what can deep learning learn from linear regression?

Benjamin Recht
University of California, Berkeley

Collaborators

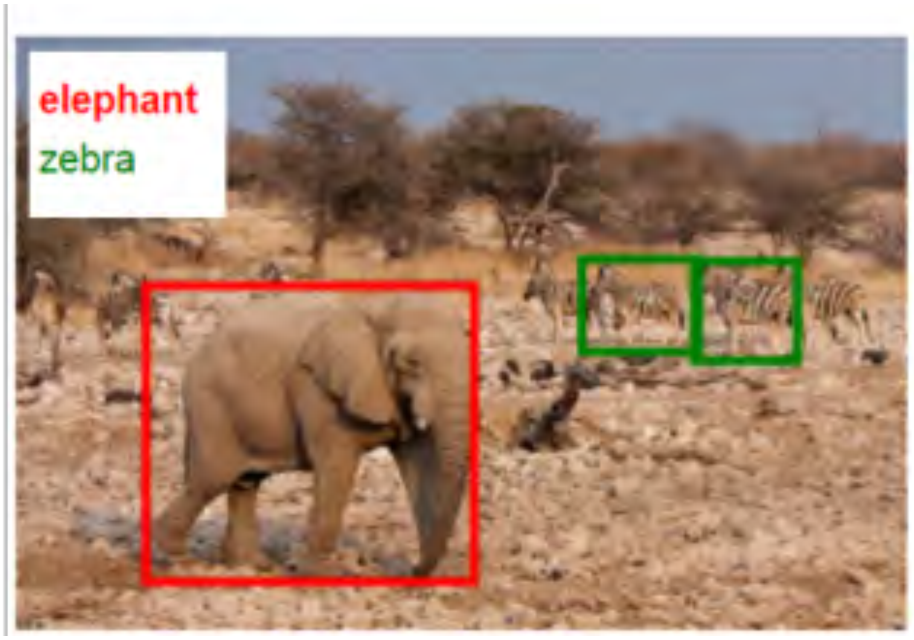


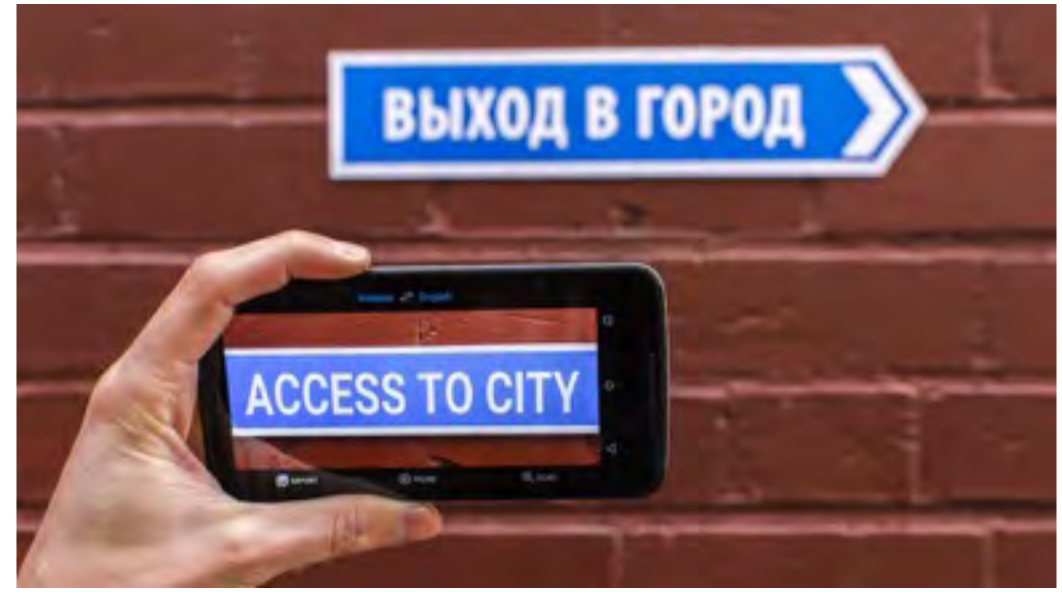




- Joint work with Samy Bengio, Moritz Hardt, Michael Jordan, Jason Lee, Max Simchowitz, Oriol Vinyals, and Chiyuan Zhang.

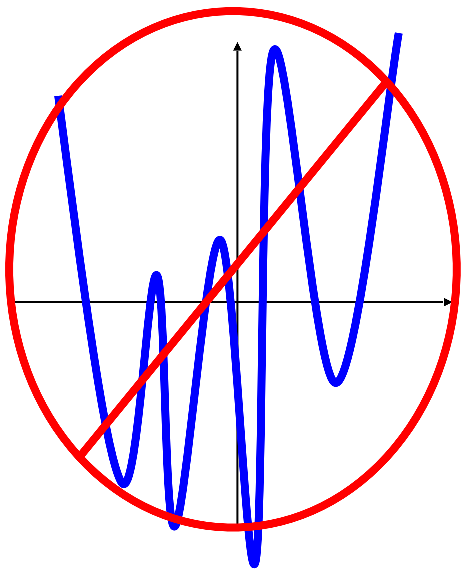
Successes of Depth Abound

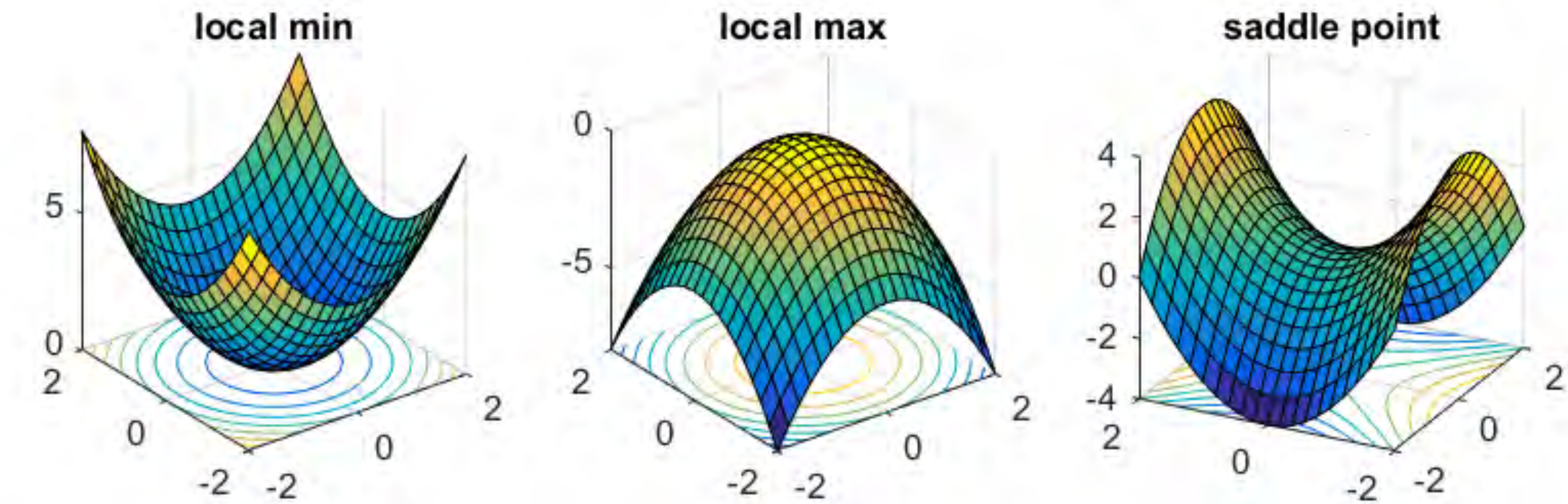




trustable, scalable, predictable

What makes optimization of deep models hard?





No clear consensus!

