CS 229br: Advanced Topics in the theory of machine learning

Boaz Barak

Ankur Moitra
MIT 18.408

Yamini Bansal
Official TF

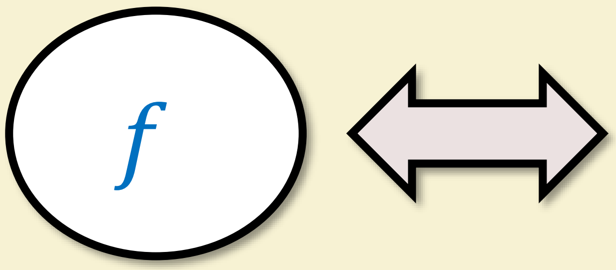
Dimitris Kalimeris
Unofficial TF

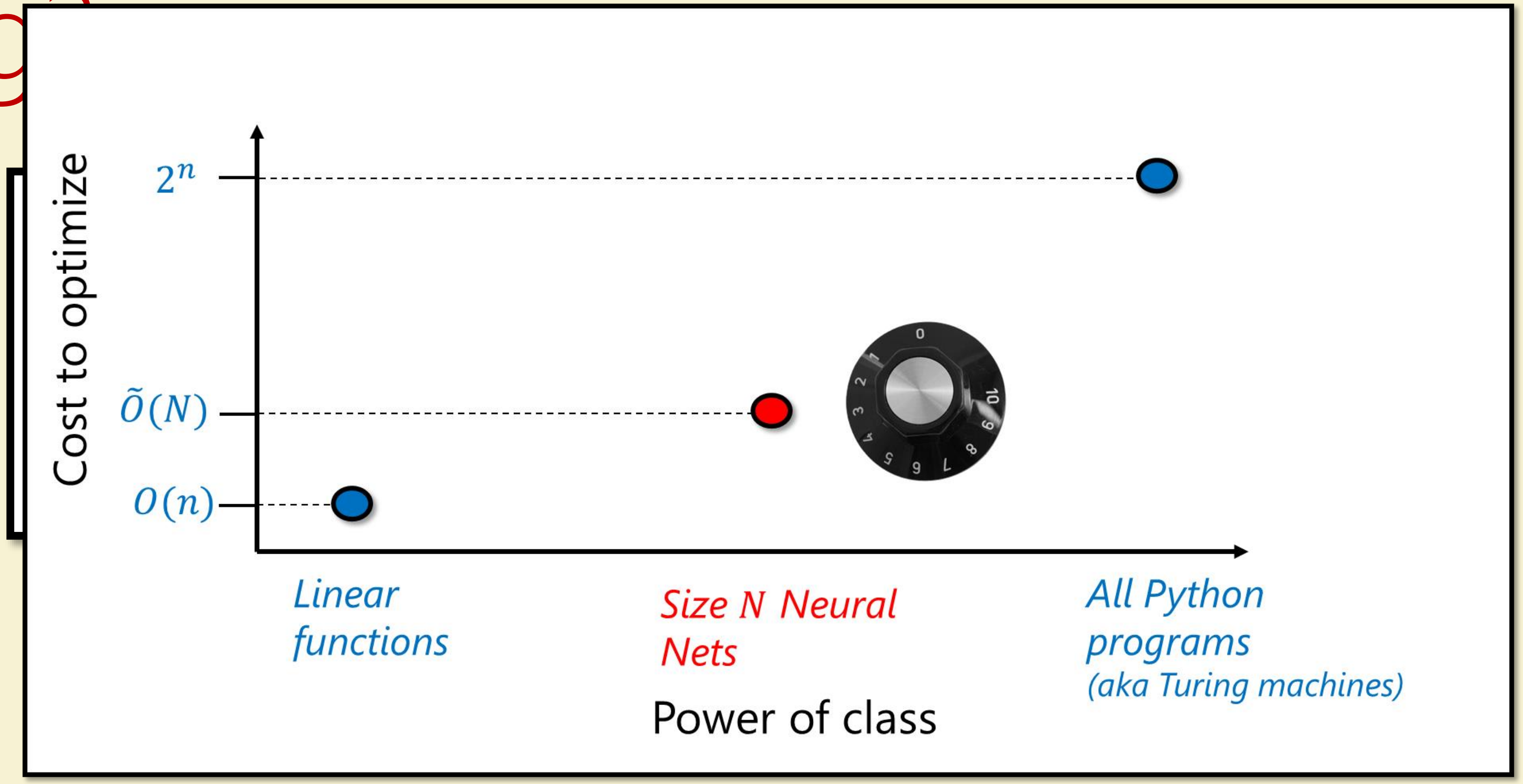
Gal Kaplun
Unofficial TF

Preetum Nakkiran
Unofficial TF

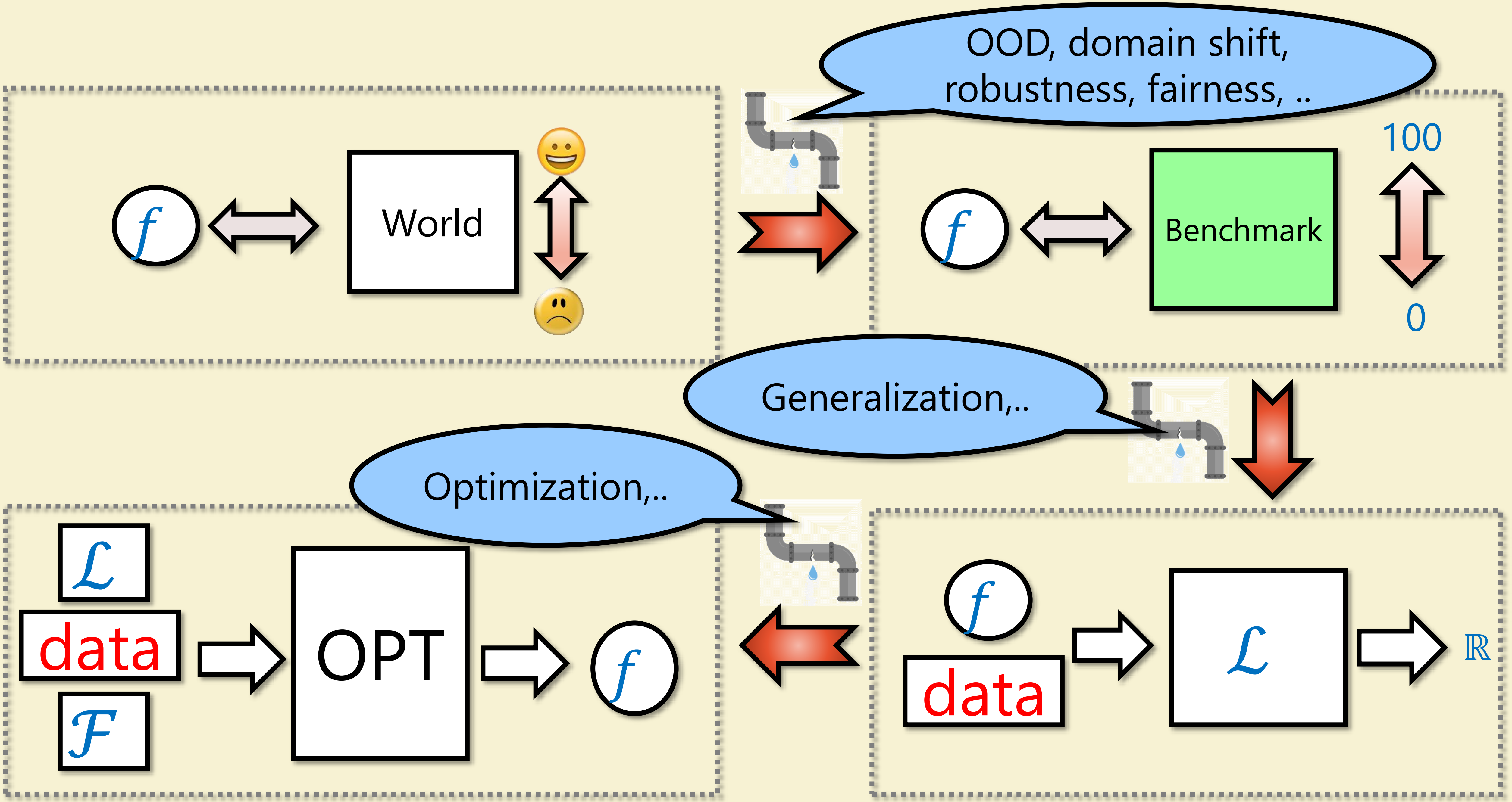
Join slack team and sign in on your devices!

What is learning?

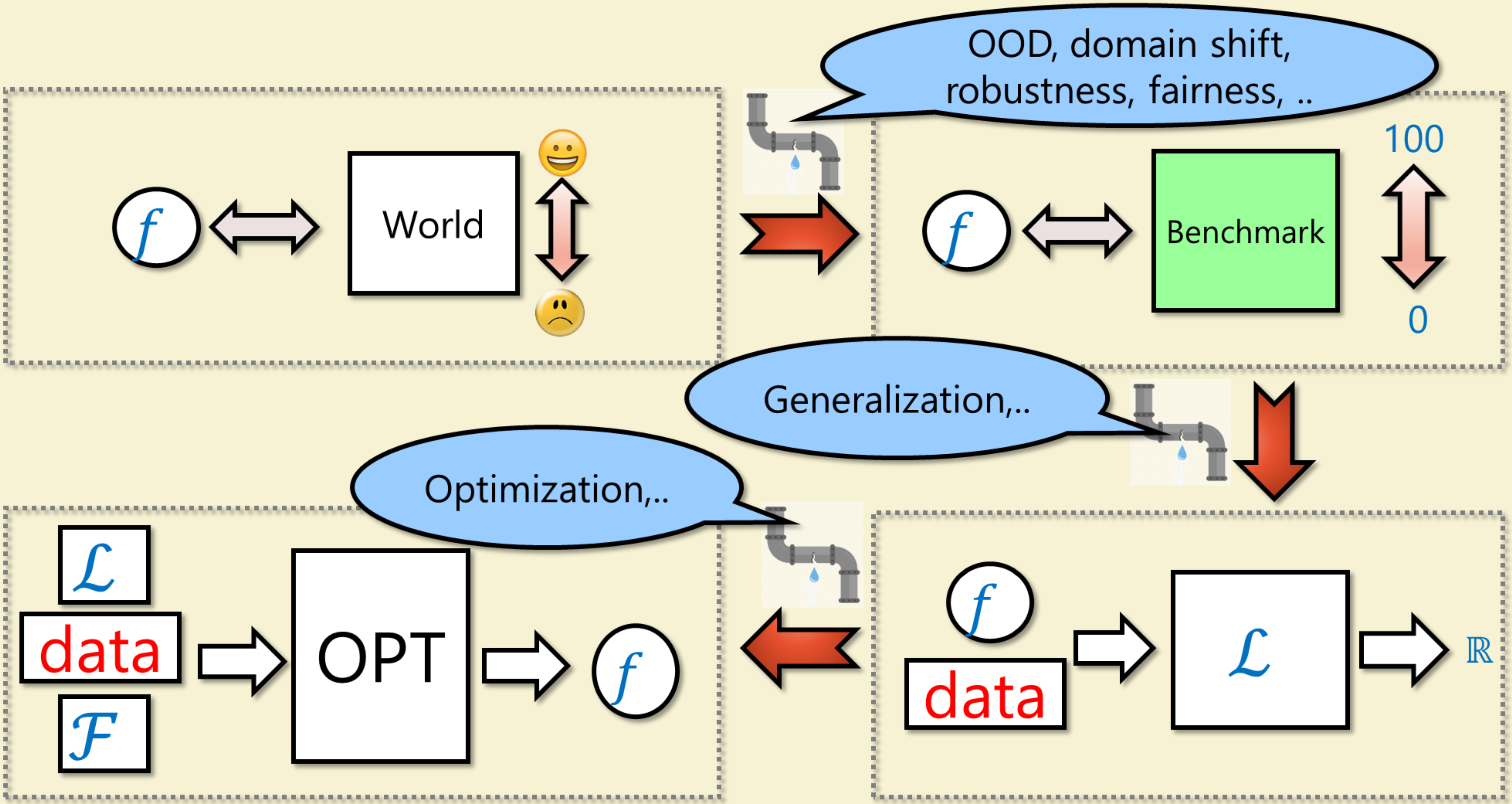




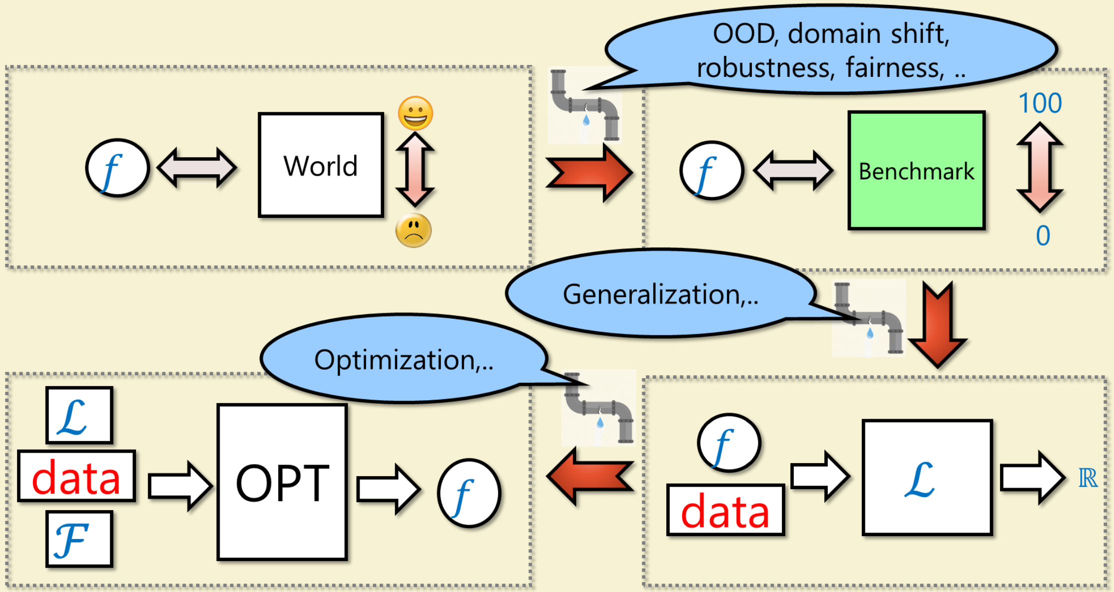
What is deep learning?



