambda}{d_J} \rightarrow \left\|x_J\right\|^2 = \lambda$$

$$\widehat{\boldsymbol{w}}_{\boldsymbol{MN}} = \mathbf{arg}\ \mathbf{min}_{\boldsymbol{Xw=Y}} \|\boldsymbol{w}\|_{\mathbf{2}}$$

Proof: Gram matrix is
$XX^\top = \left(X_S X_S^\top + X_J X_J^\top\right) \to \left(X_S X_S^\top + \lambda I\right)$

Equivalent to Ridge Regression: $\langle \widehat{w}_{MN}, x \rangle \xrightarrow{d_J \to \infty} \langle \widehat{w}_\lambda, x_S \rangle$

$$\widehat{w}_\lambda = \arg \min_{w_S \in \mathbb{R}^{d_S}} \|Y - X_S w_S\|^2 + \lambda \|w_S\|^2$$

For $\boldsymbol{\lambda = o(n)}$ and $\boldsymbol{d_J \to \infty}$, $L(\widehat{w}_{MN}) = L(\widehat{w}_\lambda) \xrightarrow{n \to \infty} L(w^*) = \sigma^2$

$L(w) = \mathbb{E}[(\langle w, x \rangle - y)^2]$

Goal: consistency in a noisy (non-realizable) setting

$$L(\widehat{w}_n) \xrightarrow{n\to\infty} L(w^*) = \sigma^2 > 0$$

For a balanced predictor, $\arg\min \hat{L}(\widehat{w}_n), \|\widehat{w}_n\|$, e.g. with $\hat{L}(\widehat{w}_n) = \sigma^2$ and $\|\widehat{w}_n\|$ small:

ensuring $\widehat{w}_n \in \mathcal{W}_n$
e.g. $\|\widehat{w}_n\| \le B$ and $\mathcal{W}_n = \{\|w\| \le B\}$

e.g. $O\left(\sqrt{\|\widehat{w}_n\|^2/n}\right)$

$$L(\widehat{w}_n) \le \hat{L}(\widehat{w}_n) + \sup_{w\in\mathcal{W}_n} \left(L(w) - \hat{L}(w)\right)$$

$L(w^*) = \sigma^2$

0

Goal: consistency in a noisy (non-realizable) setting

$$L(\widehat{w}_n) \xrightarrow{n\to\infty} L(w^*) = \sigma^2 > 0$$

For an interpolating predictor:

ensuring $\widehat{w}_n \in \mathcal{W}_n$

e.g. $\|\widehat{w}_n\| \le B$ and $\mathcal{W}_n = \{\|w\| \le B\}$

e.g. $O\left(\sqrt{\|\widehat{w}_n\|^2/n}\right)$

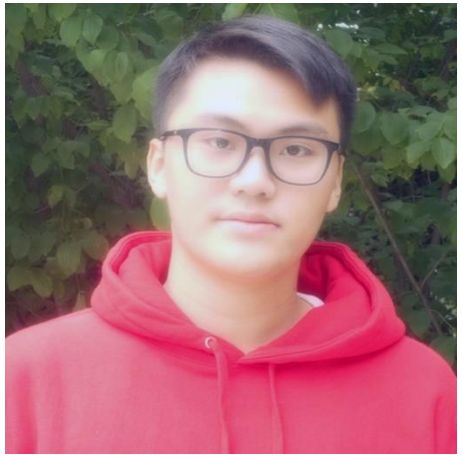
$$L(\widehat{w}_n) \le \hat{L}(\widehat{w}_n) + \sup_{w\in\mathcal{W}_n} \left(L(w) - \hat{L}(w)\right)$$

$= 0$

for an interpolator ???? σ^2

Can Uniform Convergence Explain Benign Overfitting?

with



Lijia Zhou
UChicago

Frederic Koehler
Stanford

Danica Sutherland
TTIC → UBC

Via Uniform Convergence?

