Superpolynomial lower bounds on learning 1-layer neural nets with gradient descent

ICML 2020

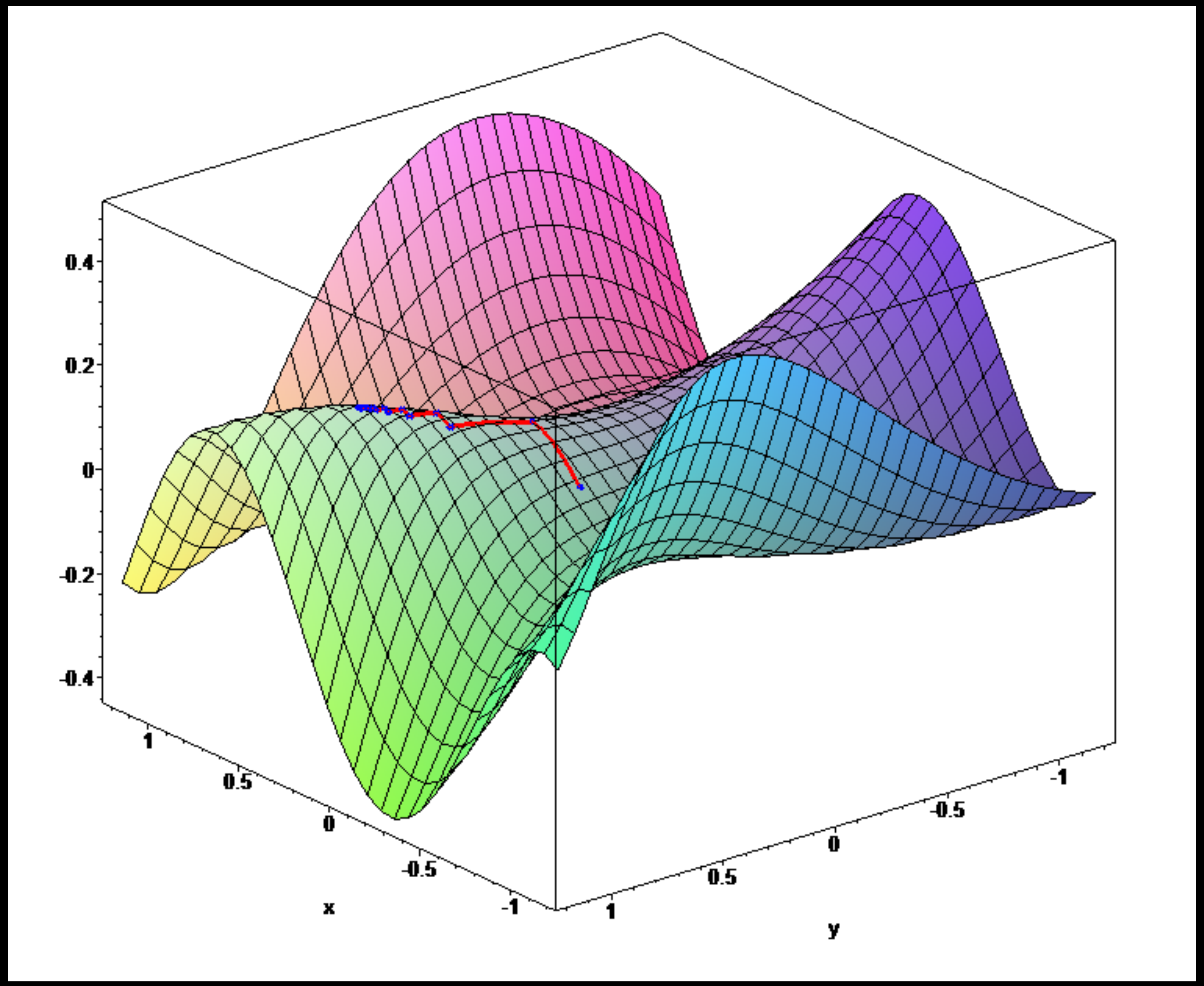
Joint work with Surbhi Goel, Zhihan Jin, Sushrut Karmalkar, and Adam Klivans