OOD, domain shift,
robustness, fairness, ..
f
World
f
Benchmark
100
0
Generalization,..
Optimization,..
$\mathcal{L}$
data
$\mathcal{F}$
OPT
f
f
data
$\mathcal{L}$
$\mathbb{R}$



OOD, domain shift, robustness, fairness, ..
f
World
f
Benchmark
100
0
Generalization,..
Optimization,..
$\mathcal{L}$
data
$\mathcal{F}$
OPT
f
f
data
$\mathcal{L}$
$\mathbb{R}$



God's view:

$$f = \mathrm{argmax}_f\ \mathbb{E}[\ 😀(f)\ \mid \mathrm{data}\ ]$$

$$f = \mathrm{argmax}_f\ \mathbb{E}[\ 😀(f \leftrightarrow w)\ ]$$
$$w \sim W|\mathrm{data}$$

Perfect A $\Rightarrow$ Perfect B $\neq$ Approx A $\Rightarrow$ Approx B

This seminar

- Taste of research results, questions, experiments, and more
- Goal: Get to state of art research:
  - Background and language
  - Reading and reproducing papers
  - Trying out extensions
- Very experimental and "rough around the edges"
- A lot of learning on your own and from each other
- Hope: Very interactive – in lectures and on slack

CS 229br: Survey of very recent research directions, emphasis on experiments

MIT 18.408: Deeper coverage of foundations, emphasis on theory

Student expectations

Not set in stone but will include:

- Pre-reading before lectures
- Scribe notes / blog posts (adding proofs and details)
- Applied problem sets (replicating papers or mini versions thereof, exploring)
  Note: *Lectures will not teach practical skills – rely on students to pick up using suggested tutorials, other resources, and each other.*
  Unofficial TFs happy to answer questions!
- Some theoretical problem sets.
- Might have you grade each other's work
- Projects – self chosen and directed.

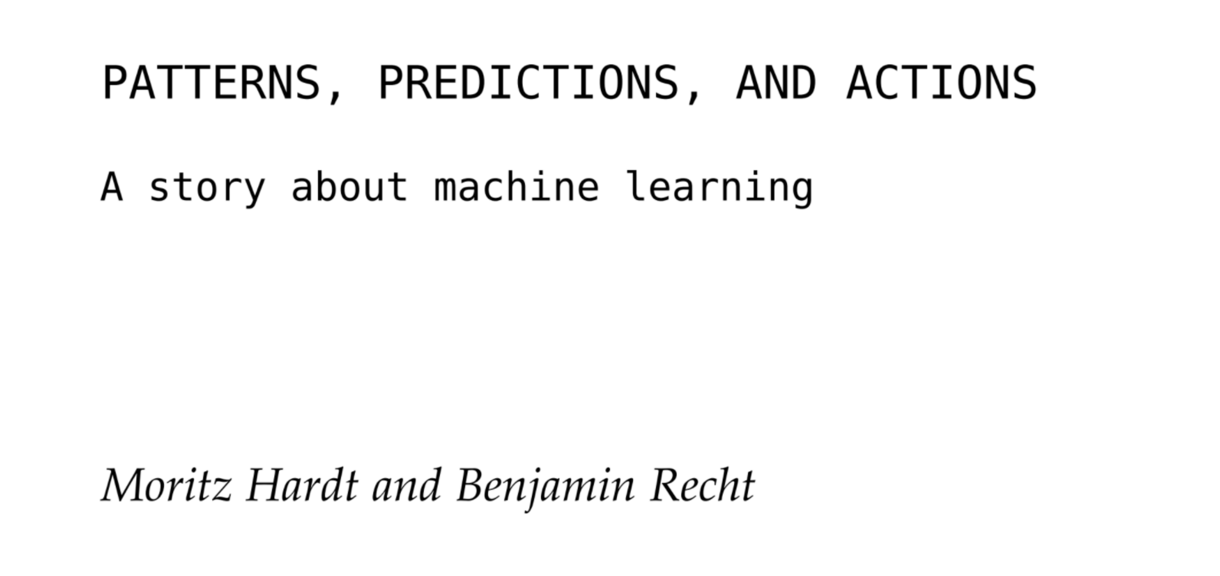
HW0 on slack

Grading: We'll figure out some grade – hope that's not your loss function ☺

Rest of today's lecture:

Blitz through classical learning theory

- Special cases, mostly one dimension
- "Proofs by picture"


PATTERNS, PREDICTIONS, AND ACTIONS

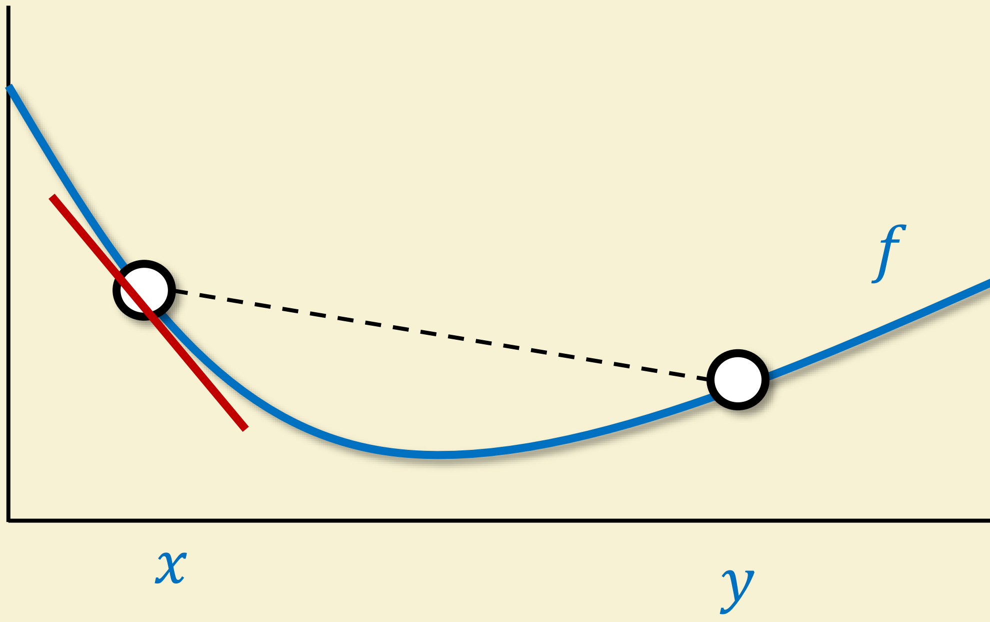
A story about machine learning

Moritz Hardt and Benjamin Recht

Part I: Convexity & Optimization

Convexity

1. Line between $(x, f(x))$ and $(y, f(y))$ above f.
2. Tangent line at x (slope $f'(x)$) below f.
3. $f''(x) > 0$ for all x

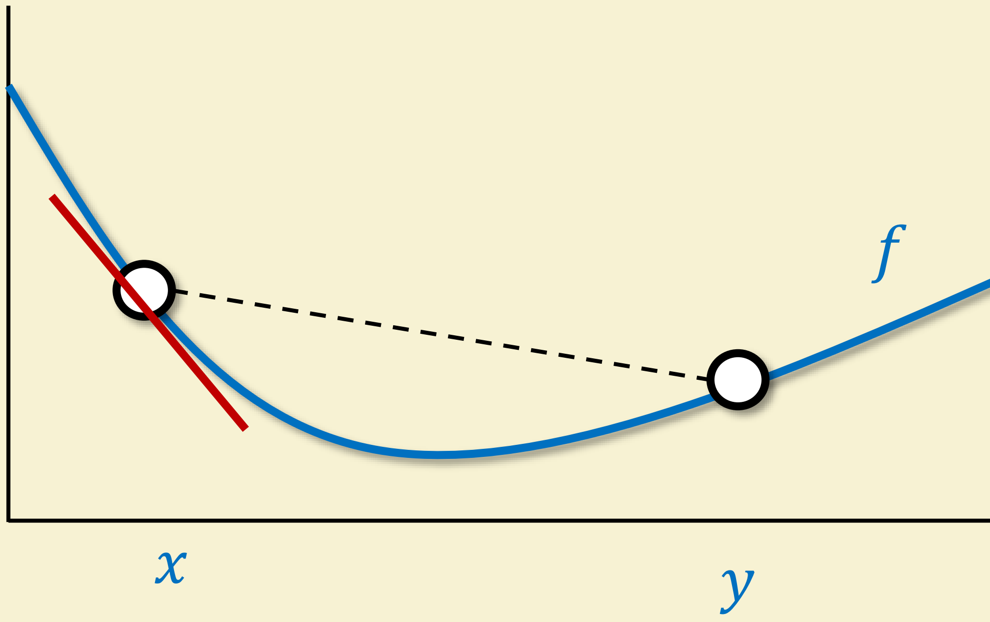


CLAIM: $f''(x) < 0$ implies tangent line at x above f

Proof: $f(x+\delta) = f(x) + \delta f'(x) + \delta^2 f''(x) + O(\delta^3)$ implies f below line

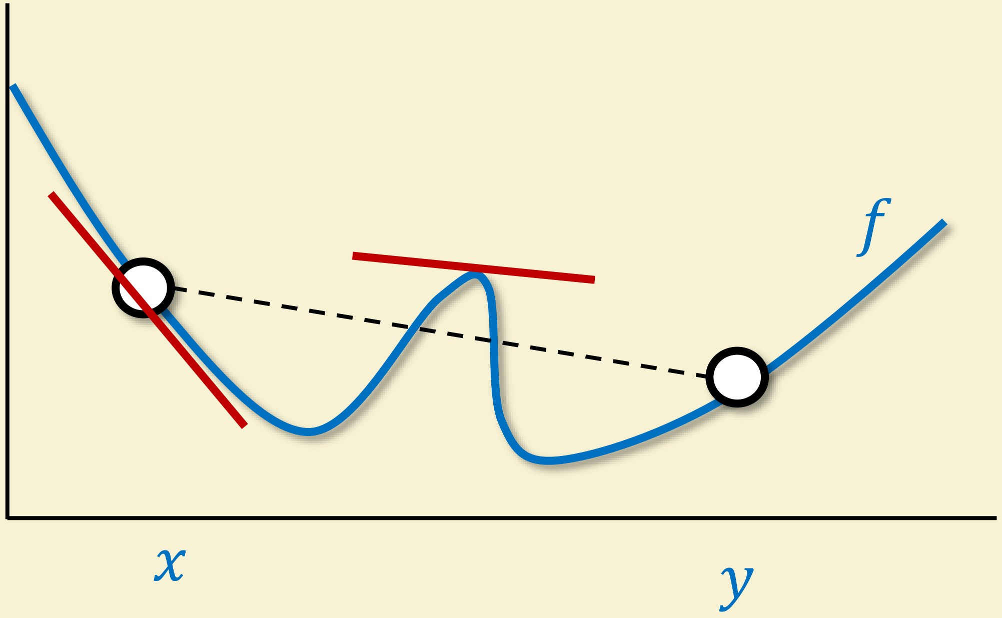
Convexity

1. Line between $(\mathrm{x}, f(x))$ and $(y, f(y))$ above f.
2. Tangent line at x (slope $f'(x)$) below f.
3. $f''(x) > 0$ for all x



CLAIM: If f above $(\mathrm{x}, f(x)) - (y, f(y))$ then exists z with tangent line at z above f

Proof by picture:



Gradient Descent

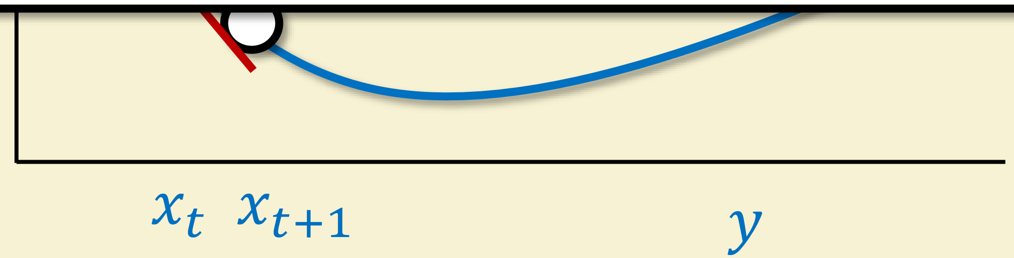
Dimension d:

$f'(x) \to \nabla f(x) \in \mathbb{R}^d$

$f''(x) \to H_f(\mathrm{x}) = \nabla_2 f(x) \in \mathbb{R}^{d \times d}$ (psd)

$\eta \sim 1/\lambda_d$ drop by $\sim \frac{\lambda_1}{\lambda_d} \|\nabla\|^2$

$$x_{t+1} = x_t - \eta f'(x_t)$$



$$f(x_t + \delta) \approx f(x_t) + \delta f'(x_t) + \frac{\delta^2}{2} f''(x_t)$$

$$f(x_{t+1}) \approx f(x_t) - \eta f'(x_t)^2 + \frac{\eta^2 f'(x_t)^2}{2} f''(x_t) \quad = f(x_t) - \eta f'(x_t)^2 \left(1 - \frac{\eta f''(x_t)}{2}\right)$$

- If $\eta < 2/f''(x_t)$ then make progress
- If $\eta \sim 2/f''(x_t)$ then drop by $\sim \eta f'(x_t)^2$

Stochastic Gradient Descent

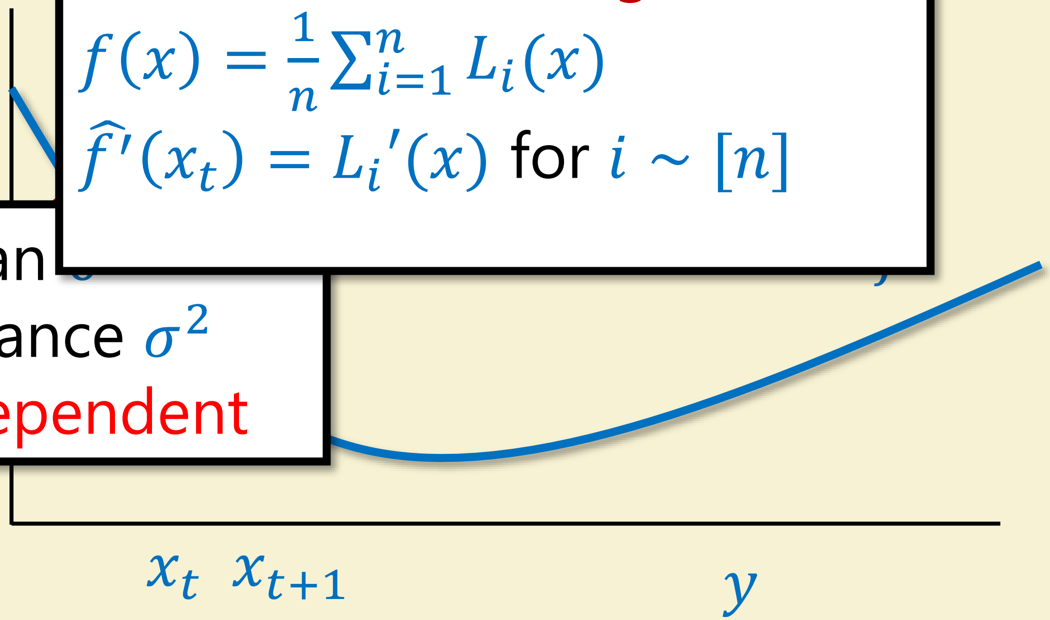
$x_{t+1} = x_t - \eta \widehat{f}'(x_t)$

$\mathbb{E}[\widehat{f}'(x)] = f'(x_t)$, $V[\widehat{f}'(x)] = \sigma^2$

Assume $\widehat{f}'(x) = f'(x) + N$

Mean
Variance σ^2
Independent

In Machine Learning:
$f(x) = \frac{1}{n}\sum_{i=1}^{n} L_i(x)$
$\widehat{f}'(x_t) = L_i{}'(x)$ for $i \sim [n]$



$$f(x_t + \delta) \approx f(x_t) + \delta f'(x_t) + \frac{\delta^2}{2} f''(x_t)$$

$$f(x_{t+1}) \approx f(x_t) - \eta f'(x_t)^2 \left(1 - \frac{\eta f''(x_t)}{2}\right) + \eta^2 \sigma^2 f''(x_t)$$

- If $\eta < 2/f''(x_t)$ and (*) $\eta\sigma^2 \ll f'(x_t)^2/f''(x)$ then make progress
- If $\eta \sim 2/f''(x_t)$ and (*) then drop by $\sim \eta f'(x_t)^2$

Part II: Generalization

Supervised learning

Empirical Risk Minimization (ERM):
$A(S) = \arg\min_{f \in \mathcal{F}} \hat{\mathcal{L}}_S(f)$

Distribution (X, Y) over $\mathcal{X} \times \{\pm 1\}$

$S = (x_i, y_i)_{i=1..n}$

data $\Rightarrow$ Learning Algorithm A $\Rightarrow$ $f \in \mathcal{F}$

$\mathcal{L}(f) = \Pr[f(X) \neq Y]$

Population 0-1 loss

$\hat{\mathcal{L}}_S(f) = \frac{1}{n} \sum_{i=1}^{n} 1_{f(x_i) \neq y_i}$

Empirical 0-1 loss

Generalization gap: $\mathcal{L}(f) - \hat{\mathcal{L}}_S(f)$ for $f = A(S)$

Bias Variance Tradeoff

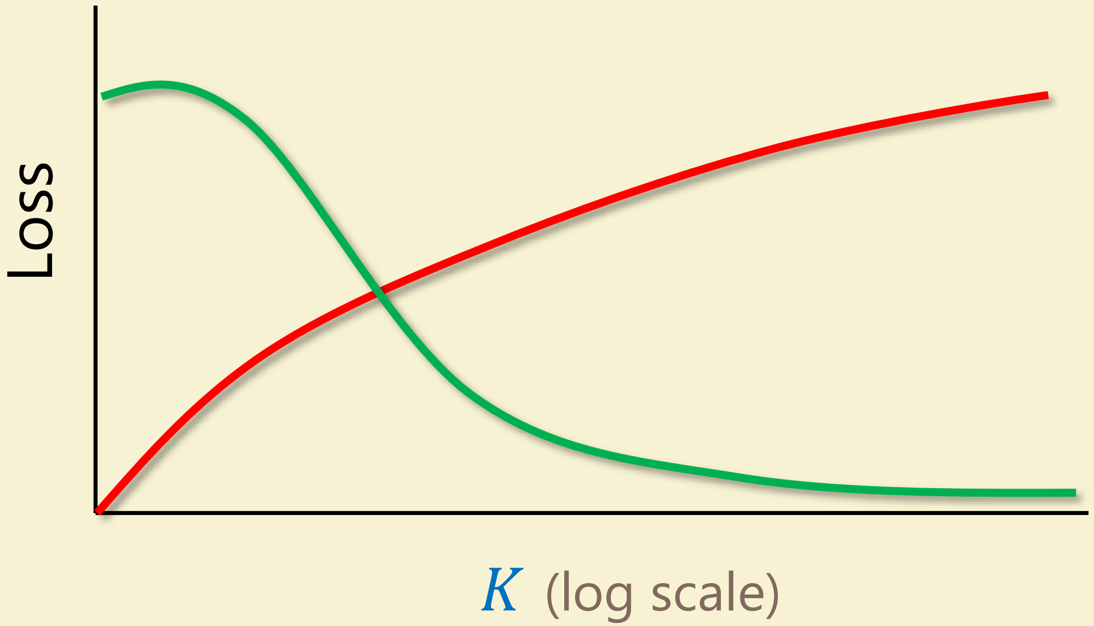
Empirical Risk Minimization (ERM):

$$A(S) = \arg\min_{f\in\mathcal{F}} \hat{\mathcal{L}}_S(f)$$

$S = (x_i, y_i)_{i=1..n}$

Population 0-1 loss $\mathcal{L}(f) = \Pr[f(X) \neq Y]$ bias Emp variance ss $\hat{\mathcal{L}}_S(f) = \frac{1}{n}\sum_{i=1}^{n} 1_{f(x_i)\neq y_i}$

Assume $\mathcal{F}_K = \{f_1, \dots, f_K\}$, $\hat{\mathcal{L}}(f_i) = \mathcal{L}(f_i) + N(0, 1/n)$



$\sim \sqrt{\frac{\log K}{n}}$?

$\sim (\log K)^{-\alpha}$??

Bias Variance Tradeoff

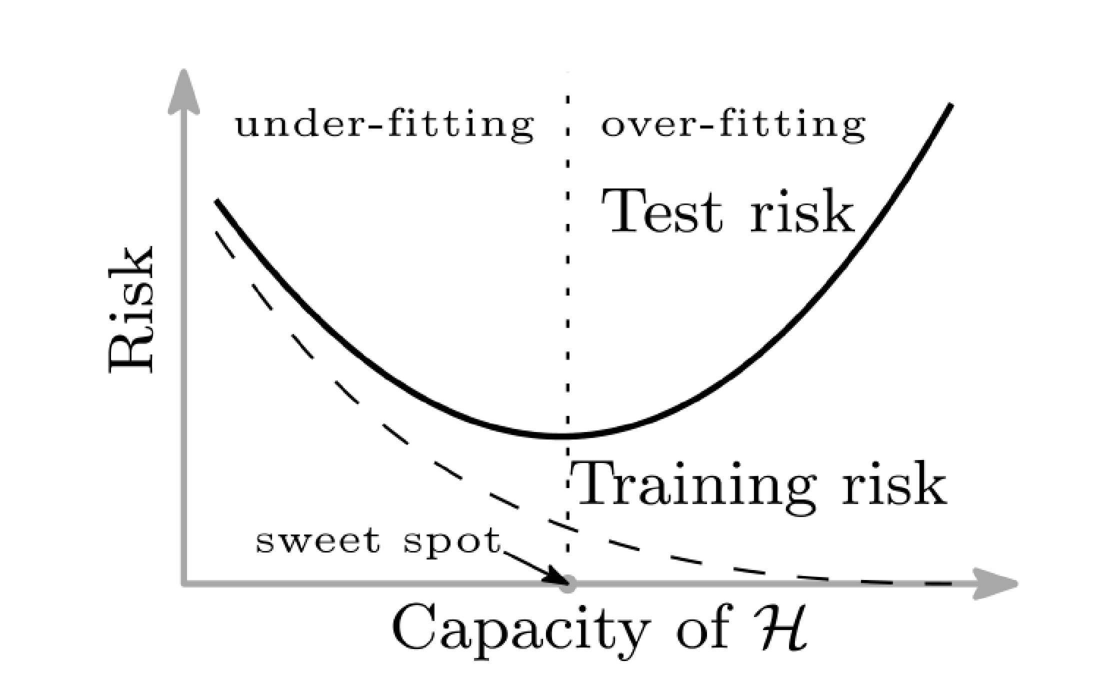
Empirical Risk Minimization (ERM):

$$A(S) = \arg\min_{f \in \mathcal{F}} \hat{\mathcal{L}}_S(f)$$

$S = (x_i, y_i)_{i=1..n}$ → Learning Algorithm A → $f \in \mathcal{F}$

Population 0-1 loss $\mathcal{L}(f) = \Pr[f(X) \neq Y]$ bias mp variance ss $\hat{\mathcal{L}}_S(f) = \sum_{i=1}^{n} 1_{f(x_i) \neq y_i}$

Assume $\mathcal{F}$ ... $) + N(0, 1/n)$



$\sim \sqrt{\frac{\log K}{n}}$?

$\sim (\log K)^{-\alpha}$??