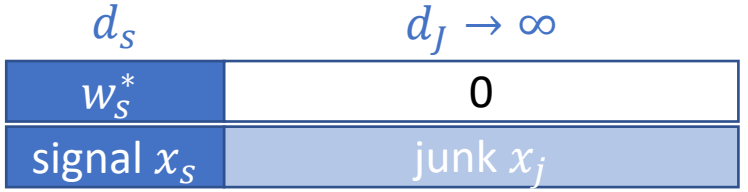
For Lipschitz Loss: $\sup_{\|w\|^2 \le B^2} \left|L(w) - \hat{L}(w)\right| \le 2\, Lip \sqrt{\frac{B^2 \mathbb{E}[\|x\|^2]}{n} + O_p\left(\frac{1}{n}\right)}$

Setting $B = \|\hat{w}\|$, relevant quantity is $\frac{\|\hat{w}\|^2 \mathbb{E}[\|x\|^2]}{n} = \frac{\left(\|w_S^*\|^2 + \frac{\sigma^2}{\lambda_n} n\right)(d_S + \lambda_n)}{n} \to \frac{\frac{\sigma^2}{\lambda_n} n \cdot \lambda_n}{n} = \sigma^2$

$1 \ll \lambda_n \ll n$

Recall junk feature setting: $y = \langle w^*, x \rangle + \xi, \xi \sim \mathcal{N}(0, \sigma^2)$

$w^* = \left(w_S^*\ , 0_{d_J}\right)$



$x = (x_S, x_J) \quad x_S \sim \mathcal{N}(0, I_{d_S}) \quad x_J \sim \mathcal{N}\left(0, \frac{\lambda}{d_J} I_{d_J}\right)$

$\mathbb{E}[\|x\|^2] = d_S + \lambda$

$\hat{w}_{MN} = (w_S^*, 0) + \sum_i \frac{\xi_i}{\lambda} (0, x_{i_J}) \quad \rightarrow \mathbb{E}[\|\hat{w}_{MN}\|^2] = \|w_S^*\|^2 + \frac{\sigma^2}{\lambda} n$

Via Uniform Convergence?

For Lipschitz Loss: $\sup_{\|w\|^2 \le B^2} \left|L(w) - \hat{L}(w)\right| \le 2\, Lip \sqrt{\frac{B^2 \mathbb{E}[\|x\|^2]}{n} + O_p\left(\frac{1}{n}\right)}$

Setting $B = \|\hat{w}\|$, relevant quantity is $\frac{\|\hat{w}\|^2 \mathbb{E}[\|x\|^2]}{n} = \frac{\left(\|w_S^*\|^2 + \frac{\boldsymbol{\sigma^2}}{\boldsymbol{\lambda_n}}\boldsymbol{n}\right)(d_S + \boldsymbol{\lambda}_n)}{n} \to \frac{\frac{\sigma^2}{\lambda_n} n \cdot \lambda_n}{n} = \sigma^2$

But:

- We get $2 \cdot Lip \cdot \sqrt{\sigma^2} = 2 \cdot Lip \cdot \sigma$ instead of σ^2
- Squared loss isn't even Lipschitz
- We know such a bound is loose when $\hat{L}(w) \approx 0$
  (variance of bias p of coin $= p(1-p) \approx p$, Chernoff vs Hoeffding)

Uniform Convergence of Interpolators

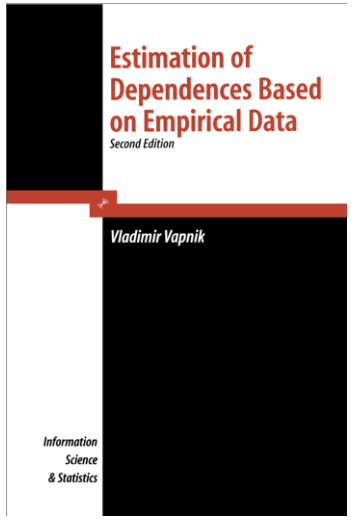
- Instead of: $\sup_{w \in \mathcal{W}} |L(w) - \hat{L}(w)|$
- Bound: $\sup_{w \in \mathcal{W}, \hat{L}(w)=0} |L(w) - \hat{L}(w)| = \sup_{w \in \mathcal{W}, \hat{L}(w)=0} L(w)$

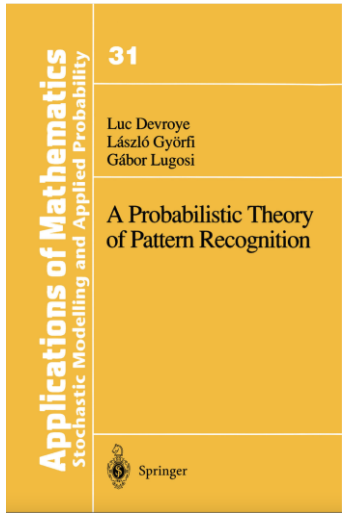
Used in the noiseless setting since at least Vapnik. Below: [Devroy et al '96]