"We prove that recovering the global minimum becomes harder as the network size increases." arXiv:1412.0233

"Difficulty originates from the proliferation of saddle points, not local minima, especially in high dimensional problems of practical interest. " arXiv:1406.2572

"Local extrema with low generalization error have a large proportion of almost-zero eigenvalues in the Hessian with very few positive or negative eigenvalues." arXiv:1611.01838

It's hard to hit a saddle

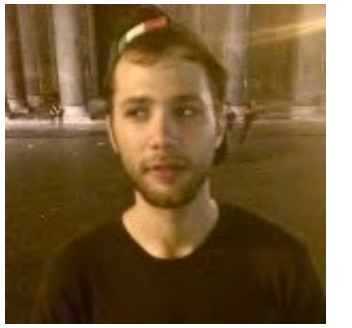
$$f(x) = \frac{1}{2}\sum_{i=1}^{d} a_i x_i^2$$

Gradient descent: $x_i^{(k+1)} = (1 - t a_i) x_i^{(k)}$

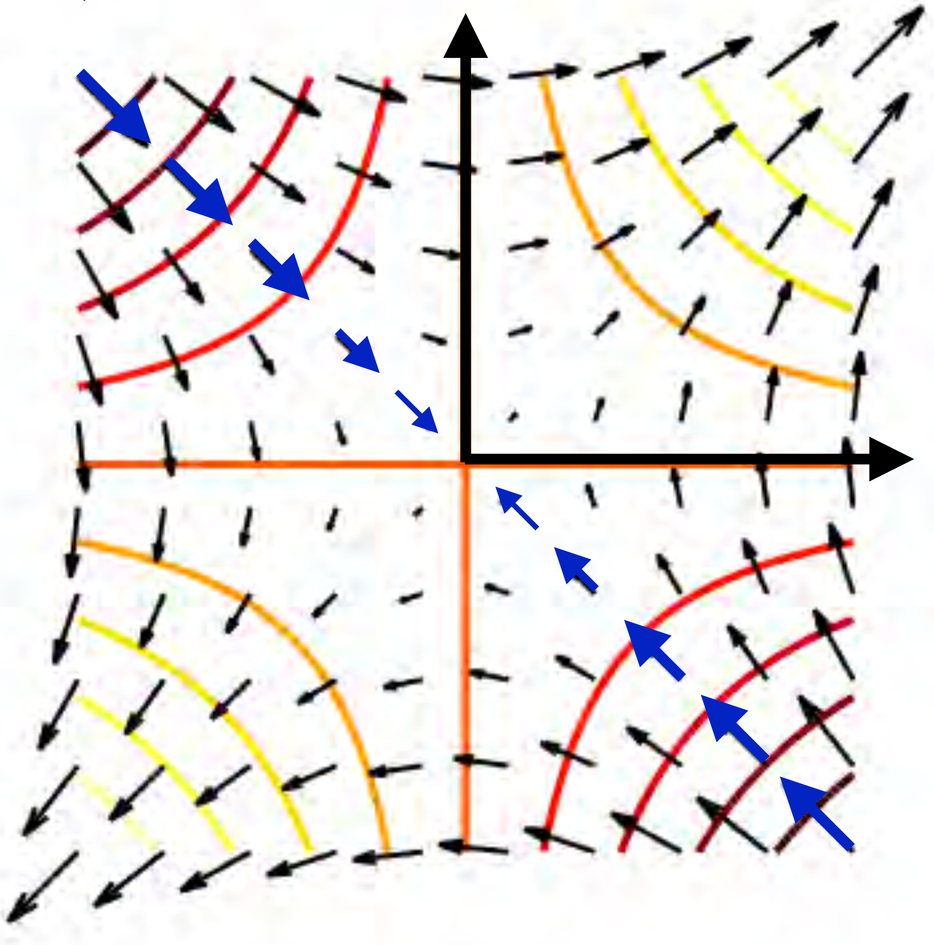
After k steps $x_i^{(k)} = (1 - t a_i)^k x_i^{(0)}$

If $t|a_i| < 1$ {
- converges to 0 if all a_i are positive
- diverges almost surely if single a_i is negative

It's hard to hit a saddle



$f(x, y) = xy$



If you are not on the line {x=-y}, you diverge at an exponential rate

This picture fully generalizes to the nonconvex case

Thm: [Lee et al, 2016] For the short-step gradient method, the basin of attraction of strong saddle points has measure zero.

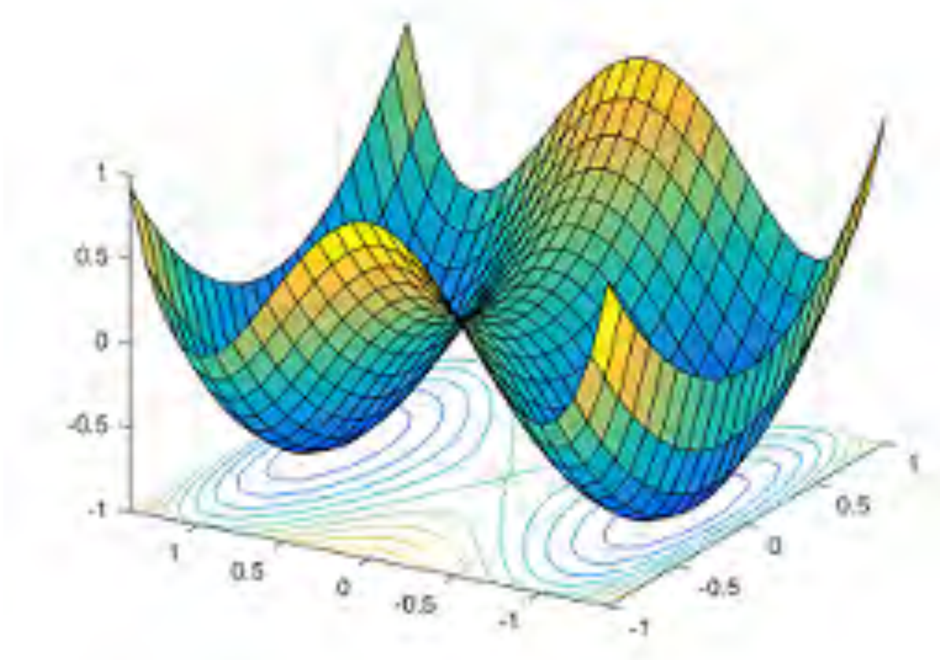
Simple consequence of the *Stable Manifold Theorem* (Smale *et al*)

This is our fault, optimizers.

- Too many fragile examples in text books

- Minor perturbations in initial conditions always repel you from saddles.

$$f(x,y) = x_1^4 - 2x_1^2 + x_2^2$$



Flatness is what makes things hard

$$f(x) = \sum_{i,j=1}^{d} Q_{ij} x_i^2 x_j^2$$

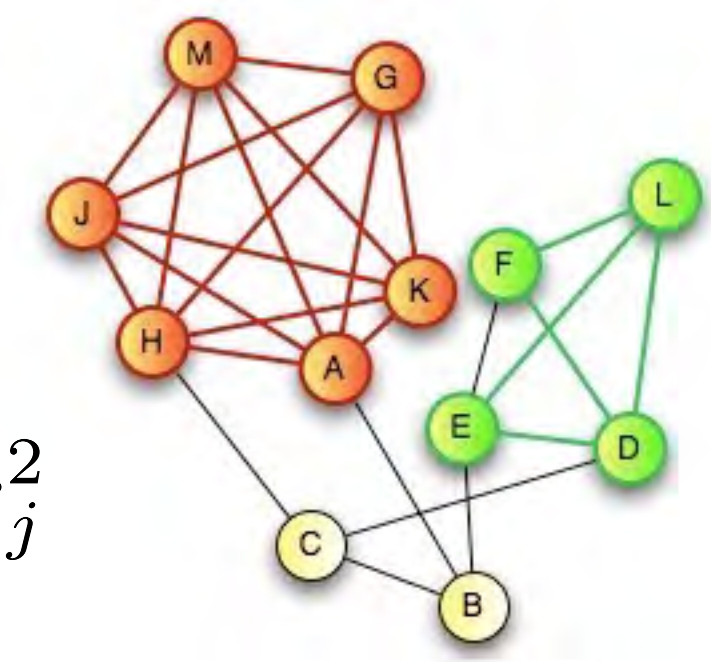
$\nabla f(0) = 0$ Is 0 a global min, saddle, or global max?

$\nabla^2 f(0) = 0$ f is *super flat* at 0.