Can prove:

GAP $\leq O\left(\sqrt{\frac{\log |\mathcal{F}|}{n}}\right)$

Population 0-1 loss $\mathcal{L}(f) = \Pr[f(X) \neq Y]$ Empirical 0-1 loss $\hat{\mathcal{L}}_S(f) = \sum_{i=1}^{n} 1_{f(x_i) \neq y_i}$

Can prove:

GAP $\leq O\left(\sqrt{\frac{\log |\mathcal{F}|}{n}}\right)$

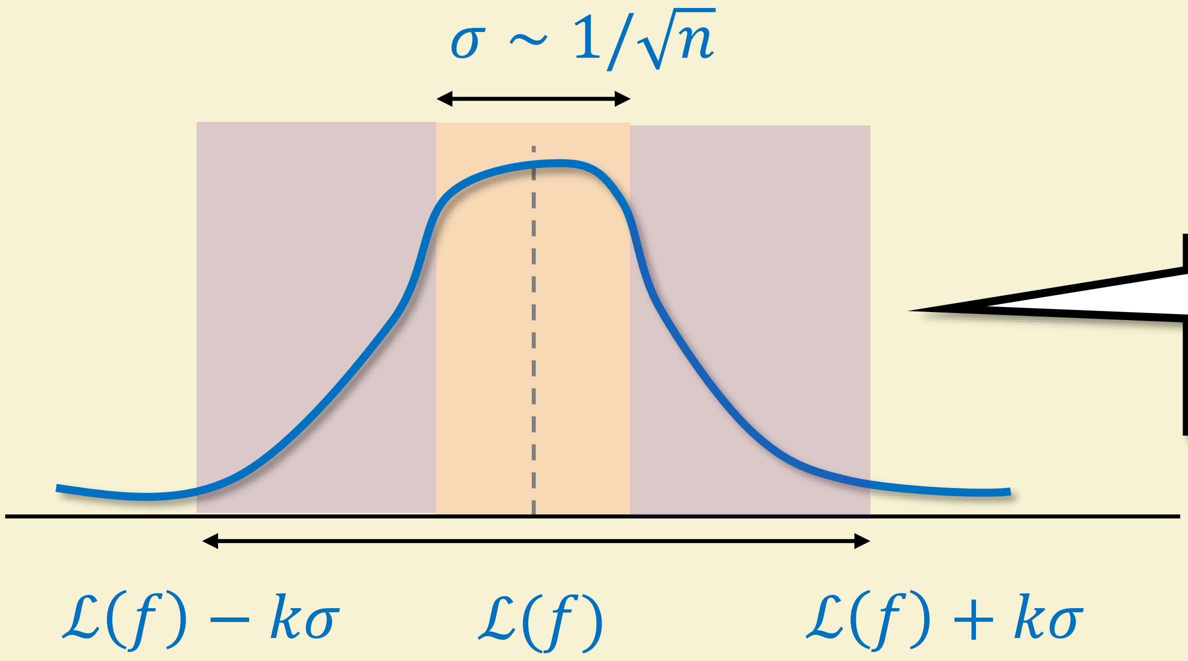
THM: with high prob over $S \sim (X, Y)^n$, $\max_{f \in \mathcal{F}} \left|\mathcal{L}(f) - \hat{\mathcal{L}}_S(f)\right| \leq O\left(\sqrt{\frac{\log|\mathcal{F}|}{n}}\right)$

Population 0-1 loss $\mathcal{L}(f) = \Pr[f(X) \neq Y]$ Empirical 0-1 loss $\hat{\mathcal{L}}_S(f) = \sum_{i=1}^{n} 1_{f(x_i) \neq y_i}$

THM: with high prob over $S \sim (X,Y)^n$, $\max_{f \in \mathcal{F}} \left| \mathcal{L}(f) - \hat{\mathcal{L}}_S(f) \right| \leq O\left(\sqrt{\frac{\log|\mathcal{F}|}{n}}\right)$

LEM: For every fixed f, $\Pr_S\left[\left|\mathcal{L}(f) - \hat{\mathcal{L}}_S(f)\right| \geq \frac{k}{\sqrt{n}}\right] \leq \exp(-c \cdot k^2)$

PF: Central limit thm / Chernoff / Hoeffding / ….



$\Pr[|X - \mathbb{E}X| \geq k\sigma] \leq \exp(-c \cdot k^2)$

LEM⇒THM: Set $k = 10 \cdot \sqrt{\log|\mathcal{F}|}/c$

Generalization bounds

Can prove:

$$\text{GAP} \leq O\left(\sqrt{\frac{\log|\mathcal{F}|}{n}}\right)$$

$\log|\mathcal{F}|$ can be replaced with dimensionality / capacity of $\mathcal{F}$

VC dimension:
$\max d$ s.t. $\mathcal{F}$ can fit every labels $y \in \{\pm 1\}^d$ on every samples $x_1, \ldots, x_d$

Rademacher complexity:
$\max d$ s.t. $\mathcal{F}$ can fit random labels $y \sim \{\pm 1\}^d$ on random $x_1, \ldots, x_d \sim X$

PAC Bayes:
$d = I(A(S); S)$ where S is training set.

Margin bounds:
$\max d$ s.t. linear classifier satisfies $\langle w_i, x_i \rangle y_i > \|w_i\| \cdot \|x_i\| / \sqrt{d}$ for correct predictions

General form: If $C(f) \ll n$ then $\hat{\mathcal{L}}(f) - \mathcal{L}(f) \approx 0$

Intuition: Can't "overfit" – if you do well on n samples, must do well on population

Challenge: Many modern deep nets and natural C , $C(f) \gg n$

Hope: Find better C?

UNDERSTANDING DEEP LEARNING REQUIRES RE-THINKING GENERALIZATION

Chiyuan Zhang*
Massachusetts Institute of Technology
chiyuan@mit.edu

Samy Bengio
Google Brain
bengio@google.com

Moritz Hardt
Google Brain
mrtz@google.com

Benjamin Recht†
University of California, Berkeley
brecht@berkeley.edu

Oriol Vinyals
Google DeepMind
vinyals@google.com

ICLR 2017

General form: If $C(f) \ll n$ then $\hat{\mathcal{L}}(f) - \mathcal{L}(f) \approx 0$

Intuition: Can't "overfit" – if you do well on n samples, must do well on population

Challenge: Many modern deep nets and natural C , $C(f) \gg n$

Hope: Find better C?

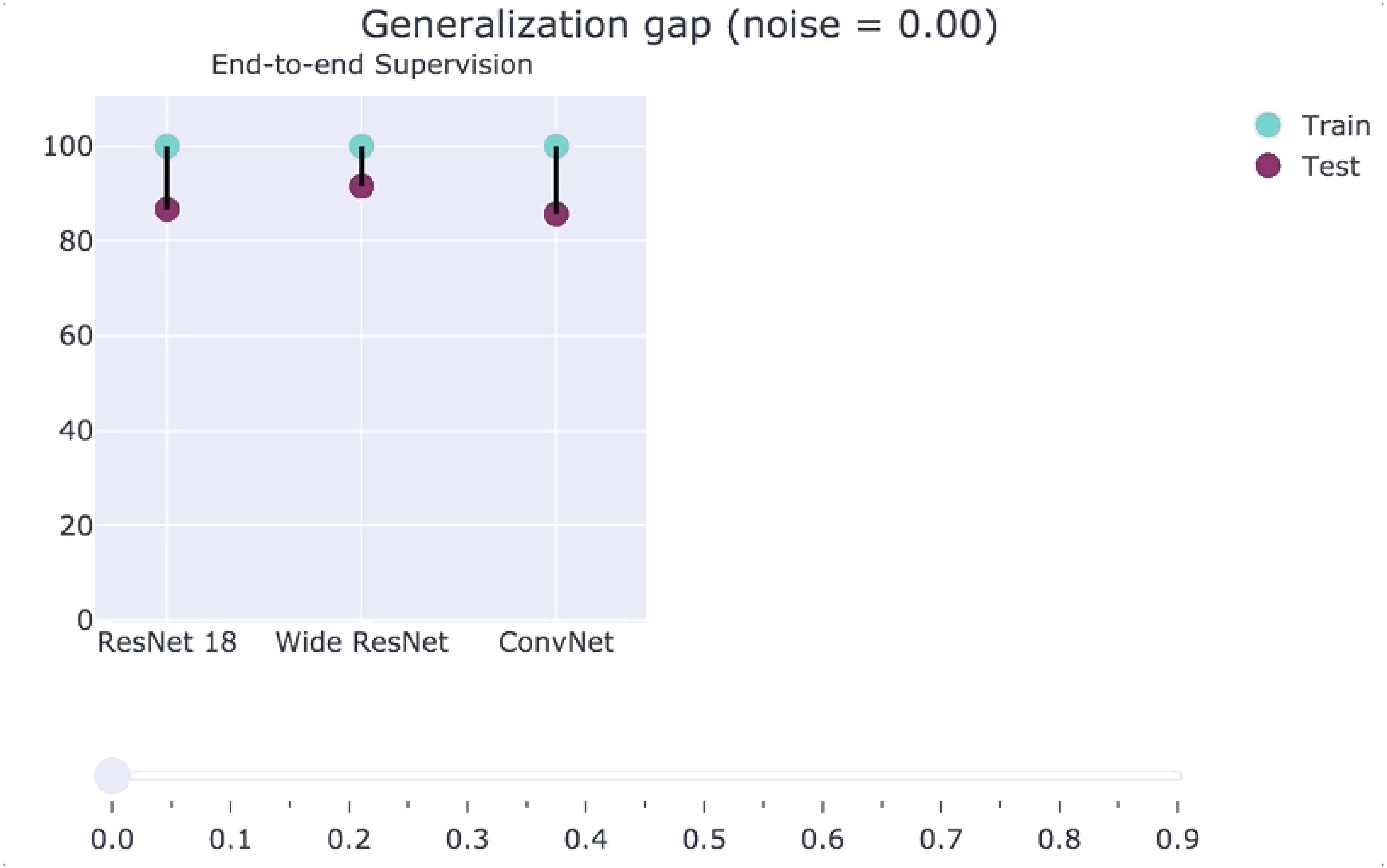
Performance on CIFAR-10

Classifier	Train	Test
Wide ResNet[1]	100%	92%
ResNet 18[2]	100%	87%
Myrtle[3]	100%	86%

Performance on noisy CIFAR-10:*

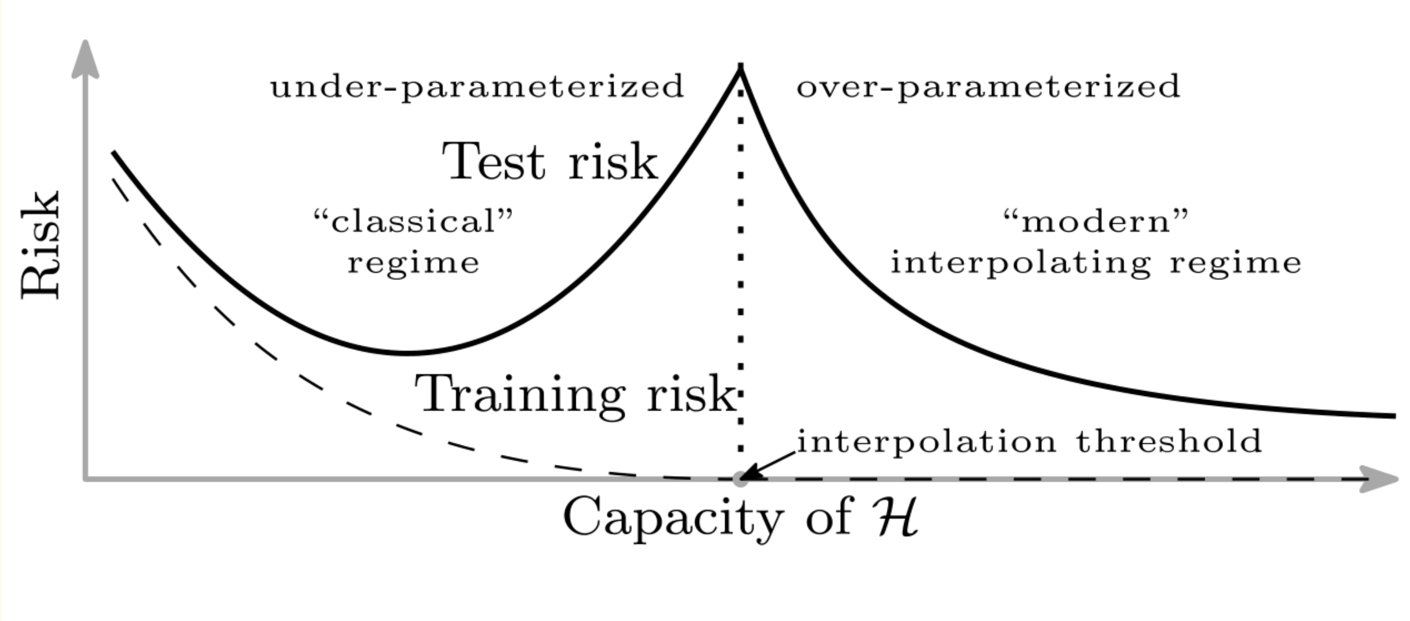
Train	Test
100%	10%
100%	10%
100%	10%

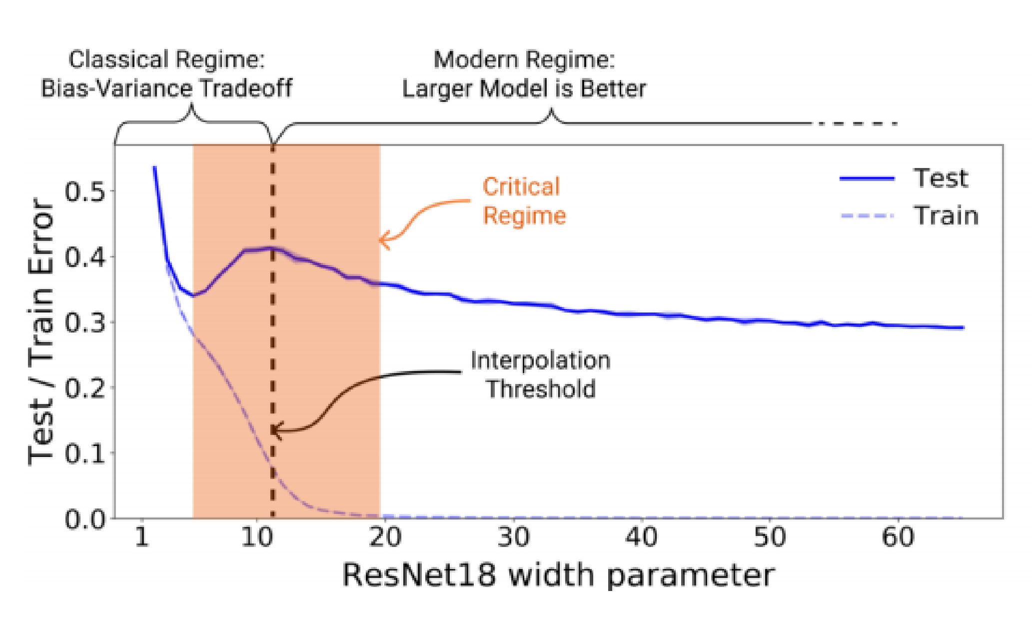
[1] Zagoruyko-Komodakis'16 [2] He-Zhang-Ren-Sun'15 [3] Page '19



Bansal-Kaplun-B, ICLR 2021

"Double descent"





Belkin, Hsu, Ma, Mandal, PNAS 2019.