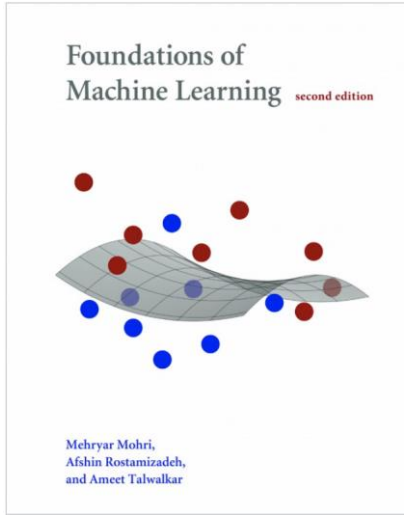
based on the random permutation argument developed in the original proof of the Vapnik-Chervonenkis inequality (1971).

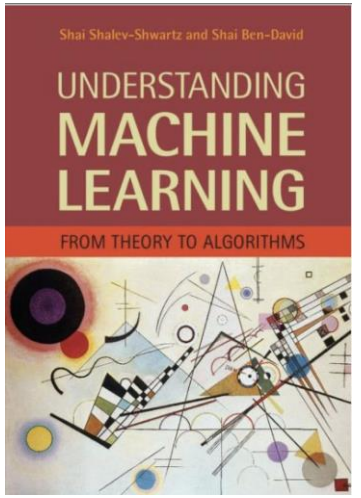
PROOF. For $n\epsilon \leq 2$, the inequality is clearly true. So, we assume that $n\epsilon > 2$. First observe that since $\inf_{\phi \in C} L(\phi) = 0$, $\widehat{L}_n(\phi_n^*) = 0$ with probability one. It is easily seen that

$$L(\phi_n^*) \leq \sup_{\phi: \widehat{L}_n(\phi)=0} |L(\phi) - \widehat{L}_n(\phi)|.$$









Realizable, Non-Realizable and Optimistic

high prob bound on $\|x\|$

$$\forall_{\|w\|^2 \le B^2} \; L(w) - \hat{L}(w) \le \tilde{O}_P\left(\frac{B^2\|x\|^2}{n} + \sqrt{\hat{L}(w)\frac{B^2\|x\|^2}{n}}\right)$$

$$\forall_{\|w\|^2 \le B^2, \hat{L}(w)=0} \; L(w) \le c\frac{B^2\|x\|^2}{n} + o_P(1)$$

If $c = 1$: $L(w_{MN}) \le \sup(\cdots) \xrightarrow{n\to\infty} \frac{\|w_{MN}\|^2\|x\|^2}{n} = \sigma^2$ (for $1 \ll \lambda_n \ll n$)

[S Sridharan Tewari 2010]: $c \le 200{,}000 \log^3 n$

[Koehler Zhou Southerland S 2021] : at least for Gaussian $x \sim \mathcal{N}(\mu, \Sigma)$: $c = 1$

- Recall Expected Rademacher Complexity of $\{x \mapsto \langle w, x\rangle | w \in \mathcal{W}\}$ when $x \sim \mathcal{N}(0, \Sigma)$:

$$\mathcal{R}_n(\mathcal{W}, \Sigma) = \mathbb{E}_{x_1,..,x_n \sim \mathcal{N}(0,\Sigma), z_1..z_n \sim Unif(\pm 1)} \left[\sup_{w \in \mathcal{W}} \left| \frac{1}{n} \sum_i z_i \langle w, x_i \rangle \right| \right]$$

- **Theorem** (informal): For any Σ, and any splitting $\Sigma = \Sigma_1 + \Sigma_2$ s.t. $rank(\Sigma_1) = o(n)$, it holds with high probability that for all $w \in \mathcal{W}$

$$L(w) \leq \left(1 + o(1)\right) \left(\sqrt{\hat{L}(w)} + \mathcal{R}_n(\mathcal{W}, \Sigma_2) \right)^2$$

$$\Rightarrow \sup_{w \in \mathcal{W}, \hat{L}(w) = 0} L(w) \leq \left(1 + o(1)\right) \mathcal{R}_n^2(\mathcal{W}, \Sigma_2)$$