Aravind Gollakota, June 2020

University of Texas at Austin

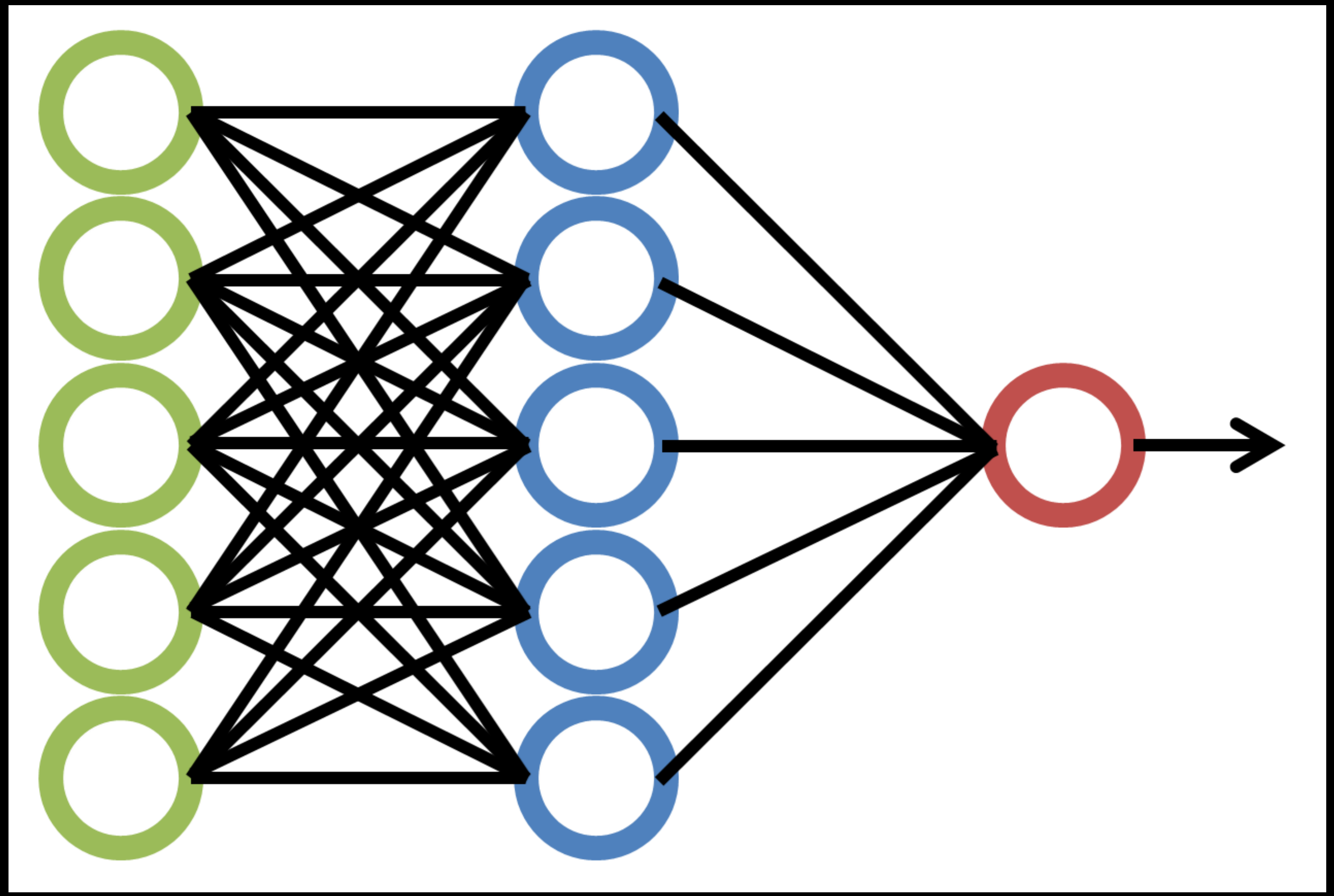
Training neural networks using gradient descent

- Have labeled training data (x, y)
- Want to train a neural network $f_\theta(x)$
- Define loss $L(\theta) = \mathbb{E}\left[(f_\theta(x) - y)^2\right]$
- Minimize loss using gradient descent:
  $\theta \leftarrow \theta - \eta \nabla L(\theta)$