Deciding if there is a descent direction at 0 is NP-complete

$$f(x) = \sum_{i,j=1}^{d} Q_{ij} x_i^2 x_j^2$$

$$Q = I - A + s \cdot \mathbf{1}\mathbf{1}^T$$



A = adjacency matrix of G

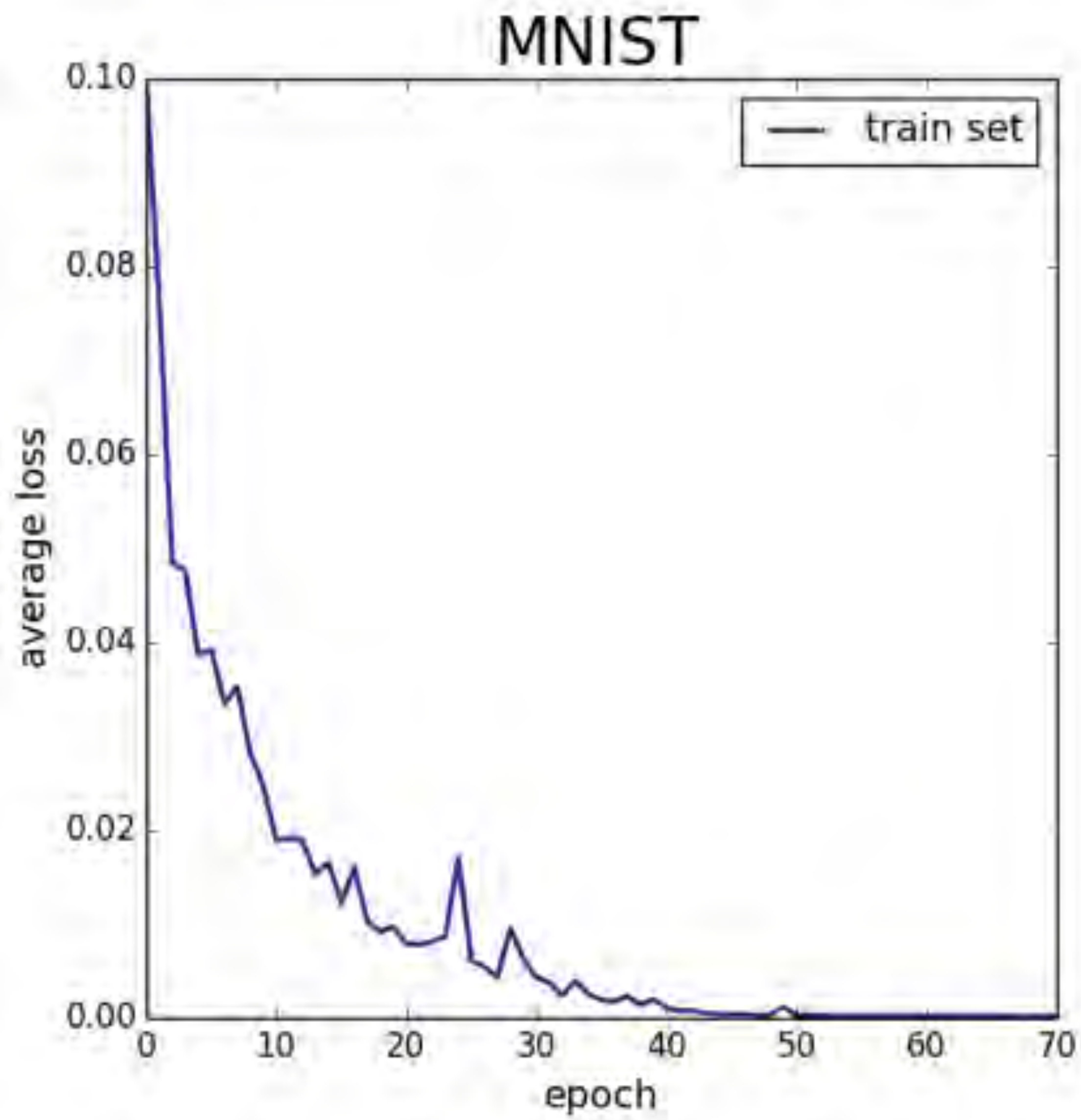
G has an clique of size larger than 1/(1-s) if and only if 0 is not a local minimizer*.

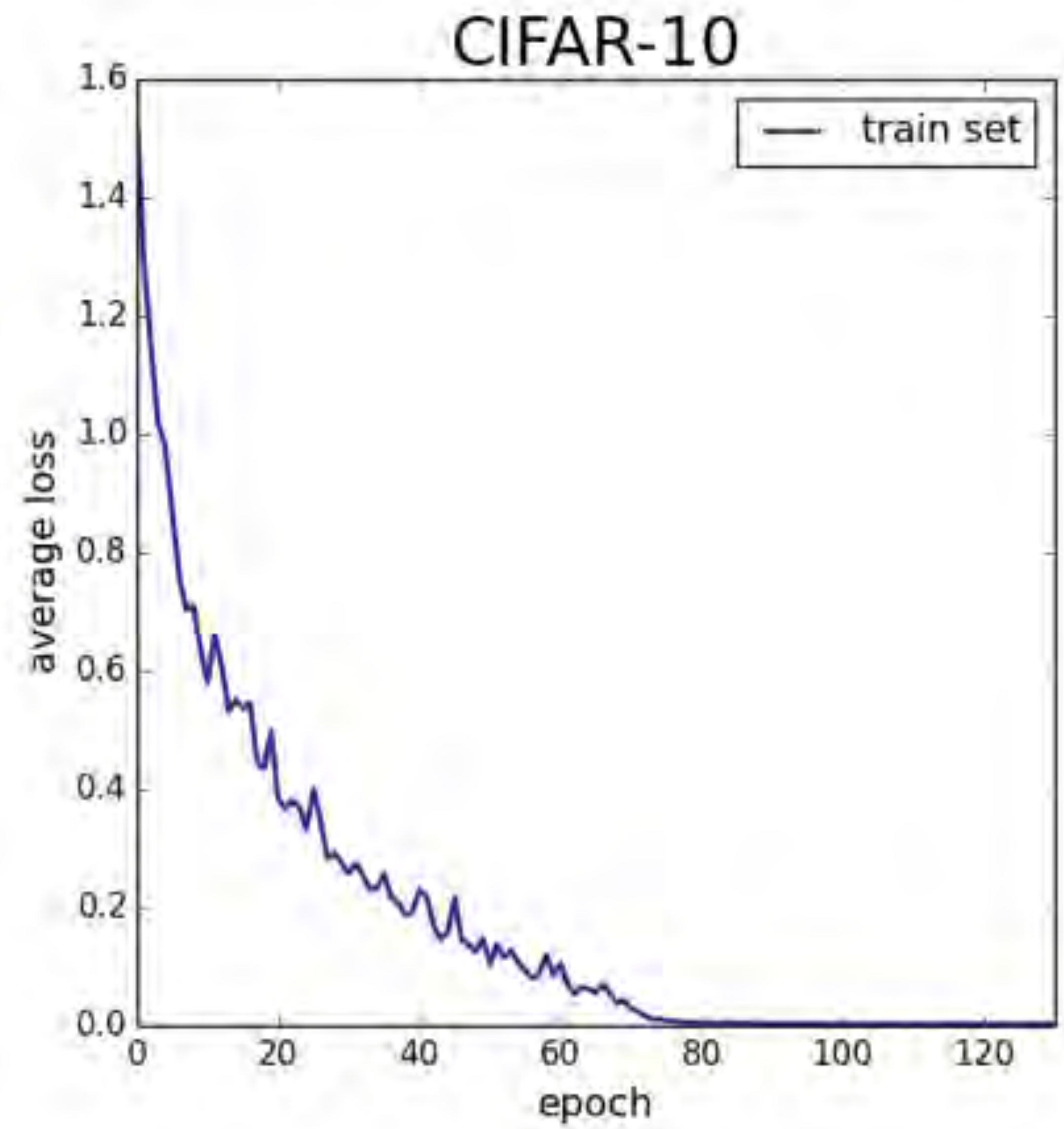
Thm [Barak *et al.* 2016]: Finding a maximum clique is F-hard