- **Corollary**: Since $\mathcal{R}_n(\{\|w\|_2 \leq B\}, \Sigma) = \sqrt{\frac{B^2 \mathbb{E}[\|x\|^2]}{n}}$, then w.h.p.

$$\sup_{\|w\| \leq B, \hat{L}(w) = 0} L(w) \leq \left(1 + o(1)\right) \frac{B^2 \mathbb{E}_{x \sim \Sigma_2}[\|x\|^2]}{n}$$

➔Establishes consistency in Junk Features model (for $0 < \lambda_n \ll n$)

And unlike direct approach: applies also to approx min norm near-interpolators

Benign Overfitting Condition

Following [Bartlett et al 19][Tsigler Bartlett 20]

For $y = \langle w^*, x \rangle + \mathcal{N}(0, \sigma^2)$, $x \sim \mathcal{N}(0, \Sigma)$

and (w.l.o.g.) $w^* = 1$, $\Sigma = R \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix} R^T$ (with $RR^T = I$),

If: $\boldsymbol{rank(\Sigma_1) = o(n)}$, $\boldsymbol{tr(\Sigma_2) = o(n)}$, and $\boldsymbol{\text{eff}-\text{rank}(\Sigma_2) \overset{\text{def}}{=} \frac{tr(\Sigma_2)^2}{tr(\Sigma_2^2)} = \omega(n)}$

Then $\hat{w}_{MN} = \arg\min \|w\| \ \ s.t. \hat{L}(w) = 0$ is **consistent**: $L(\hat{w}_{MN}) \to \sigma^2$

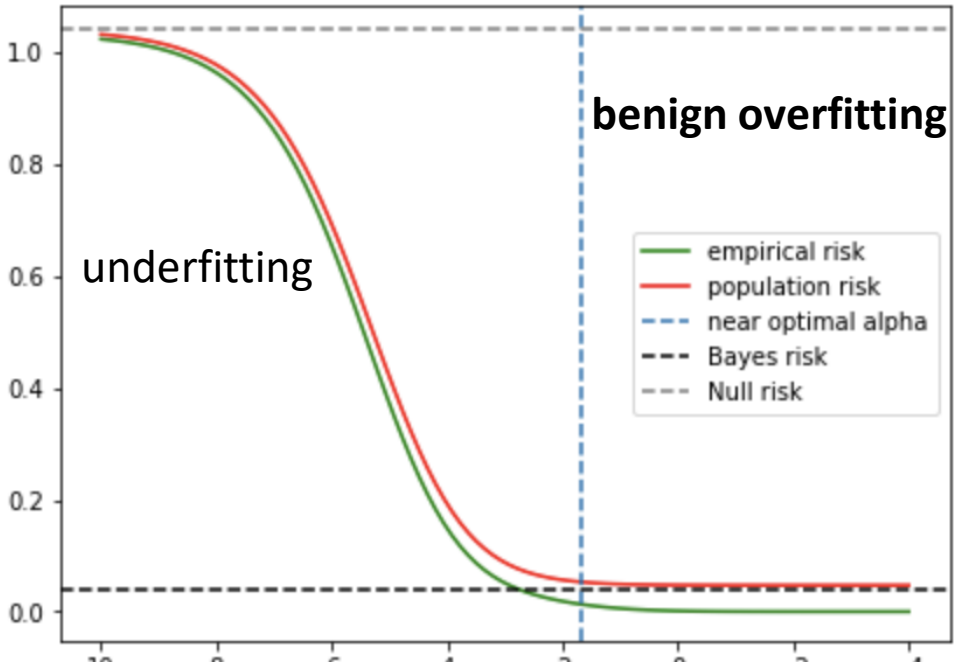
Recovered from uniform convergence analysis by calculating: $\|\hat{w}_{MN}\|^2 = \big(1 + o(1)\big) \frac{\sigma^2 n}{tr(\Sigma_2)}$

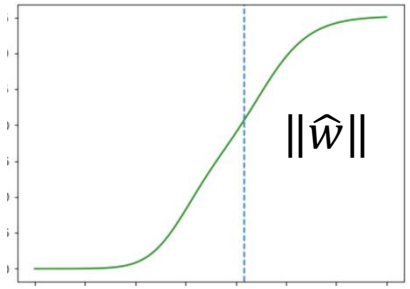
d_s	$d_J = 3000 \gg n = 300$
w_s^*	0
signal x_S	junk x_J