The realizable, Gaussian setting

- $y = g(x)$, where g is an unknown 1-hidden-layer NN
  - With ReLU or sigmoid activations
- x is distributed according to Gaussian $N(0, I)$



Our main result:

even in this simple setting, GD could fail to converge in a polynomial number of steps

Our approach

- We model gradient descent as a *statistical query (SQ)* algorithm
- We construct a *hard class* of 1-layer neural nets
- We show, unconditionally, that no SQ algorithm can learn this hard class in a *polynomial number of queries*

The statistical query model

- Have a distribution D on $\mathbb{R}^n \times \mathbb{R}$, i.e. on labeled pairs (x, y)
- Don't see individual points (x, y), instead make "statistical queries" to an oracle

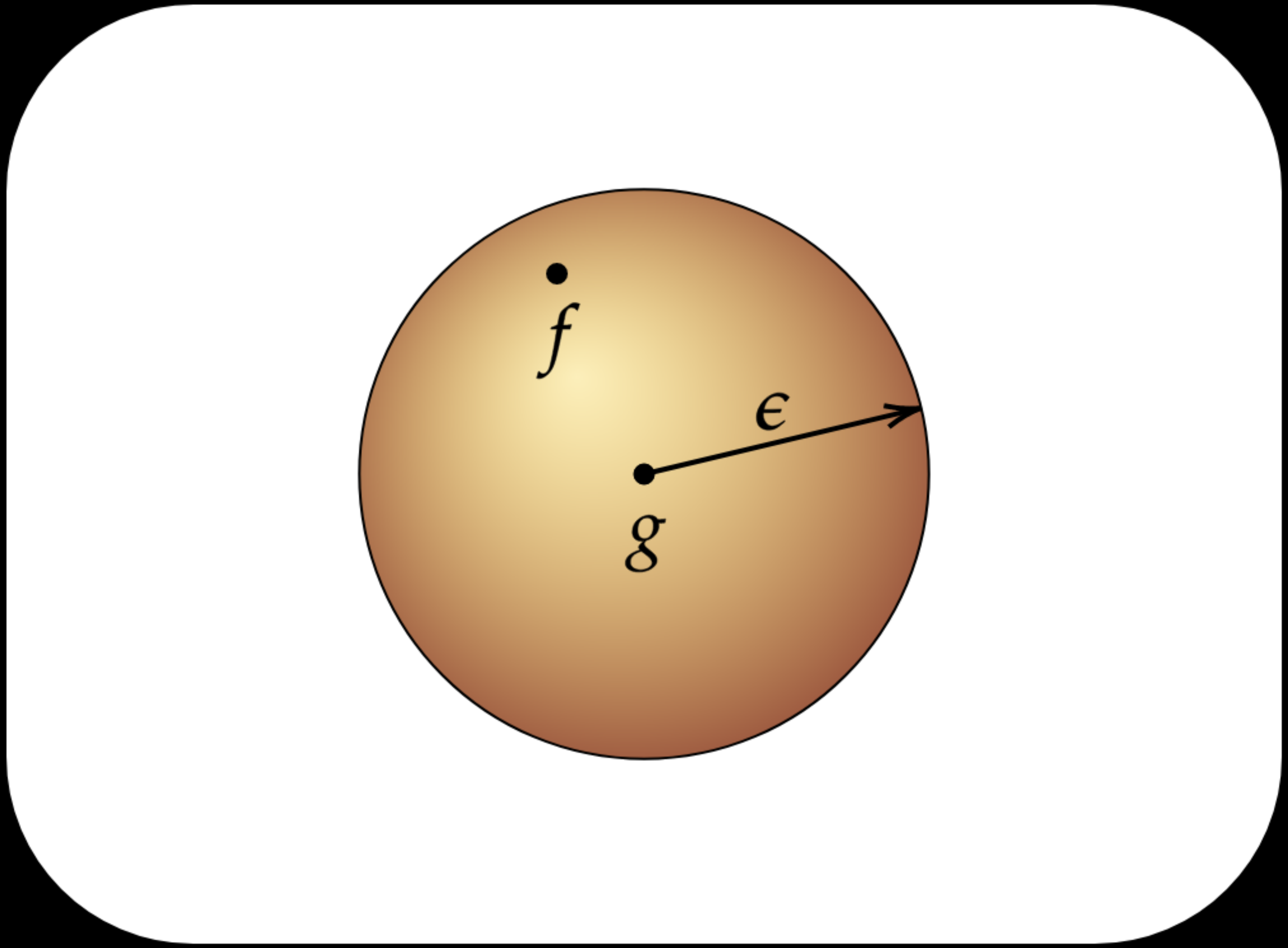
$$\varphi : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$$