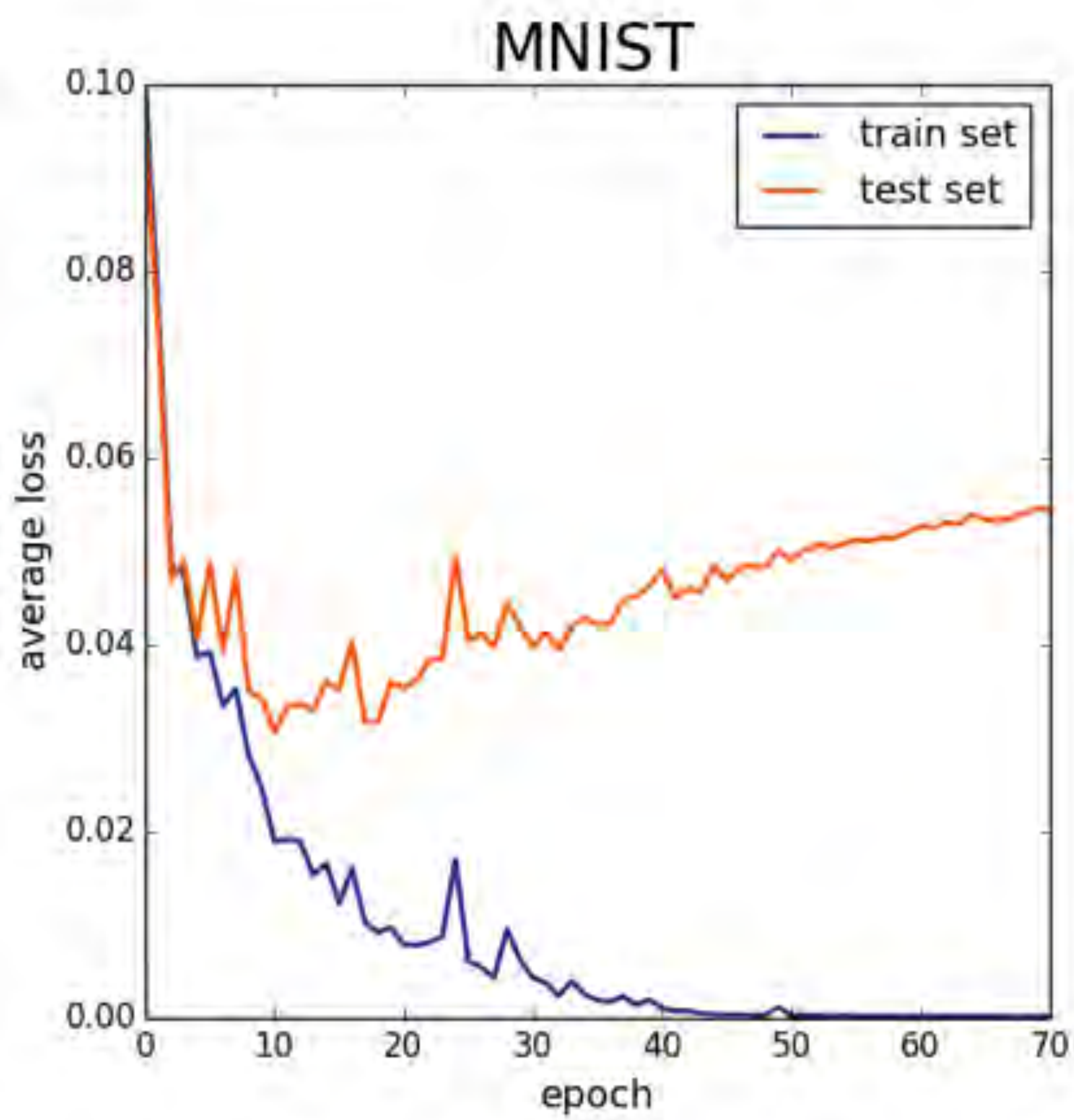
If the best solutions are flat local minima, can we ever find them?

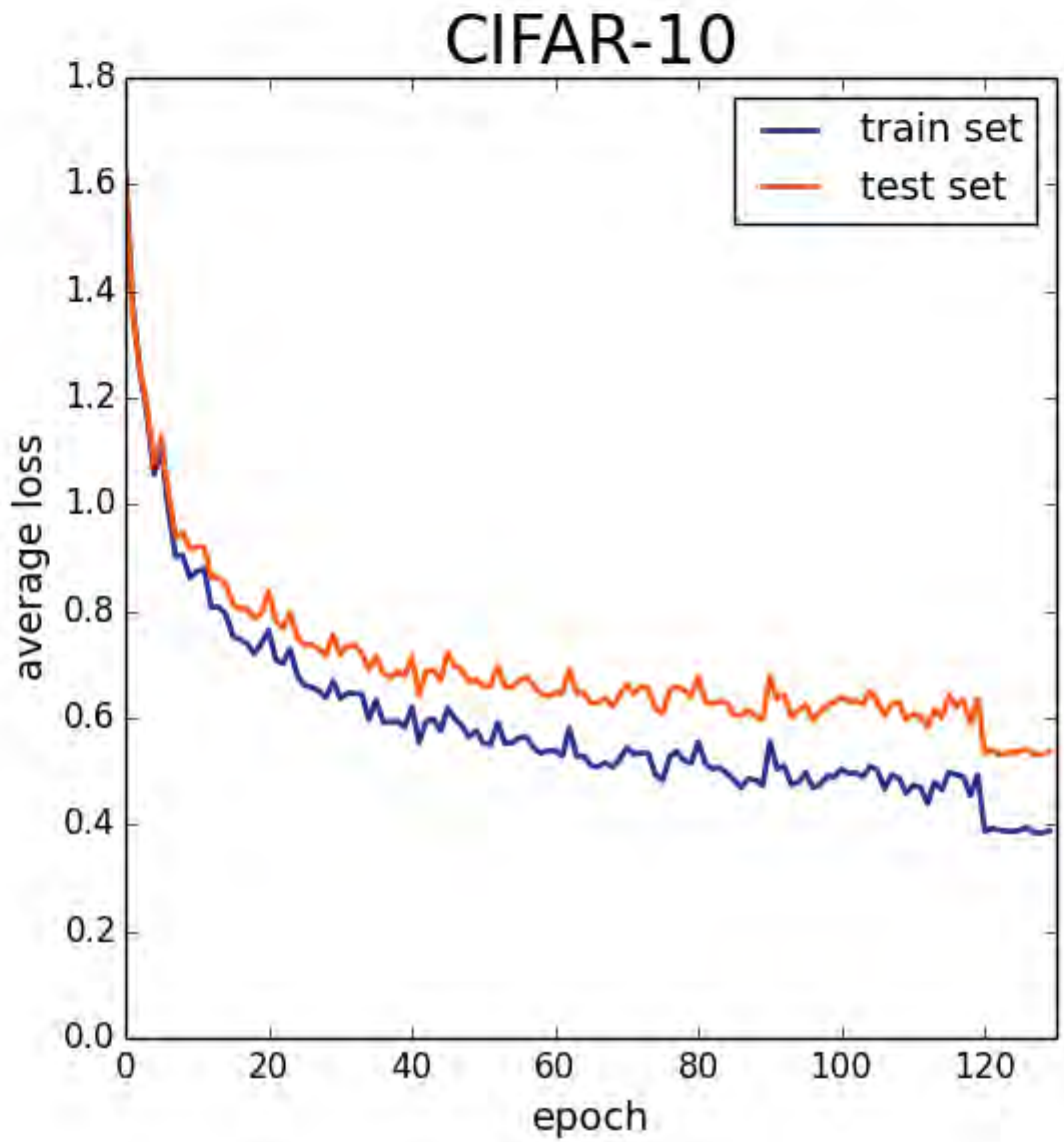
* http://www.ti.inf.ethz.ch/ew/lehre/ApproxSDP09/notes/copositive.pdf

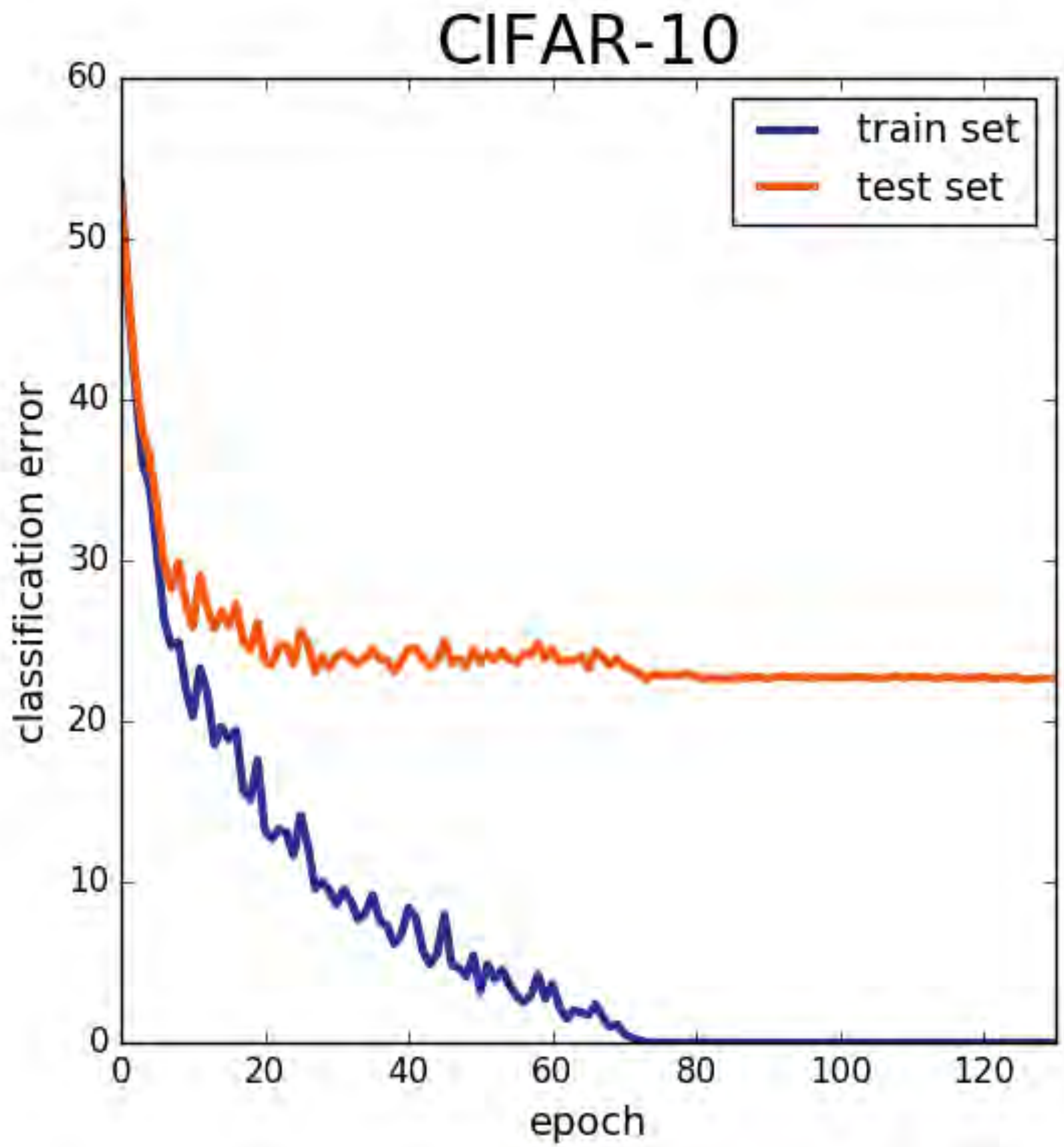
Is deep learning as hard as maximum clique?