$\sigma_i = \frac{1}{d}$ (flat)

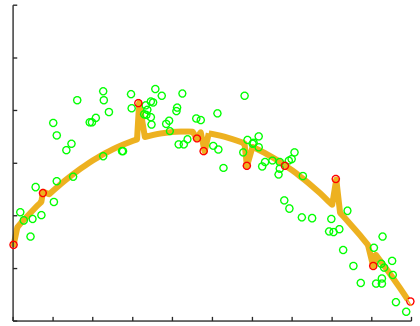
Effective rank:

$$\frac{tr(\Sigma_J)^2}{tr(\Sigma_J^2)} = \frac{\left(\sum_i \frac{1}{d}\right)^2}{\sum_i \frac{1}{d^2}} = d$$





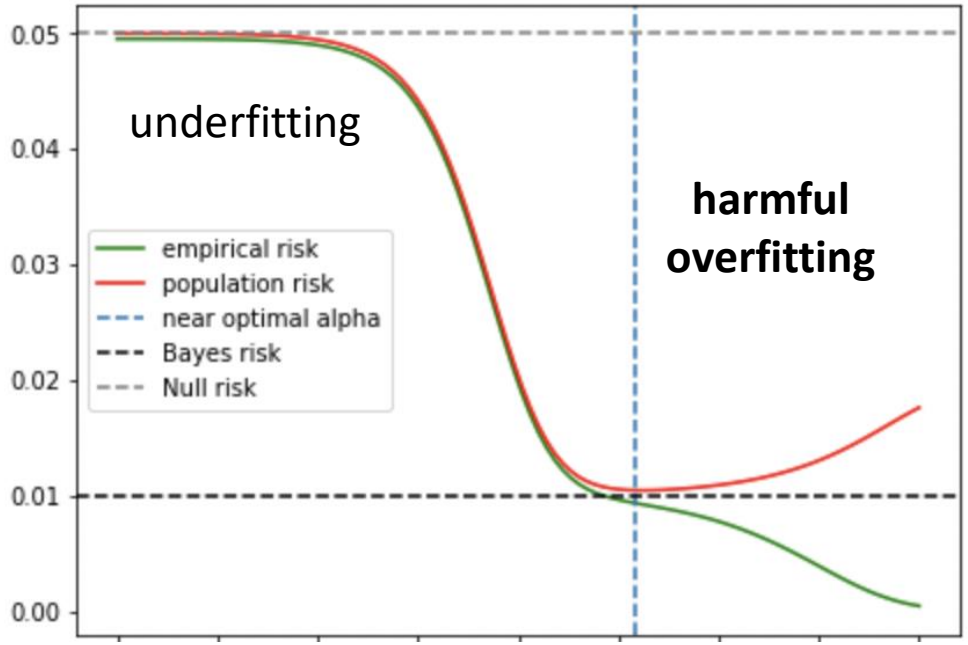
$\cong$



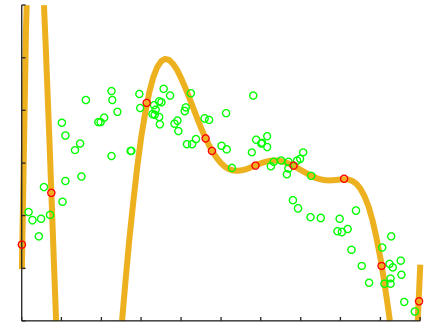
$\sigma_i = \frac{1}{i^2}$

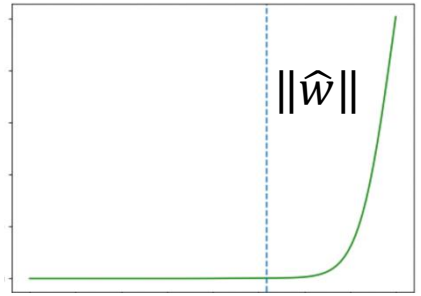
Effective rank:

$$\frac{tr(\Sigma_J)^2}{tr(\Sigma_J^2)} = \frac{\left(\sum_i \frac{1}{i^2}\right)^2}{\sum_i \frac{1}{i^4}} = \frac{5}{2}$$