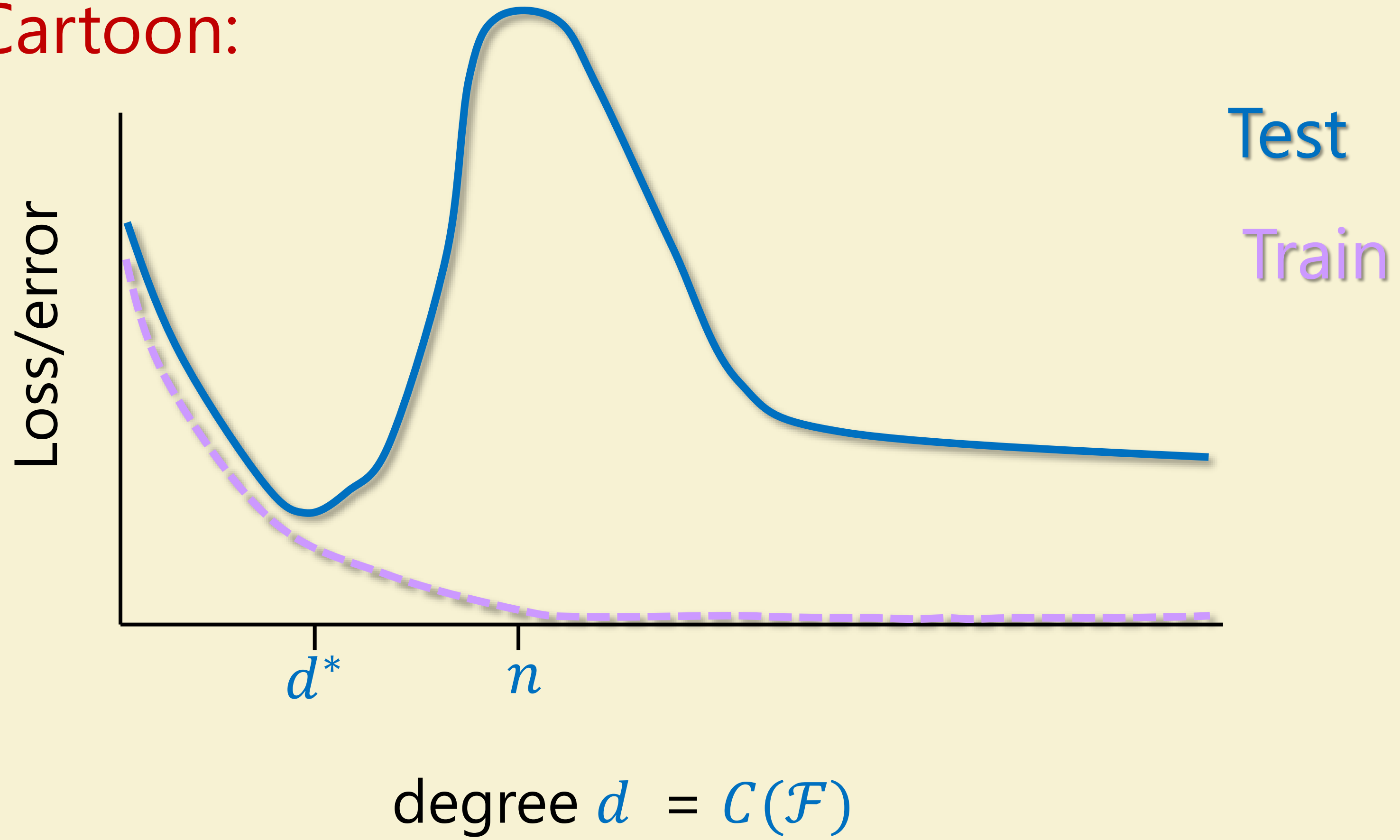
Nakkiran-Kaplun-Bansal-Yang-B-Sutskever, ICLR 2020

Example:

Data: $(x_i, f^*(x_i) + N)$ where f^* degree d^* polynomial

Algorithm: SGD to minimize $\sum(f(x_i) - y_i)^2$ where $f \in \mathcal{F}_d$=degree d polys

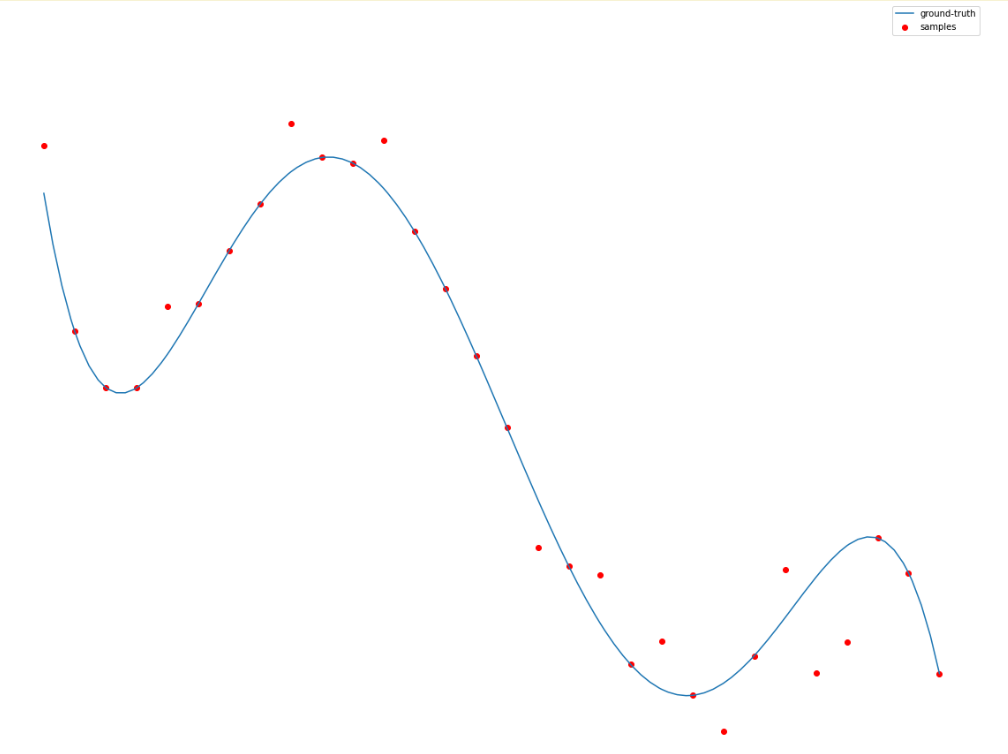
Cartoon:



Example:

Data: $(x_i, f^*(x_i) + N)$ where f^* degree d^* polynomial

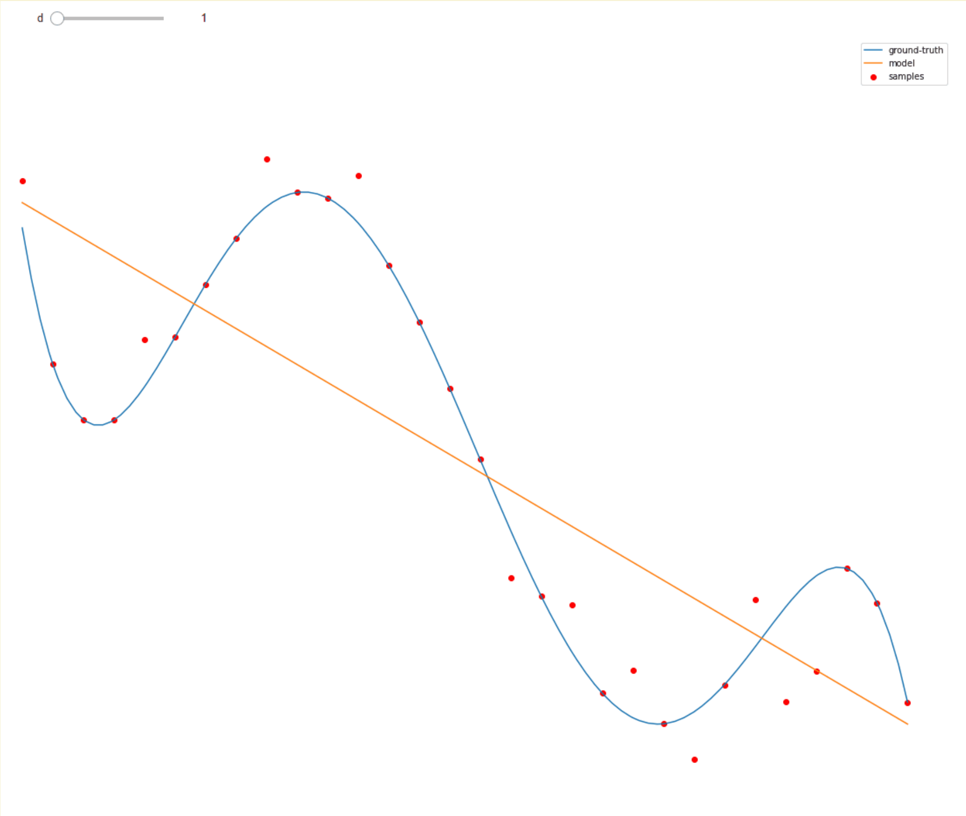
Algorithm: SGD to minimize $\sum(f(x_i) - y_i)^2$ where $f \in \mathcal{F}_d$=degree d polys

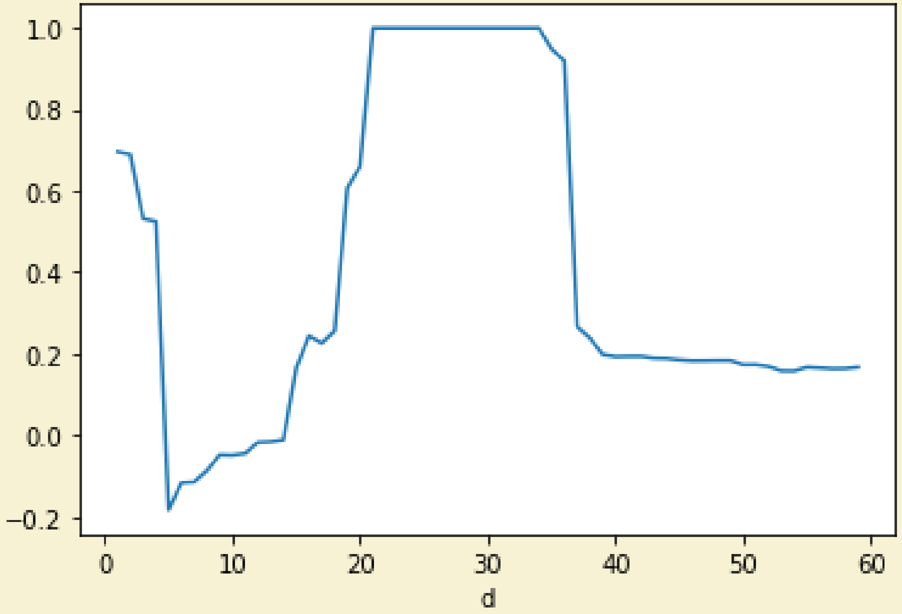


Example:

Data: $(x_i, f^*(x_i) + N)$ where f^* degree d^* polynomial

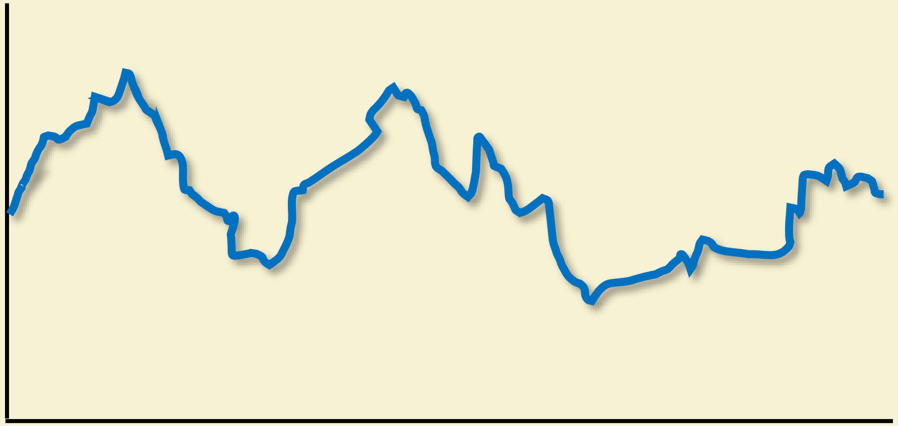
Algorithm: SGD to minimize $\sum(f(x_i) - y_i)^2$ where $f \in \mathcal{F}_d$=degree d polys

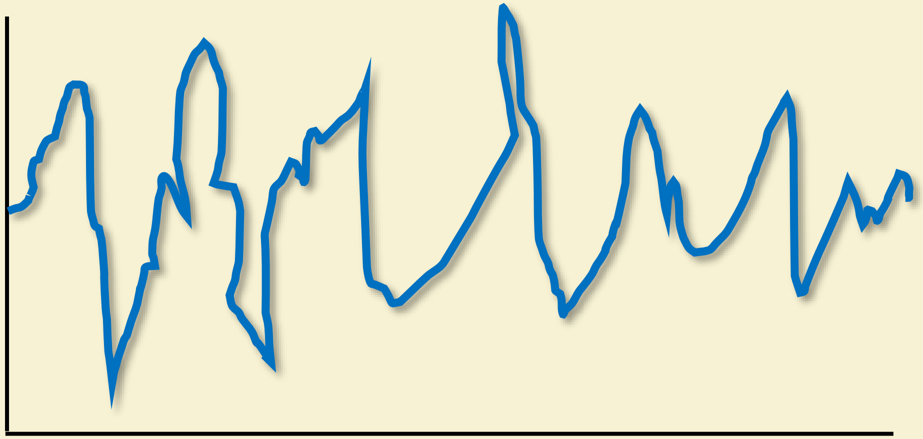




Part III: Approximation and Representation

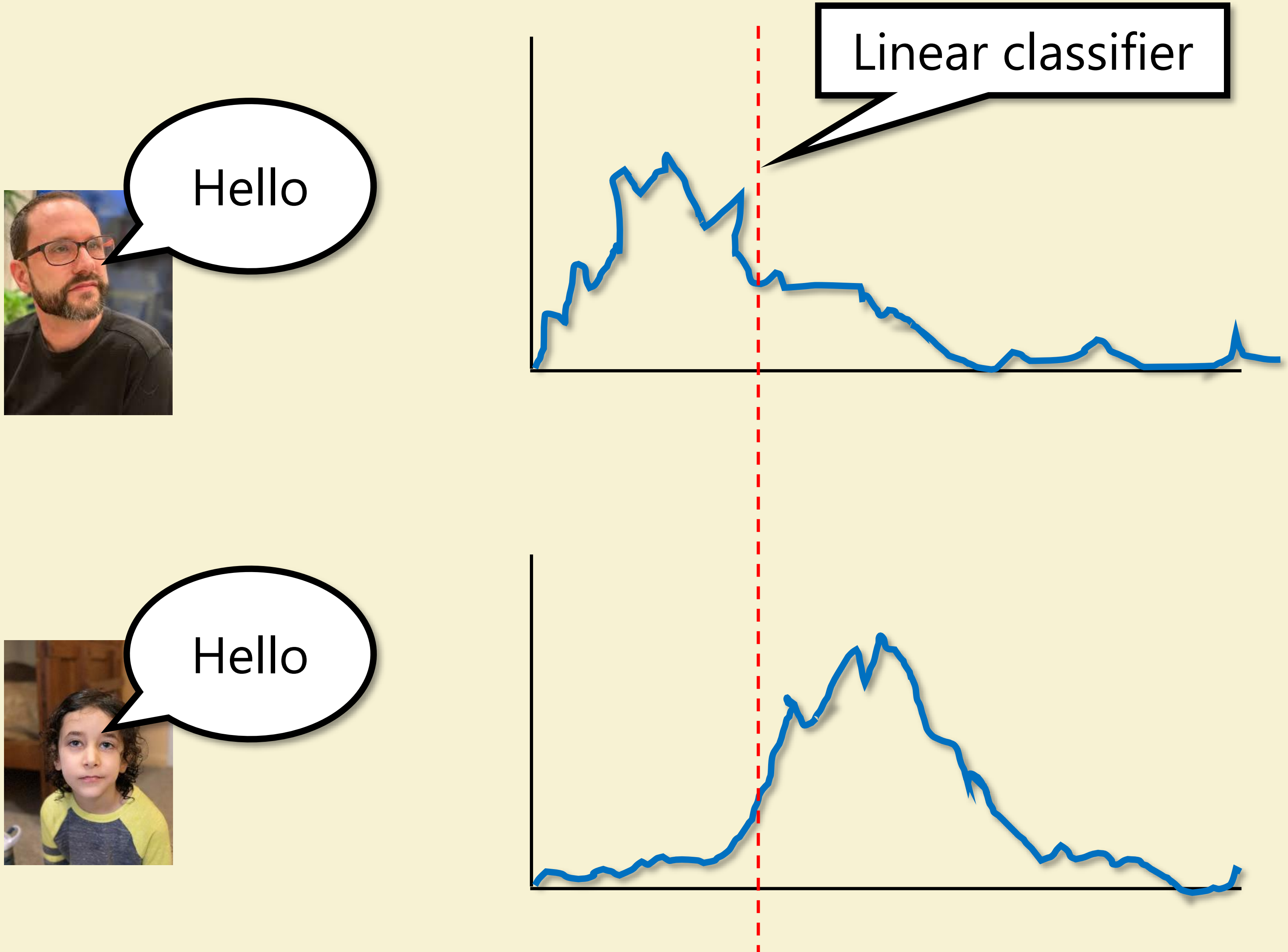
Example:





Example:

After Fourier Transform



Fourier Transform

$$\int_0^1 |f - g| < \epsilon$$

Every continuous $f: [0,1] \to \mathbb{R}$ can be arbitrarily approximated by g of form

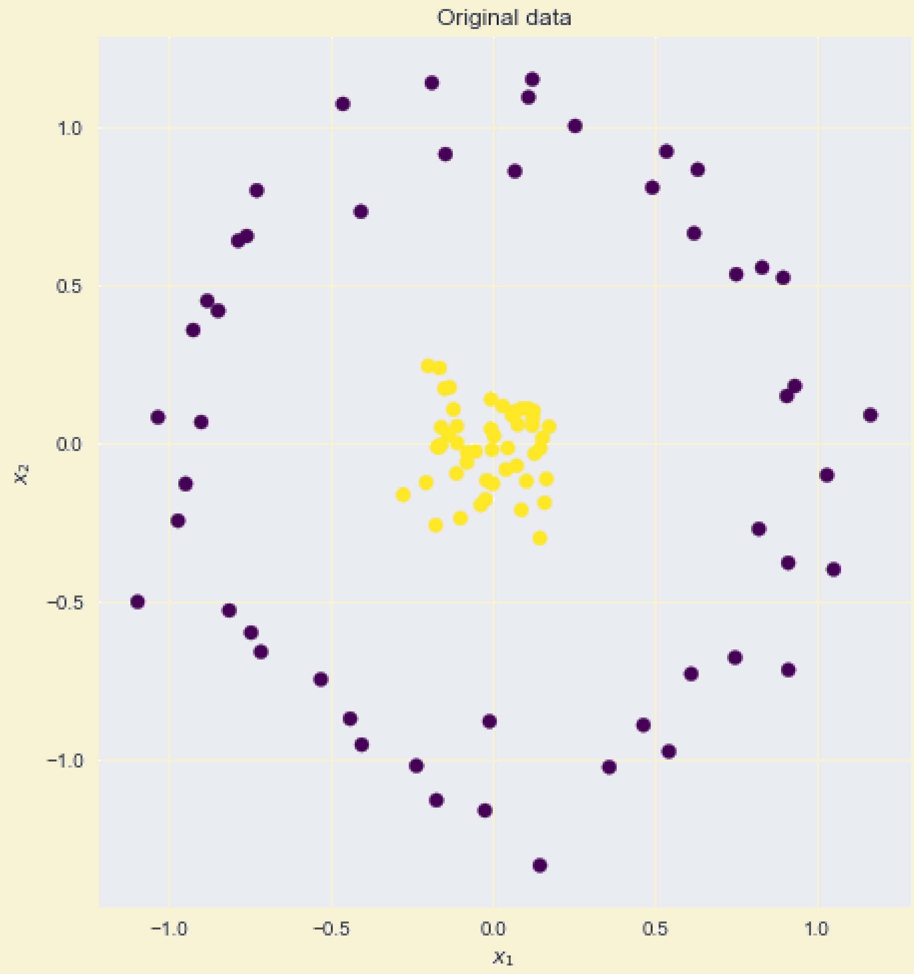
$$g(x) = \sum \alpha_j e^{-2\pi i\, \beta_j x}$$

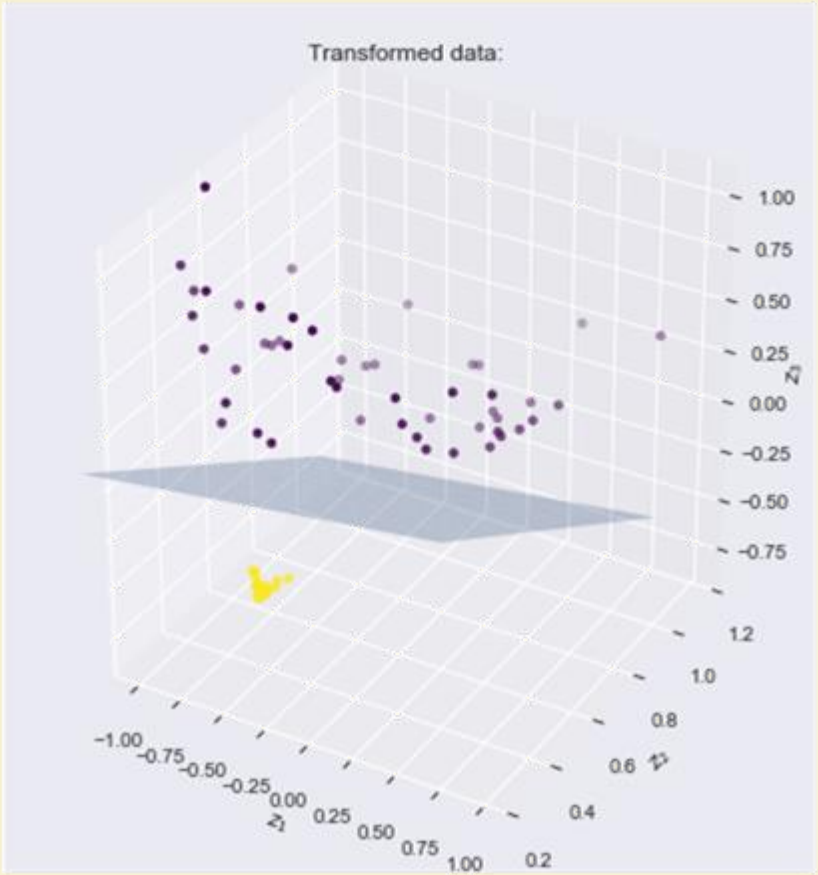
i.e., $f \ \approx$ linear in $\varphi_1(x), \dots, \varphi_N(x)$ where $\varphi_j(x) = e^{-2\pi i\, \beta_j x}$

$\varphi: \mathbb{R} \to \mathbb{R}^N$ embedding

For some natural data, representation φ is "nice" (e.g. sparse, linearly separable,…)

Tasks become simpler after transformation $x \to \varphi(x)$





$$\varphi(x, y) = (xy, x^2, y^2)$$

Xavier Bourret Sicotte, https://xavierbourretsicotte.github.io/Kernel_feature_map.html