MNIST
0.10
0.08
0.06
0.04
0.02
0.00
average loss
0
10
20
30
40
50
60
70
epoch
train set
test set
CIFAR-10
2.0
1.5
1.0
0.5
0.0
average loss
0
20
40
60
80
100
120
epoch
train set
test set


CIFAR-10
average loss
epoch
train set
test set
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
1.8
0
20
40
60
80
100
120



CIFAR-10
train set
test set
classification error
epoch
0
10
20
30
40
50
60
0
20
40
60
80
100
120

Generalization in Machine Learning

Given: i.i.d. sample $S = \{z_1, \ldots, z_n\}$ from dist D

Goal: Find a good predictor function f

$$R[f] = \mathbb{E}_z \mathrm{loss}(f; z)$$

Population risk
(test error)

unknown!

$$R_S[f] = \frac{1}{n}\sum_{i=1}^{n} \mathrm{loss}(f; z_i)$$

Empirical risk
(training error)

Minimize using SGD!

Generalization error: $R[f] - R_S[f]$

How much empirical risk underestimates population risk

We can compute R_S…

When is it a good proxy for R?

Fundamental Theorem of Machine Learning

$$R[f] = (R[f] - R_S[f]) + R_S[f]$$

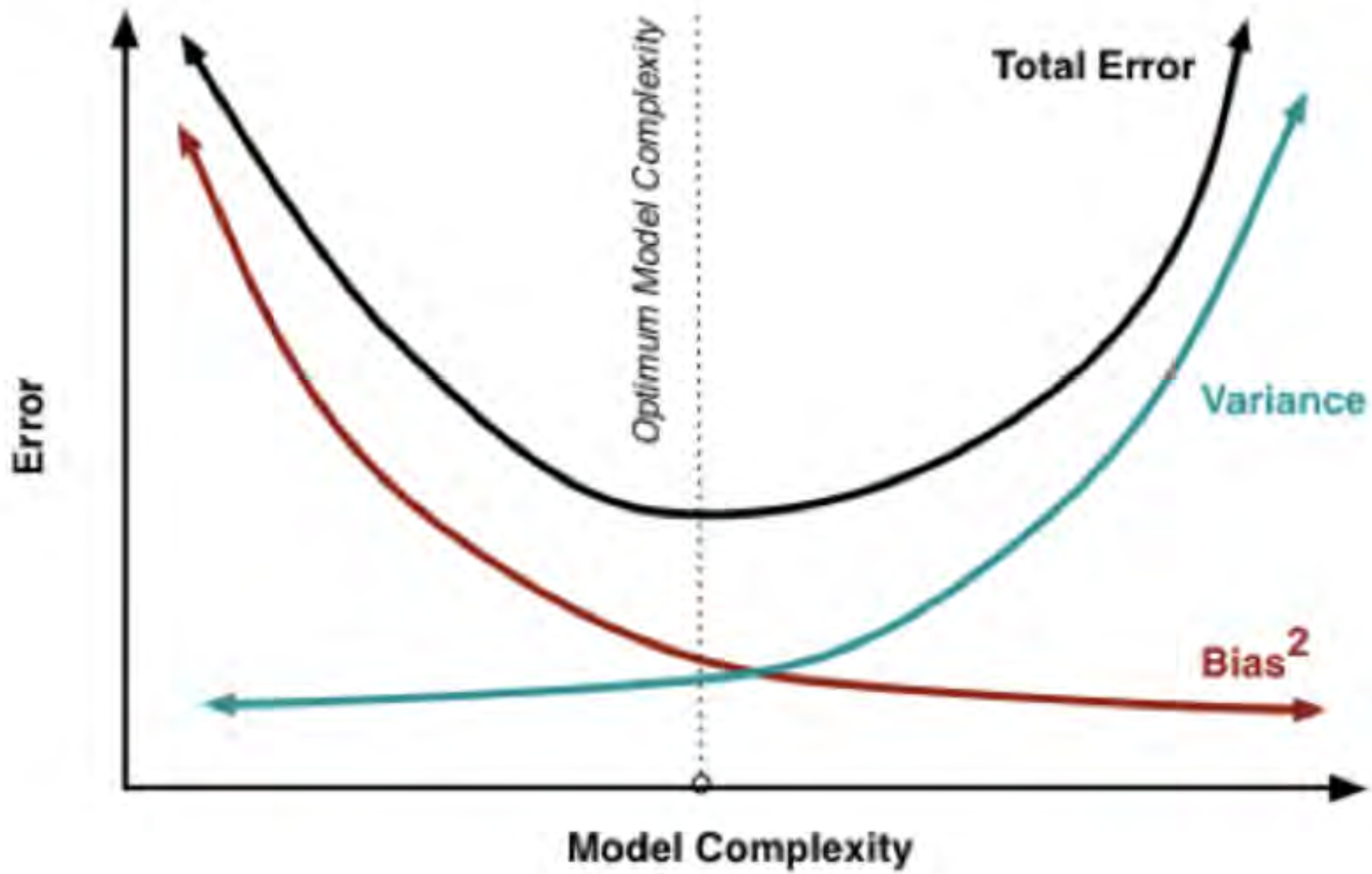
population risk | generalization error | training error

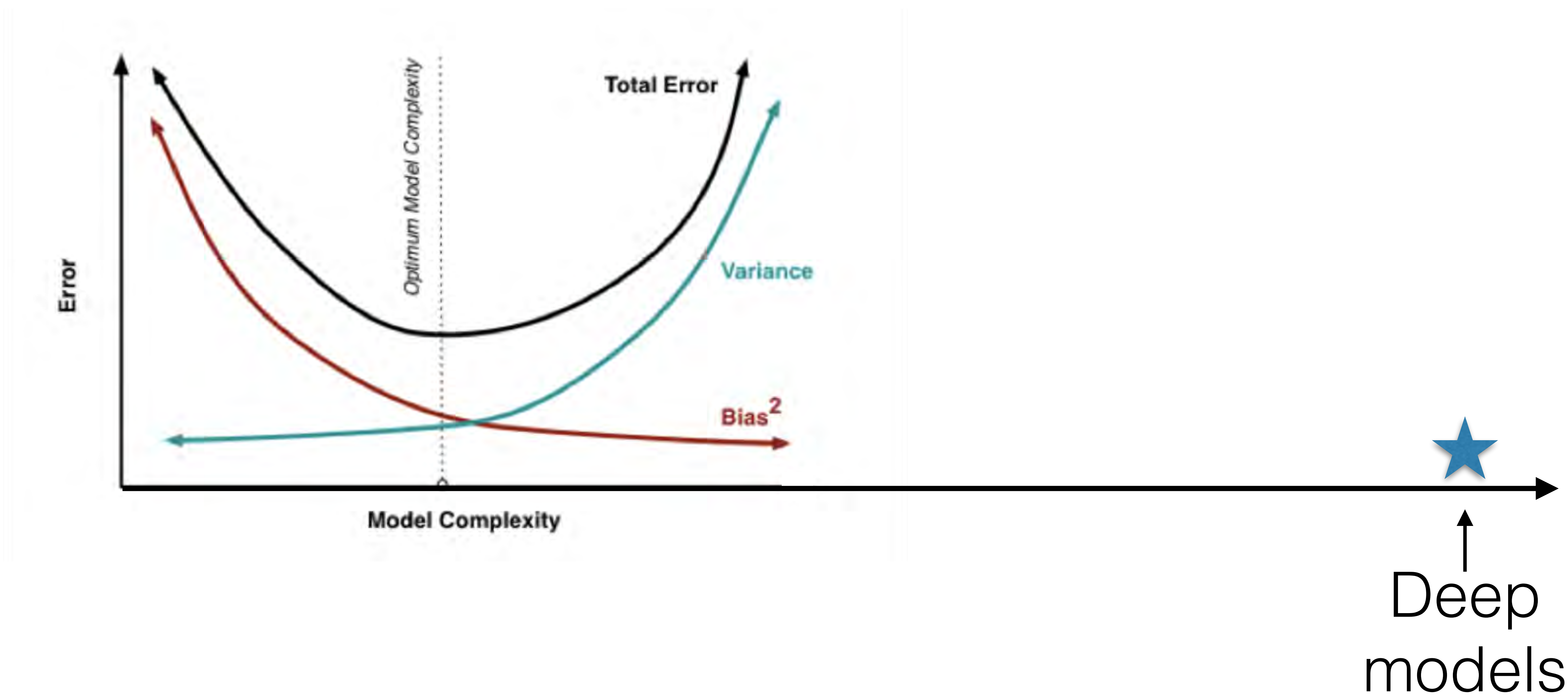
- small training error implies risk ≅ generalization error
- zero training error *does not imply* overfitting