$\cong$





d_s	$d_J = 3000 \gg n = 300$
w_s^*	0
signal x_S	junk x_J

$\sigma_i = \frac{1}{d}$ (flat)

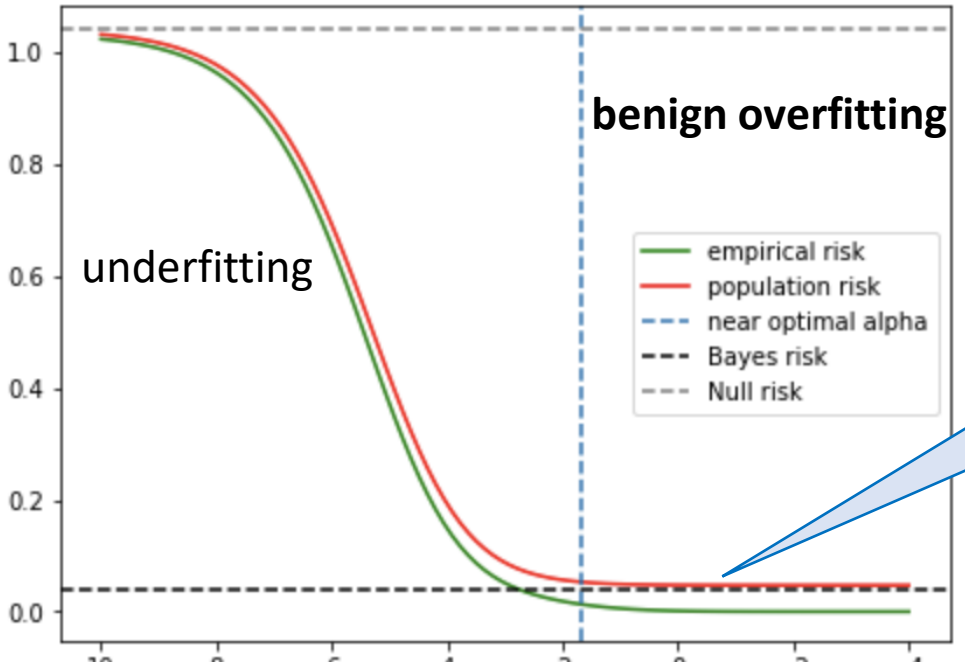
Effective rank:

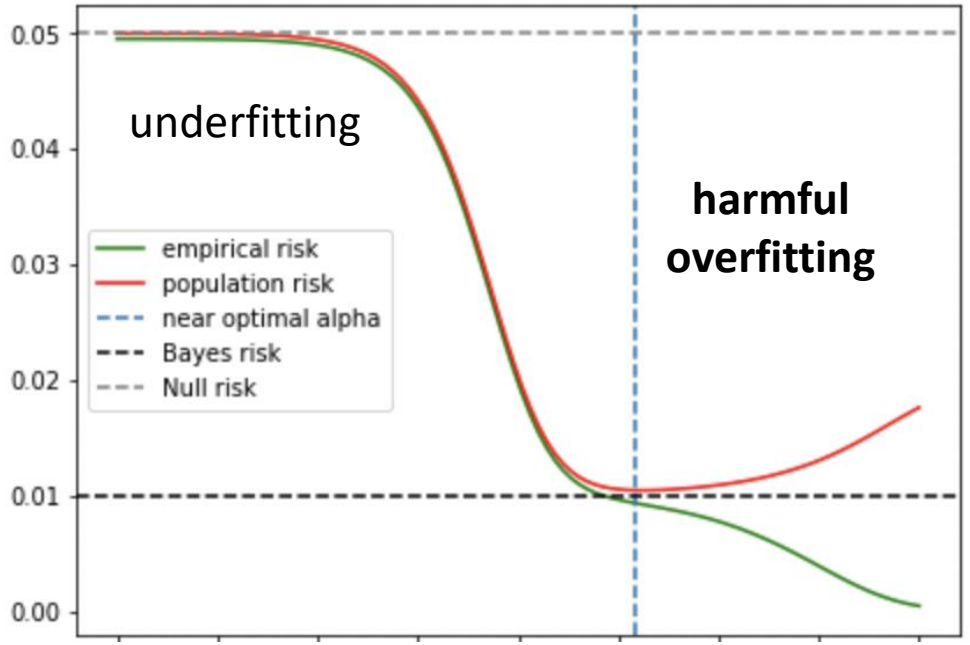
$$\frac{tr(\Sigma_J)^2}{tr(\Sigma_J^2)} = \frac{\left(\sum_i \frac{1}{d}\right)^2}{\sum_i \frac{1}{d^2}} = d$$