$$\tau \in [0,1]$$

$$\mathbb{E}[\varphi(x, y)] \pm \tau$$

Statistical query learning

- Unknown function g in a known class
- Let D_g denote the distribution of $(x, g(x))$ for $x \sim N(0, I)$
- You have SQ oracle access to D_g
- Want to output f that is ϵ-close to g



Gradient descent as an SQ algorithm

- Say our current model is $f_\theta(x)$, with parameters θ
- Consider population squared loss: $L(\theta) = \mathbb{E}\left[(f_\theta(x) - y)^2\right]$
- Its gradient is $\nabla L(\theta) = \mathbb{E}\left[\nabla_\theta (f_\theta(x) - y)^2\right]$
- Each coordinate turns out to be a statistical query
- In fact, each query is (essentially) *correlational*, i.e. of the form $\varphi(x, y) = h(x)y$

How does one prove SQ lower bounds?

- The *SQ dimension* of a function class measures its SQ complexity
    - Similar in spirit to VC dimension
- Can roughly think of as the *number of uncorrelated functions* in the class
    - Here the correlation of two functions f, g is $\mathbb{E}[f(x)g(x)]$
- Well-studied

Construction of the hard class

Input dimension n, number of hidden units k

Input: $x \in \mathbb{R}^n$

Activation function $\phi : \mathbb{R} \to \mathbb{R}$