$$R[f] = (R[f] - R[f_{\mathcal{H}}]) \quad \text{error vs best in class}$$

$$+ (R[f_{\mathcal{H}}] - R[f_\star]) \quad \text{approximation error}$$

$$+ R[f_\star] \quad \text{irreducible error}$$


Error
Optimum Model Complexity
Total Error
Variance
Bias2
Model Complexity



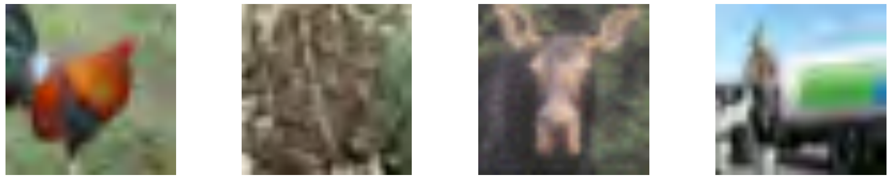
Models where p>20n are ubiquitous

How to reduce generalization error?

- Model capacity
- Regularization (norms, dropout, etc.)
- Implicit regularization (early stopping)
- Data augmentation (fake data, crops, shifts, etc.)

All of these are sufficient but by no means necessary!

Zhang, Bengio, Hardt, R., Vinyals



CIFAR10

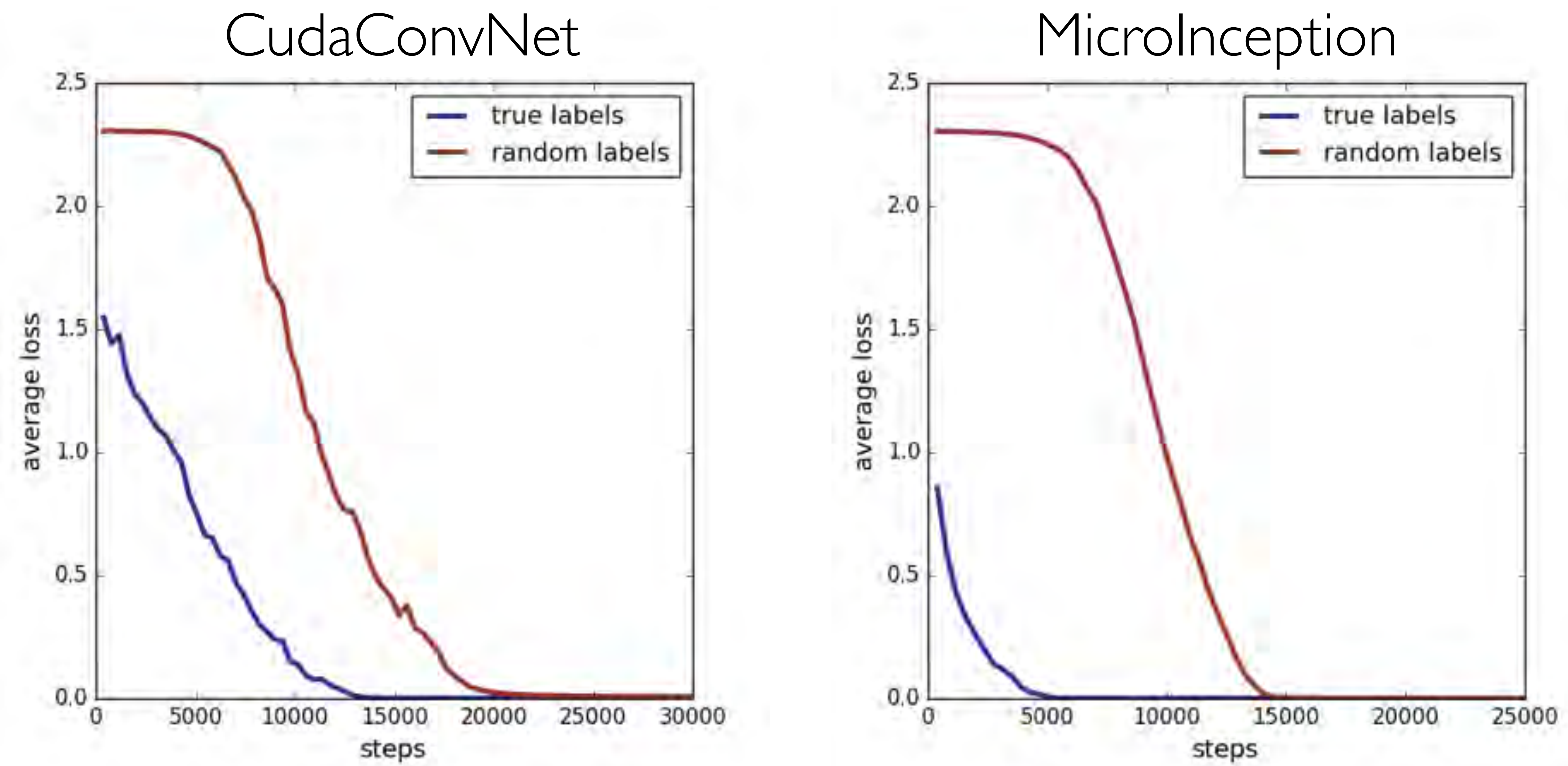
n=50,000
d=3,072
k=10

What happens when I turn off the regularizers?

Model	parameters	p/n	Train ***loss***	Test ***error***
CudaConvNet	145,578	2.9	0	23%
CudaConvNet (with regularization)	145,578	2.9	0.34	18%
MicroInception	1,649,402	33	0	14%
ResNet	2,401,440	48	0	13%