More approximation

$$ReLU(x) = \max(x, 0)$$

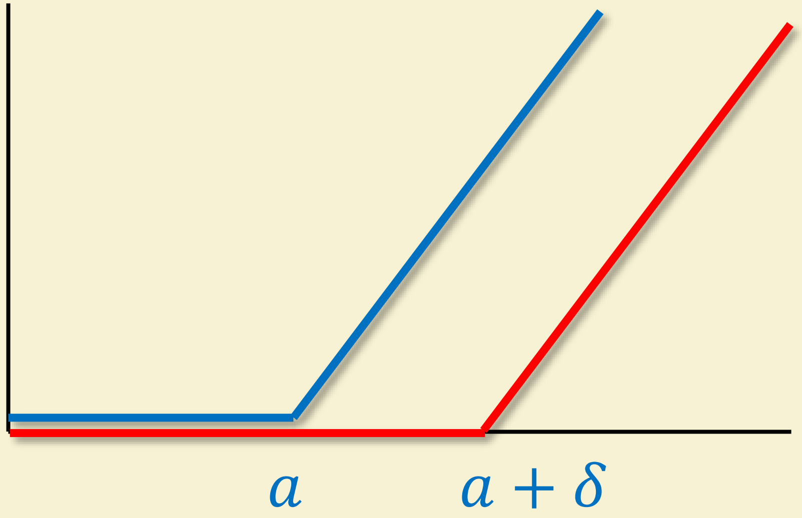
Every continuous $f: [0,1] \to \mathbb{R}$ can be arbitrarily approximated by g of form

$$g(x) = \sum \alpha_i \, ReLU(\beta_i x + \gamma_i)$$

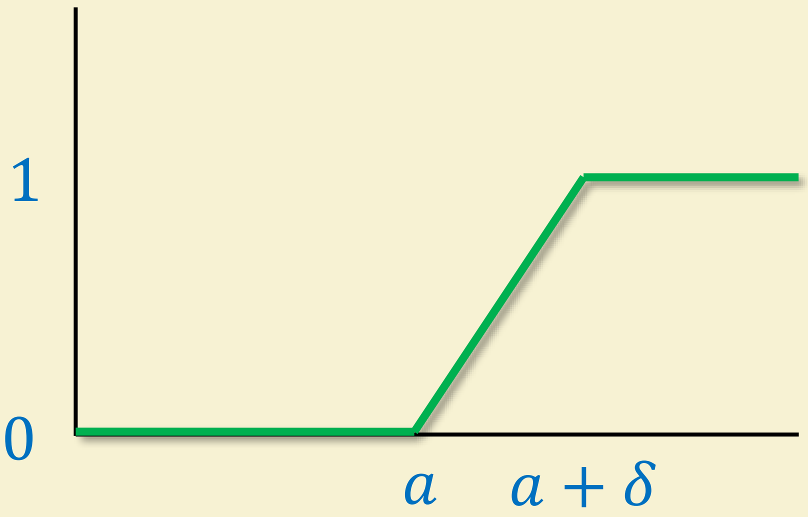
Proof by picture:

$g_a(x) = ReLU(x/\delta - a)$

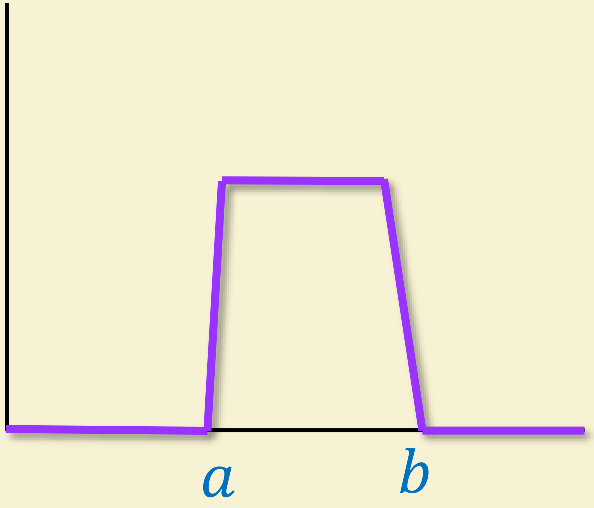
$g_{a+\delta}(x) = ReLU(x/\delta - a - \delta)$



$h_a = g_a - g_{a+\delta}$



$I_{a,b} = h_a - h_b$



More approximation

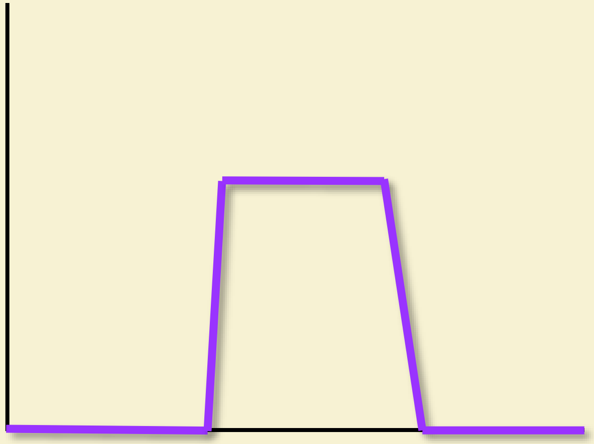
$$ReLU(x) = \max(x, 0)$$

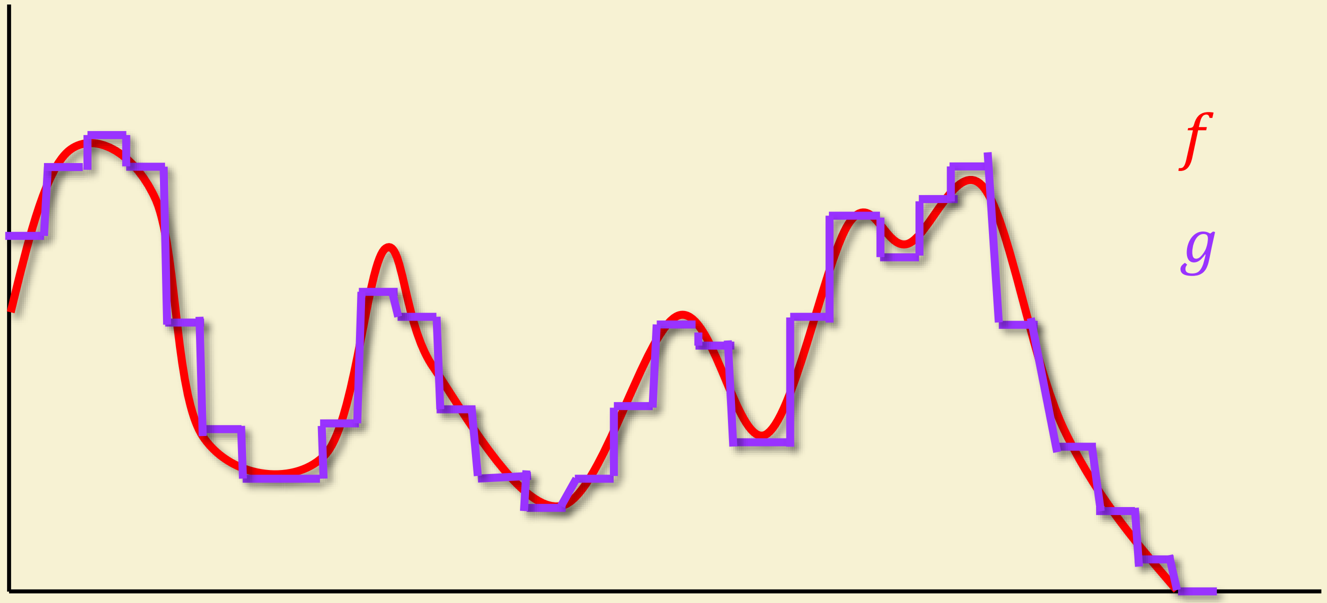
Every continuous $f: [0,1] \to \mathbb{R}$ can be arbitrarily approximated by g of form

$$g(x) \;\; = \sum \alpha_i \, ReLU(\beta_i x + \gamma_i)$$

Proof by picture:

$$I_{a,b} = h_a - h_b$$

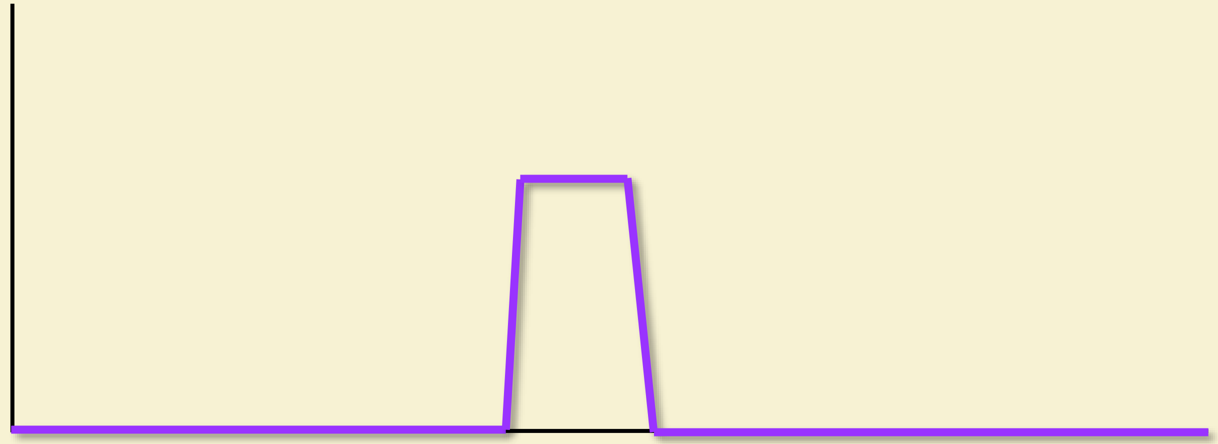




Higher dimension

$r(x) = \max(\langle \alpha, x \rangle + \beta, 0)$, $d + 1$ parameters

$I(x)$

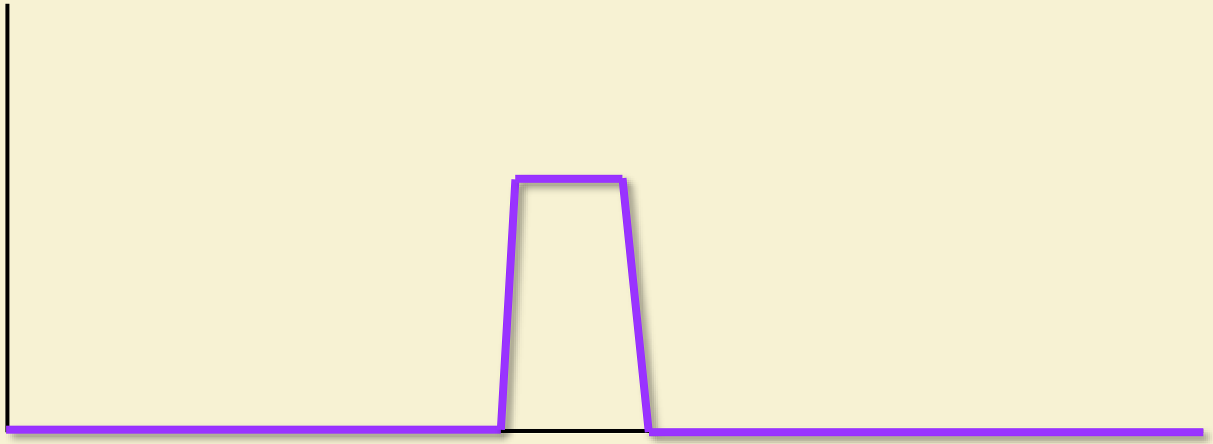


$J_{(1,0)}(x, y) = I(x)$

Higher dimension

$r(x) = \max(\langle \alpha, x \rangle + \beta, 0)$, $d + 1$ parameters

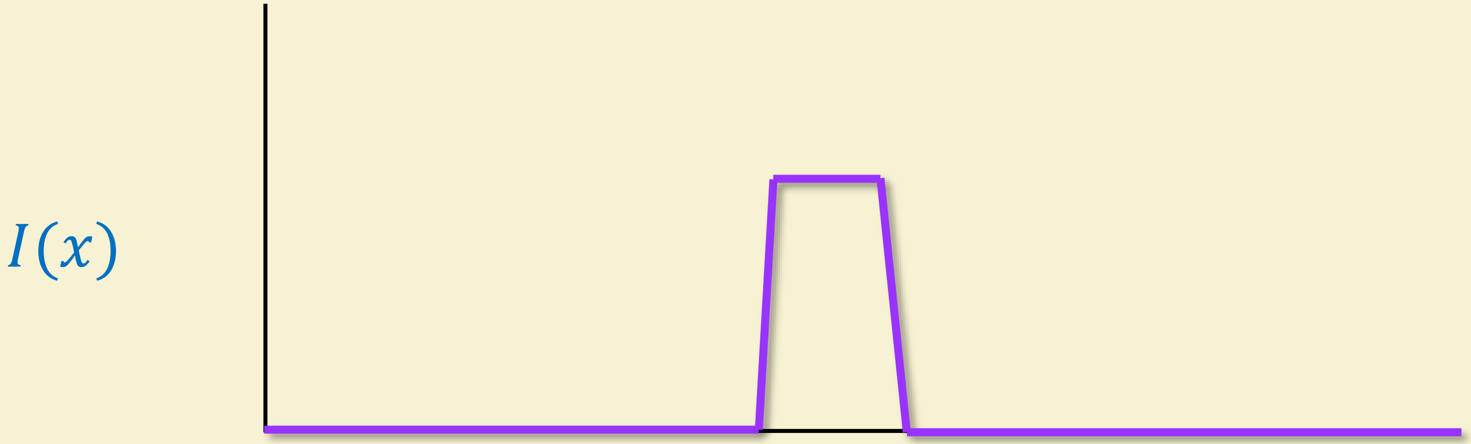
$I(x)$

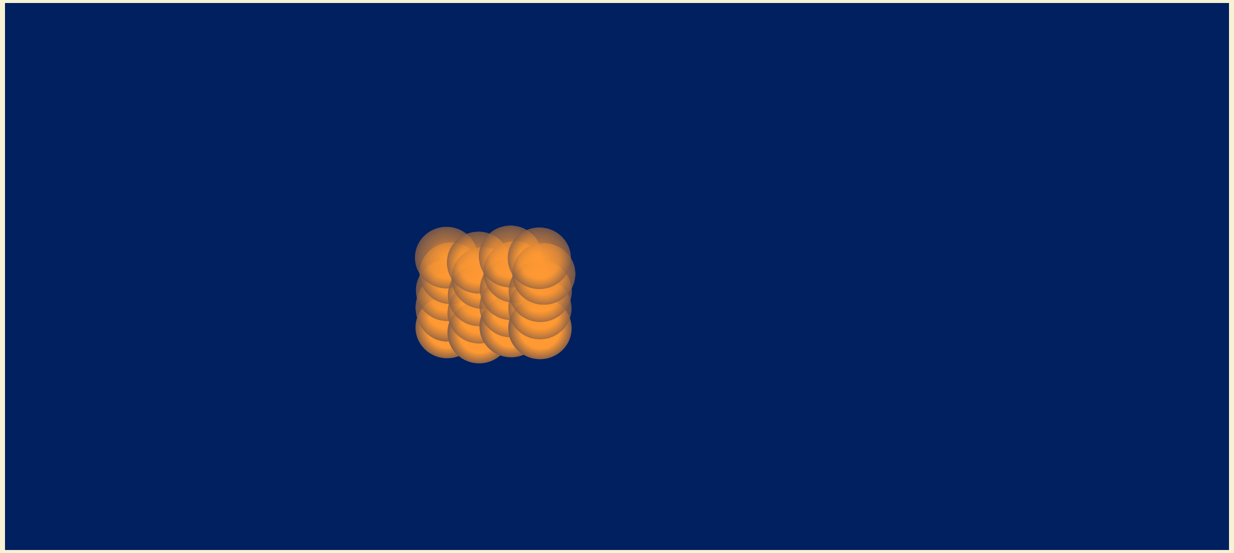


$J_{(1,0)}(x, y) = I(x)$

Higher dimension

$r(x) = \max(\langle \alpha, x \rangle + \beta, 0)$, $d+1$ parameters





How many ReLU's

Every $f: [0,1]^d \to \mathbb{R}$ can be approximated as sum of $r_{\alpha,\beta}(x) = \max(\langle \alpha, x \rangle + \beta)$

Can discretize $\mathbb{R}^{d+1}$ to $O(1)^d$ points – need at most $O(1)^d = 2^{O(d)}$ ReLUs

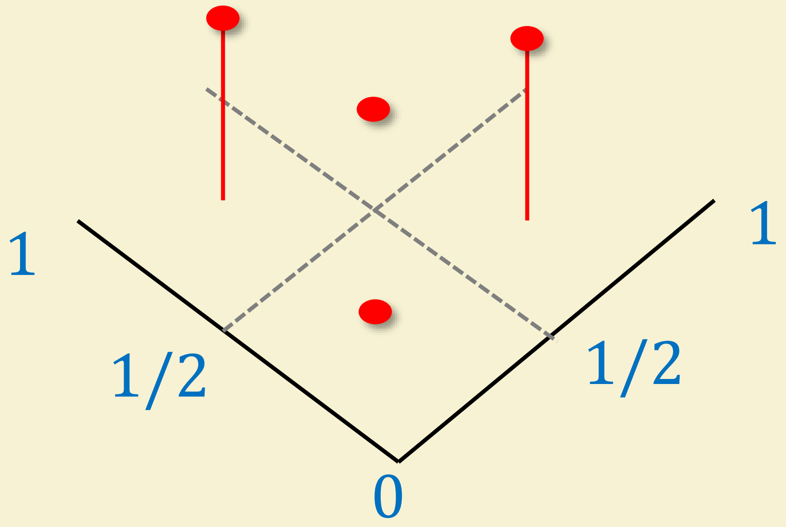
(For formal proofs, see Cybenko '89, Hornik '91, Barron '93; MIT 18.408)

THM: For some f's , $2^{c \cdot d}$ ReLU's are necessary

Random $f: [0,1]^d \to \mathbb{R}$

Choose $f(x) \in \{0,1\}$ independently at random
for $\mathrm{x} \in \{1/4, 3/4\}^d$

Extend linearly



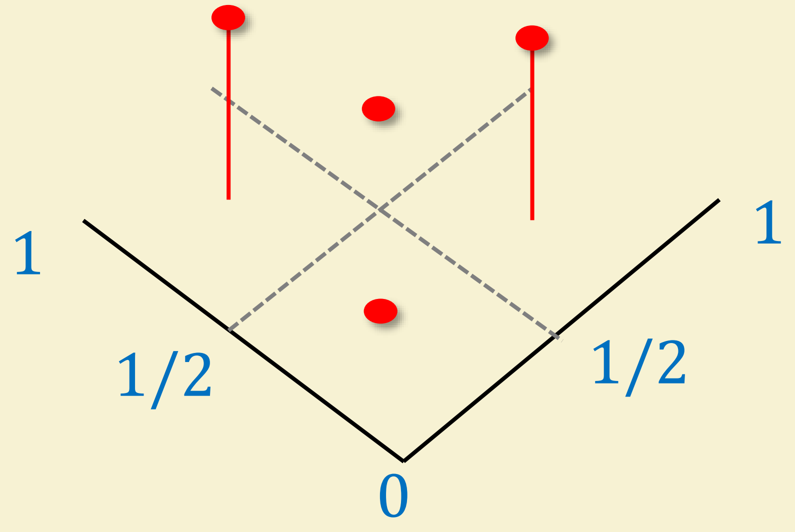
LEMMA: For every fixed $g: [0,1]^d \to \mathbb{R}$, $\Pr\left[\, \int |f - g| < 0.1 \,\right] \leq 2^{-\epsilon \cdot 2^d}$

THM: For some f's , $2^{c\cdot d}$ ReLU's are necessary

Random $f: [0,1]^d \to \mathbb{R}$

Choose $f(x) \in \{0,1\}$ independently at random
for $x \in \{1/4, 3/4\}^d$

Extend linearly



LEMMA: For every fixed $g: [0,1]^d \to \mathbb{R}$, $\Pr[\int |f-g| < 0.1] \leq 2^{-\epsilon\cdot 2^d}$

LEMMA ⇒ THEORM: Need $2^{\epsilon\cdot 2^d}$ distinct g's ⇒ need $\approx 2^d$ distinct ReLUs

PROOF OF LEMMA:
For $x \in \{1/4,3/4\}^d$ let $Z_x = 1$ iff $\int |g(y) - f(y)| dy > 0.2$ over $y \in [-1/4, +1/4]^d + x$

Each Z_x is independent with expectation at least $1/2$

$\Rightarrow \Pr[\frac{1}{2^d}\sum Z_x < 1/4] \leq 2^{-0.1\cdot 2^d}$ $\quad\quad$ $\frac{1}{2^d}\sum Z_x \geq 1/4 \Rightarrow \int |g-f| \geq 1/20$ $\blacksquare$

Bottom line

For every* function $f: \mathbb{R}^d \to \mathbb{R}$ many choices of $\varphi: \mathbb{R}^d \to \mathbb{R}^N$ s.t.

$$f \approx Linear_f \circ \varphi$$

Want to find φ that is useful for many interesting functions f

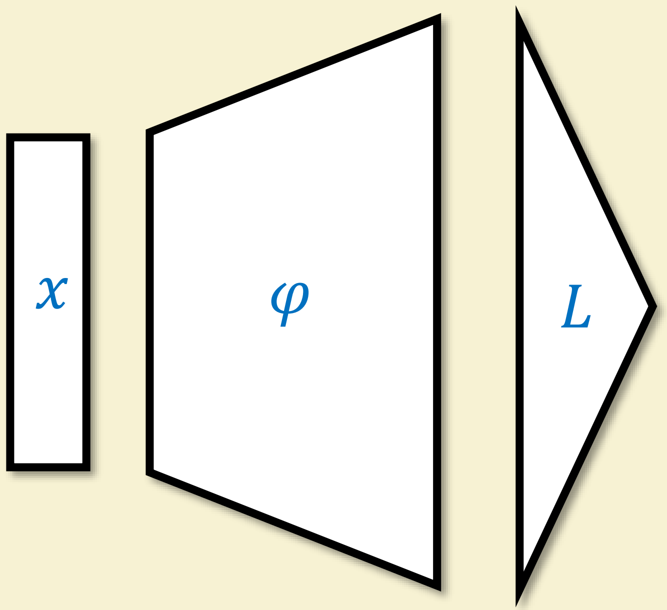
Or f's such that $\psi \circ f$ interesting for some non-linear $\psi: \mathbb{R} \to \mathbb{R}$

useful:

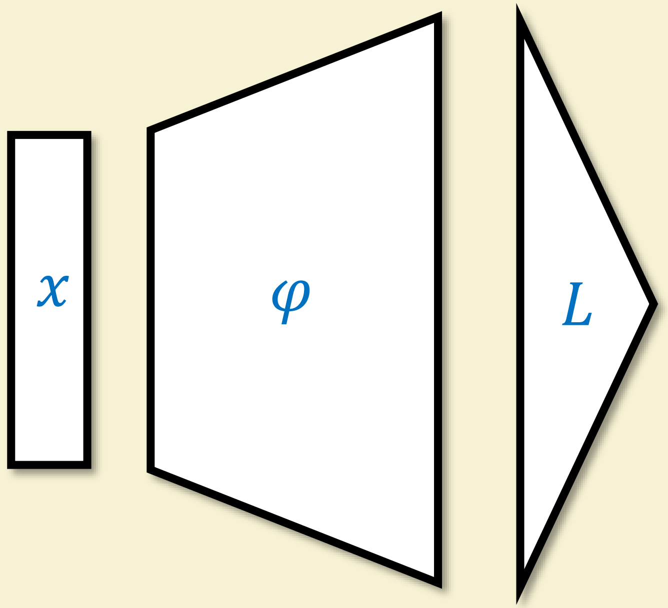
- N is not too big
- Efficiently compute $\varphi(x)$ or $\langle \varphi(x), \varphi(y) \rangle$
- $\langle \varphi(x), \varphi(y) \rangle$ large $\Leftrightarrow$ x and y are "semantically similar"
- For interesting f's, coefficients of $Linear_f$ are *structured*.
- …

Part IV: Kernels

Neural Net



Kernel



Definitely NN

Definitely Kernel

φ trained end-to-end for task at hand	φ pretrained with same data	φ pretrained on different data	φ pretrained on synthetic data	φ hand-crafted based on neural architectures	φ hand-crafted before 1990s

Kernel methods (intuitively)

Distance measure $K(x, x')$

Input: $(x_1, y_1), \dots, (x_n, y_n)$ s.t. $f^*(x_i) \approx y_i$

To approx $f^*(x)$ output y_i for x_i closest to x

or output $\sum \alpha_i y_i$ with α_i depending on $K(x, x_i)$

Hilbert Space

Linear space: $v + u$, $c \cdot v$

Dot product: $\langle u, v + c \cdot w \rangle = \langle u, v \rangle + c\langle u, w \rangle$, $\langle u, v \rangle = \langle v, u \rangle$, $\langle v, v \rangle \geq 0$

Can solve linear equations of form $\{ \langle v_i, x \rangle = b_i \}$ knowing only $\langle v_i, v_j \rangle$

Also do least-square minimization $\min_x \sum (\langle v_i, x \rangle - b_i)\hat{}2$ knowing only $\langle v_i, v_j \rangle$

DEF: Sym matrix $K \in \mathbb{R}^{n \times n}$ is p.s.d. if $v^T K v \geq 0$ for all $v \in \mathbb{R}^n$

Equivalently $\lambda_1(K), \ldots, \lambda_n(K) \geq 0$

CLAIM: K is p.s.d. iff* $u^T K v$ is inner product

PROOF ($\Rightarrow$): $u^T K v = \psi(u) \cdot \psi(v)$ where $\psi(u_1, \ldots, u_n) = (\sqrt{\lambda_1} u_1, \ldots, \sqrt{\lambda_n} u_n)$, expressed in eigenbasis

Kernel Methods

Goal: Solve linear equations, least-square minimization, etc. under φ

Observation: Enough to compute $\mathrm{K}(\mathrm{x}, \mathrm{x}') = \langle \varphi(x), \varphi(x') \rangle$

Let $\hat{x} = \varphi(x)$, given $\{\hat{x}_i, y_i\}_{i=1..n}$ want to find $\widehat{w} \in \mathbb{R}^n$ s.t. $\langle \hat{x}_i, \widehat{w} \rangle \approx y_i \ \forall i$

can compute $\widehat{w} = \sum \alpha_i \ \hat{x}_i$ using $K(x_i, x_j)$

To compute prediction on new x using $\widehat{w}$, can compute $\hat{x} = \sum \beta_i \ \hat{x}_i$ using $K(x, x_i)$