$$g_S(x) = \sum_{w \in \{-1,1\}^{\log k}} \chi(w)\ \phi(w \cdot x_S)$$

S: set of $\log k$ indices

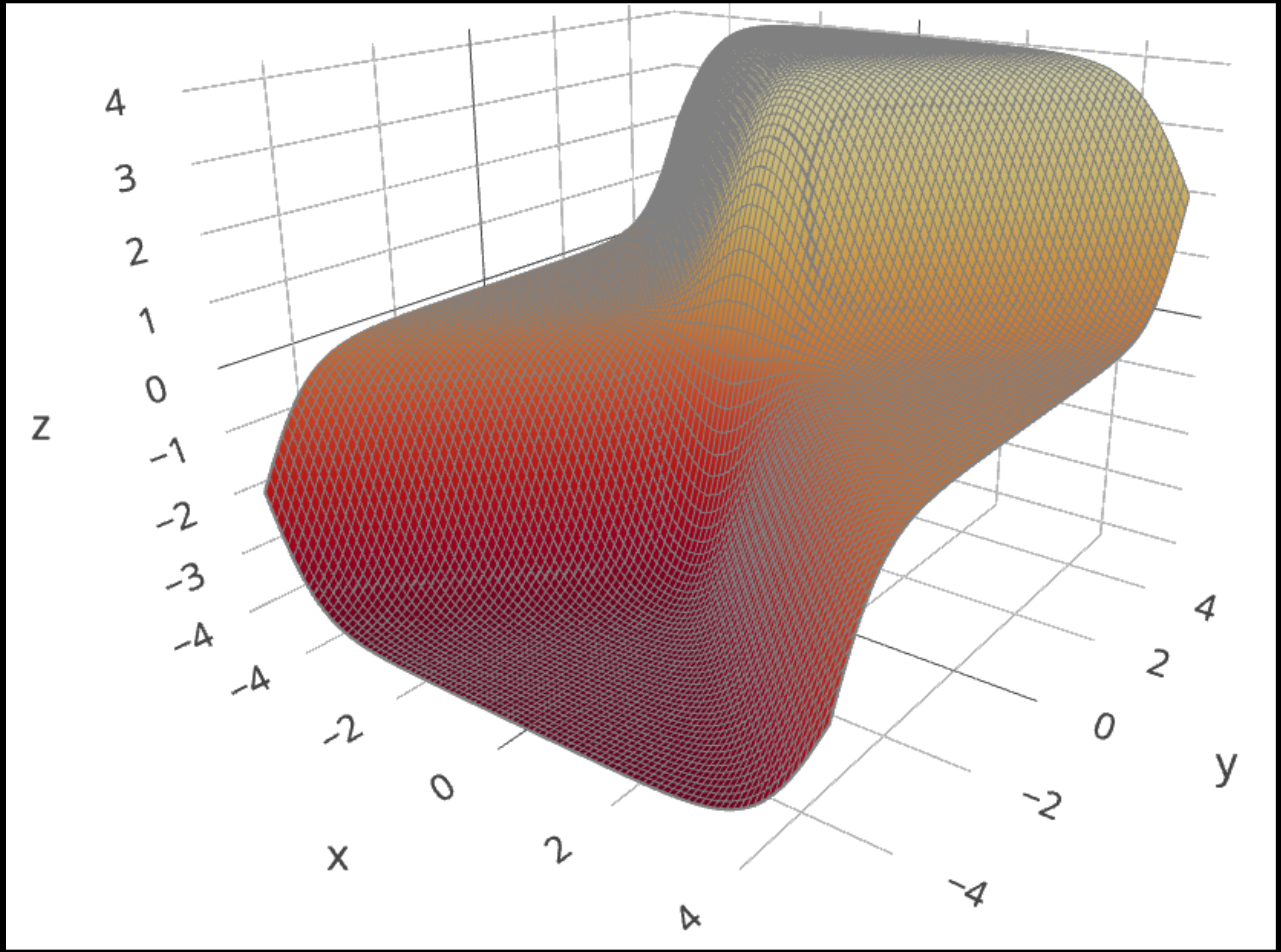
k units

$\chi(w) = w_1 w_2 \cdots w_{\log k}$

x_S: coordinates of x in S

A visualization

In 3 dimensions, with $\phi = \tanh$



These functions are uncorrelated

- For any two index sets S and T, g_S and g_T are completely uncorrelated, i.e. $\mathbb{E}\left[g_S(x)g_T(x)\right] = 0$
    - This holds under *any* spherically symmetric distribution!

SQ dimension of our construction

- Number of hidden units: $2^{\log k} = k$
- Obtain $\binom{n}{\log k} \approx n^{\Theta(\log k)}$ uncorrelated functions, one for each index set S
- SQ dimension is roughly $n^{\Theta(\log k)}$

The formal lower bound

- To learn this hard class up to error $\epsilon < 1/\text{poly}(k)$, even using tolerance $\tau = n^{-\Theta(\log k)}$, any SQ algorithm requires at least $n^{\Theta(\log k)}$ correlational queries.
- In particular, gradient descent with respect to squared loss requires at least $n^{\Theta(\log k)}$ steps.
- Technical subtlety: functions must be noticeably far from zero.
    - We show this using tools from Hermite analysis