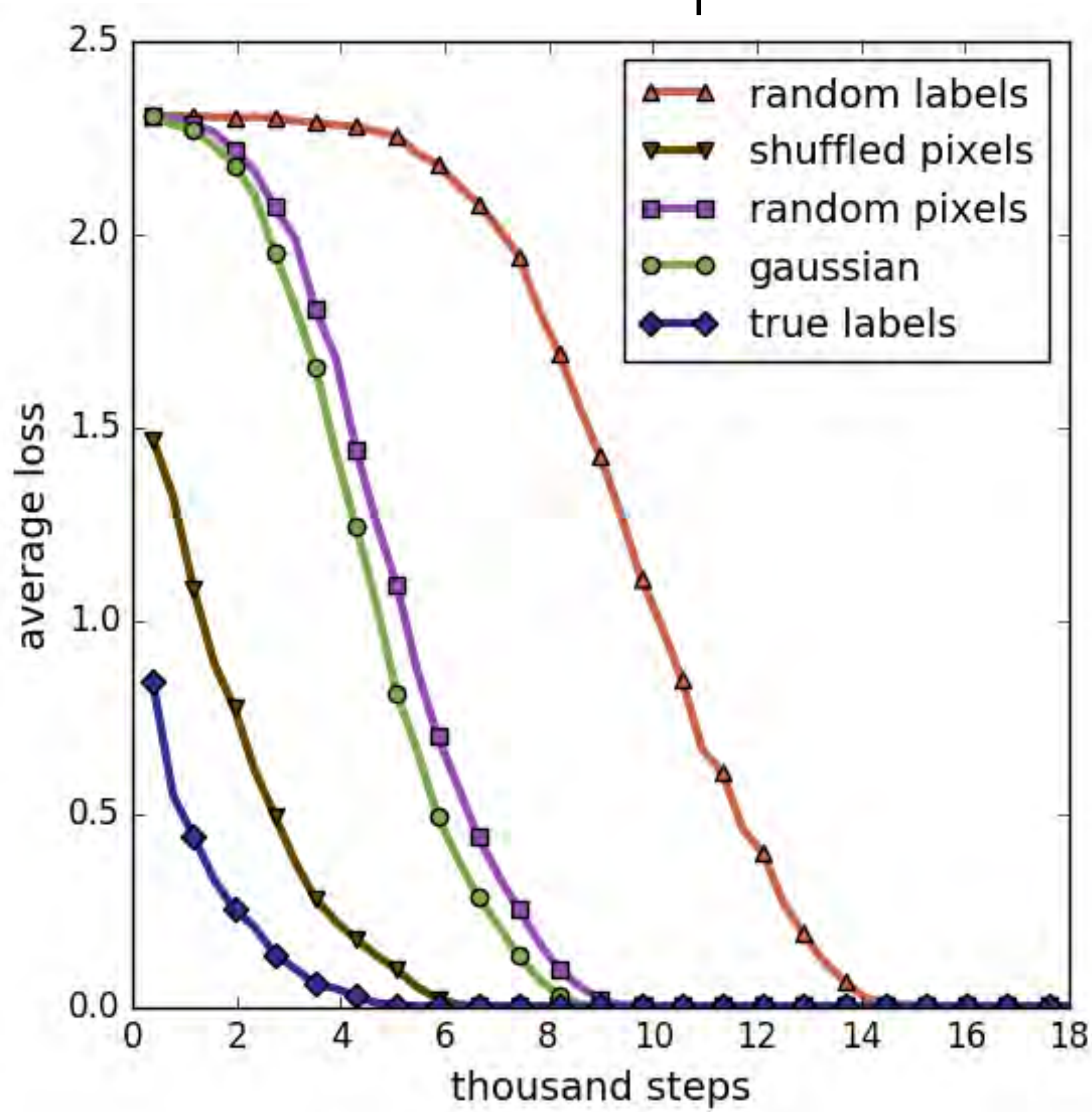
CIFAR10 with random labels

n=50,000
d=3,072
k=10



MicroInception

n=50,000
d=3,072
k=10
p=1,649,402



IMAGENET

$n = 1.3M$
$d = 150528$
$k = 1000$

accordion

ant

airplane

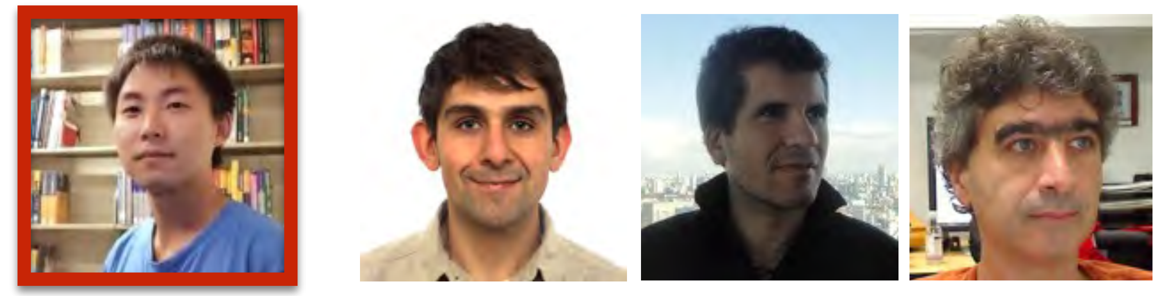
Inception model: 27 million parameters

arXiv:1512.00567v3

$d > 20n$

Rand. Labels	Fake Data	l2 reg/ dropout	Train top-1	Test top-1	Train top-5	Test top-5
No	Yes	Yes	13.7%	23.4%	2.5%	6.5%
No	Yes	No	8.2%	27.1%	1.0%	9.0%
No	No	Yes	0.6%	29.8%	0%	11.2%
No	No	No	0.5%	39.7%	0%	19.3%
Yes	No	No	4.8%	99.9%	0.9%	99.5%

Deep Nets and Generalization



Zhang, Bengio, Hardt, R., Vinyals

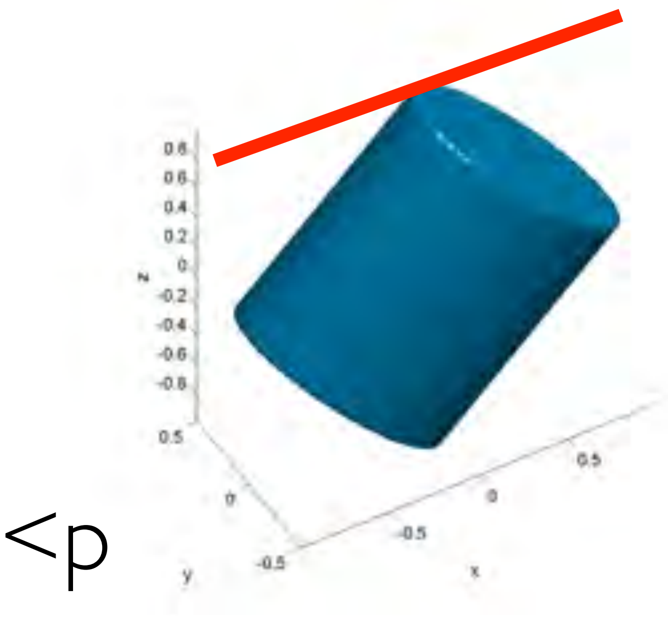
- Large, unregularized deep nets outperform shallower nets with regularization.
- Most models can fit arbitrary label patterns, even on large data-sets like imagenet.
- Popular models can fit *structureless* noise

How can we explain these phenomena?

Avoiding overfitting is hard.

- This is true in the linear case too!

$$\underset{w}{\text{minimize}} \;\; \|y - Xw\|^2$$



X n x p, $n<p$

- *Infinite* number of *global* minima.
- All global minima have the *same* Hessian.
- *At least p-n* of the Hessian eigenvalues are zero.

Which solution should we pick?

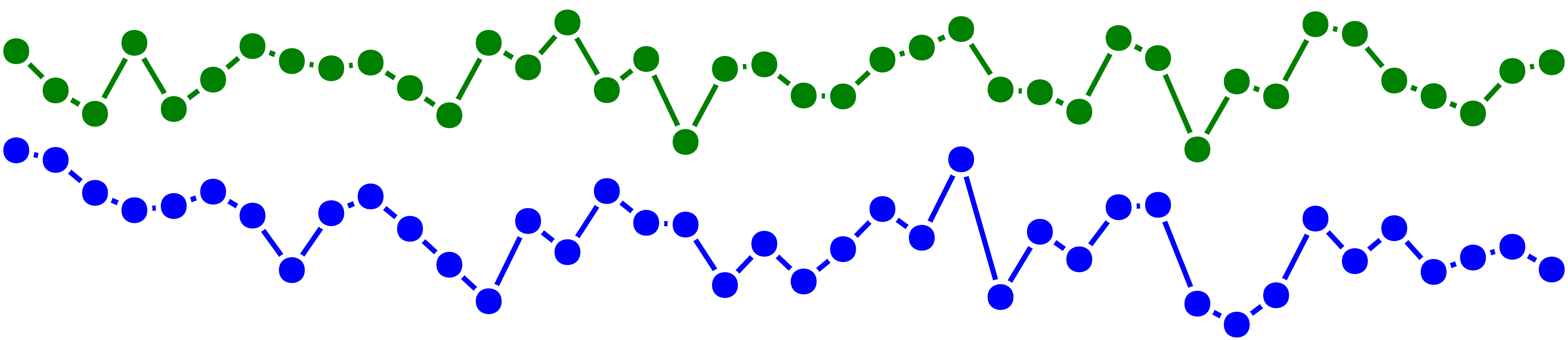
- Why do we generalize when fitting the labels exactly?
- Happens for linear models! $f(x) = w^T x$

$$\text{minimize} \quad \sum_{i=1}^{n} (w^T x_i - y_i)^2$$

SGD solution →

$$\begin{aligned} &\text{minimize} && \|w\| \\ &\text{subject to} && Xw = y \end{aligned}$$

If you run SGD you find the minimum norm solution



- Why do we generalize when fitting the labels exactly?
- Happens for linear models!