$\sigma_i = \frac{1}{i^2}$

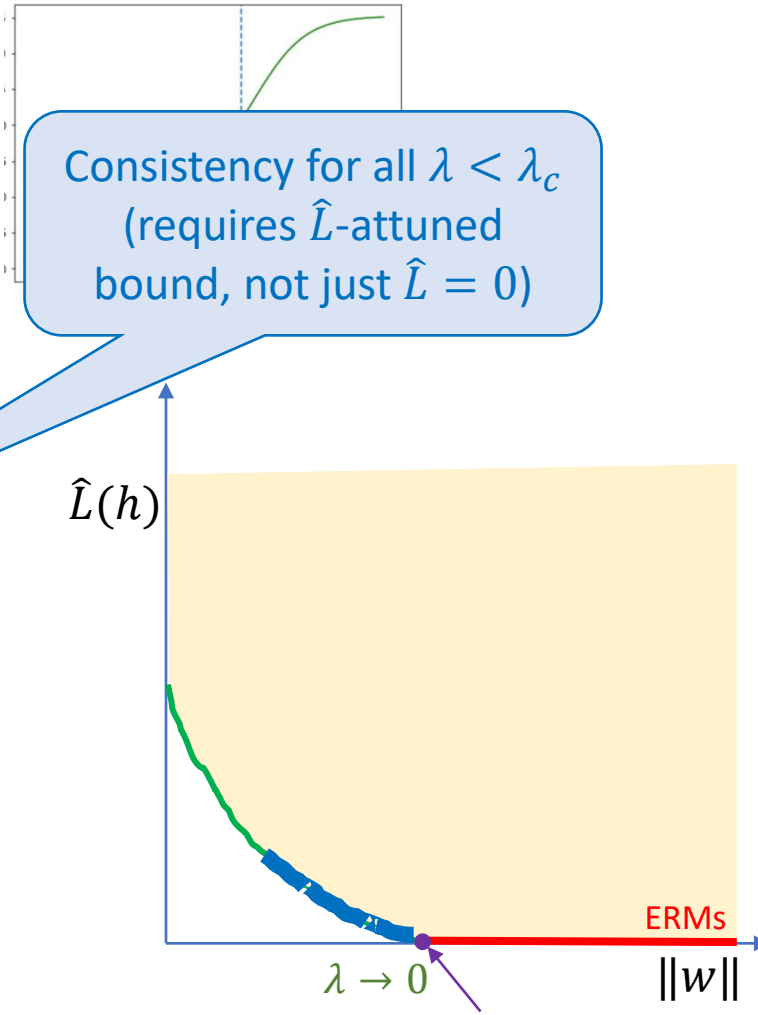
Effective rank:

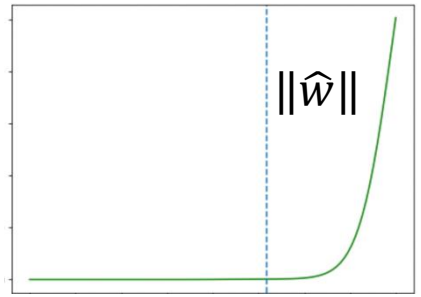
$$\frac{tr(\Sigma_J)^2}{tr(\Sigma_J^2)} = \frac{\left(\sum_i \frac{1}{i^2}\right)^2}{\sum_i \frac{1}{i^4}} = \frac{5}{2}$$





Consistency for all $\lambda < \lambda_c$ (requires $\hat{L}$-attuned bound, not just $\hat{L} = 0$)





Summary

- What we understand well using old theory:
  - Learning with overparametrized models
  - Learning improving when dimensionality increasing (since true complexity measure is not dimensionality)
  - Implicit regularization from optimization

- Questions we need to answer:
  - What is the implicit bias of the optimization methods we use?
  - How does the architecture affect the implicit bias?
  - What is the true complexity measure/inductive bias

- What requires rethinking:
  - Benign overfitting (interpolation learning in a noisy setting)

- Can we use uniform convergence to understand benign overfitting and generalization with an implicit inductive bias?
  - Maybe…