Related work

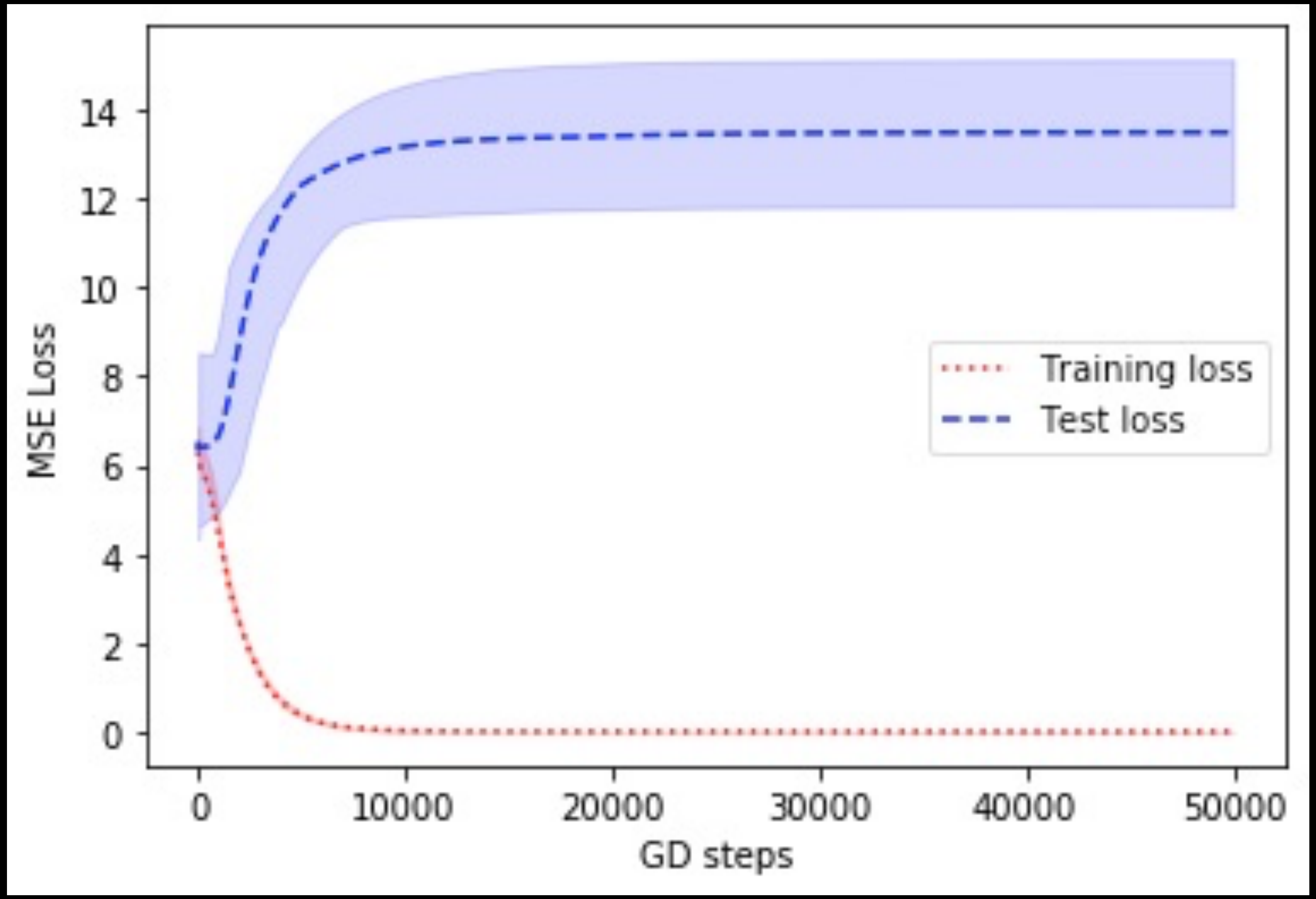
- Le Song, Santosh Vempala, John Wilmes, and Bo Xie, NeurIPS 2017
- Santosh Vempala and John Wilmes, COLT 2019
- Ohad Shamir, JMLR 2018, COLT 2019
- Shai Shalev-Shwartz, Ohad Shamir, and Shaked Shammah, ICML 2017
- *Concurrent:* Ilias Diakonikolas, Daniel Kane, Vasilis Kontonis, and Nikos Zarifis, COLT 2020

Extension to probabilistic concepts

- Boolean labels obtained by interpreting output as a probability
- For input x, say we see label $y = 0$ with probability $\sigma(g_S(x))$ and $y = 1$ otherwise
- Our lower bound extends to this setting as well
  - In fact for *general* (not just correlational) queries

Experiments

- Trained an *overparameterized* NN on data from our hard class using GD on squared loss
- Random initialization
- Input dimension: $n = 14$
- Labels: sum of $k = 512$ tanh units



Summary

- We show new superpolynomial SQ lower bounds on learning simple 1-layer neural networks
- Works under the Gaussian distribution, and with standard activations
- Extends to probabilistic Boolean labels

Thanks!