$$\text{minimize} \quad \sum_{i=1}^{n} (w^T x_i - y_i)^2$$

SGD solution → $\begin{array}{ll} \text{minimize} & \|w\| \\ \text{subject to} & Xw = y \end{array}$

If you run SGD you find the minimum norm solution

$$w_{t+1} = w_t - \eta_t \frac{d\,\text{loss}}{dz} x_i \;\rightarrow\; w_{\text{SGD}} = \sum_{i=1}^{n} \alpha_i x_i$$

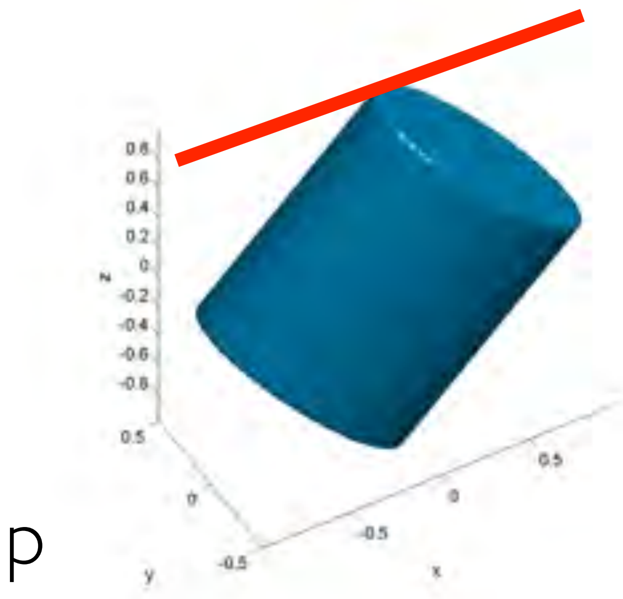
$x_i^T w_{\text{SGD}} = y_i$ → w_{SGD} satisfies KKT conditions

Avoiding overfitting is hard.

- This is true in the linear case too!

$$\begin{array}{ll} \text{minimize} & \|w\|_{\mathcal{A}} \\ \text{subject to} & Xw = y \end{array}$$

$\boldsymbol{X}$ $n \times p$, $n<p$



- Infinite number of global minima. Which one should we pick?

- Regularize to leverage structure

Sparsity | Rank | Smoothness | *Algorithm?*

Can label interpolation work for linear models?

- If you run SGD you find minimum norm solution

$$\text{minimize} \quad \sum_{i=1}^{n} (w^T x_i - y_i)^2$$

SGD solution ⇒

$$\begin{array}{ll} \text{minimize} & \|w\| \\ \text{subject to} & Xw = y \end{array}$$

KERNELIZE

KERNELIZE

$$f(x) = w^T x$$

$$x_1^T x_2 = k(x_1, x_2)$$

- If you run SGD you find minimum norm solution

minimize $\sum_{i=1}^{n} (f(x_i) - y_i)^2$

SGD solution ⇒ minimize $\|f\|$ subject to $f(x_i) = y_i$

$$f_\star(x) = \sum_{i=1}^{n} c_i k(x_i, x)$$

$$Kc = y$$

$$K_{ij} = k(x_i, x_j)$$

Overfitting with kernels

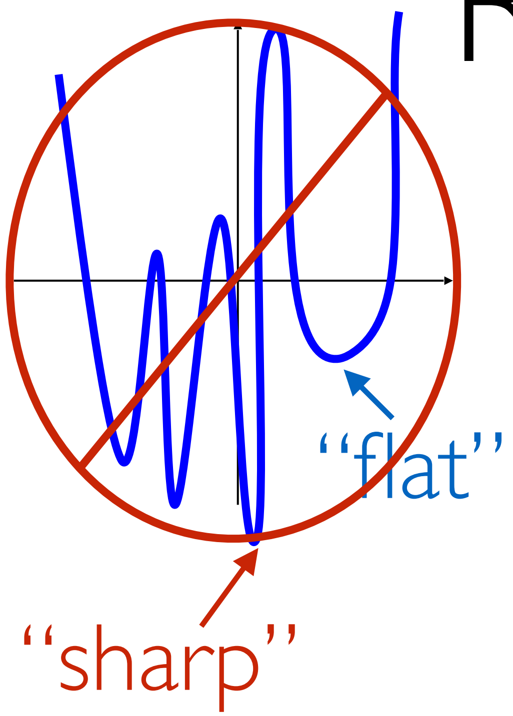
Procedure:

- Fit $Kc = y$ where K is the Gaussian kernel
  - 60k x 60k solve takes under 3 minutes with 24 cores

data set	pre-processing	test error
MNIST	none	1.2%
MNIST	gabor filters	0.6%
CIFAR10	none	46%
CIFAR10	1-layer conv-net, 32K random filters	16%

+L2 regularization gets this to 14%

Resolving "flat" vs "sharp"



minimize $\sum_{i=1}^{n} (w^T x_i - y_i)^2$

When p>n, all local minima have the same curvature.

"flat minimizers?"

minimize $||w||$
subject to $Xw = y$

$||w||^{-1}$ is the *margin* of the classifier

Small norm ⇒ loss is stable to perturbations in parameters
"flat minimizer"

Large norm ⇒ loss fluctuates with small perturbations to parameters
"sharp minimizer"

these ideas apply to deep nets

Challenge: get reasonable bounds.

…margin all over again

- In statistical learning, when all population points are classified correctly, one can show

$$\mathbb{E}[\text{test error}] \leq 4\frac{\|f_\star\|_k}{\sqrt{n}}$$

Inverse margin divided by $\sqrt{n}$

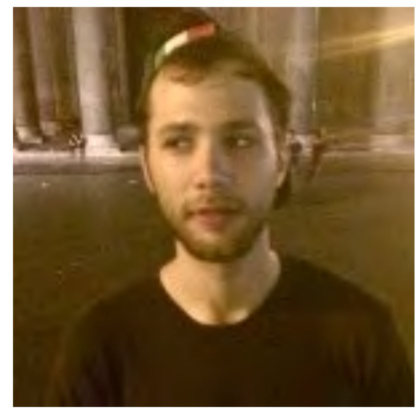
Challenge: find comparable, reasonable margin bounds for deep learning that explain experimental phenomena.

What can deep learning learn from linear regression?

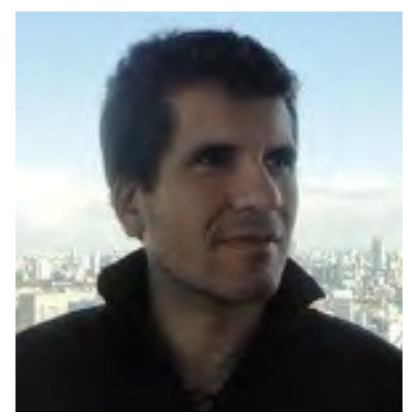
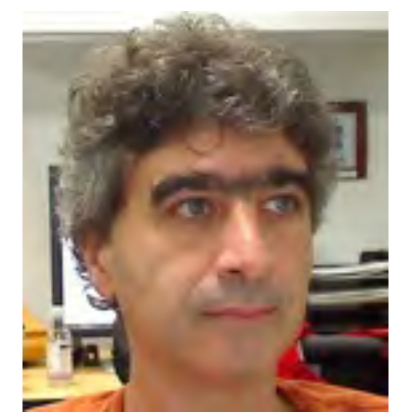
- regularization complicates optimization
- saddle points might not be an issue
- interpolation need not mean overfitting
- large margin classification is a great idea!
- stable algorithms lead to stable models

Stability and robustness are critical for guaranteeing safe, reliable performance of machine learning

Acknowledgments



- Joint work with Samy Bengio, Moritz Hardt, Michael Jordan, Jason Lee, Max Simchowitz, Oriol Vinyals, and Chiyuan Zhang.

Thanks!

References

- argmin.net
- "Gradient Descent Converges to Minimizers." Jason D. Lee, Max Simchowitz, Michael I. Jordan, and Benjamin Recht. COLT 2016, arXiv: 1602.04915
- "Understanding Deep Learning Requires Rethinking Generalization." Samy Bengio, Moritz Hardt, Benjamin Recht, Oriol Vinyals, and Chiyuan Zhang. ICLR 2017. arXiv:1611.03530
- Lecutre Notes on Approximation Algorithms and Semidefinite Programming. Bernd Gärtner and Jiří Matoušek. 2009. http://www.ti.inf.ethz.ch/ew/lehre/ApproxSDP09/index.html
- "A Nearly Tight Sum-of-Squares Lower Bound for the Planted Clique Problem." Boaz Barak, Samuel B. Hopkins, Jonathan Kelner, Pravesh K. Kothari, Ankur Moitra, Aaron Potechin. arXiv:1604.03084