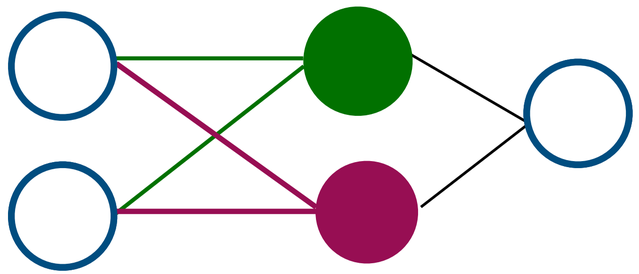
Neural Networks: A Theory Lab

Shallow neural networks



Hadi Daneshmand

University of Virginia

Regression problem

- Given $(x_1, y_1), \ldots, (x_n, y_n) \implies (x_{n+1}, ?)$
- Where $y_i = f(x_i)$ where function f is unknown

Neural Networks vs Random Features

Random Features

$$\min_{\alpha_i} \sum_j (y_j - \sum_i \alpha_i \cos(\langle v_i, x_j \rangle + b_i))^2$$

$v_i \sim p(w)$ depending on $k(x, y)$

Neural Networks

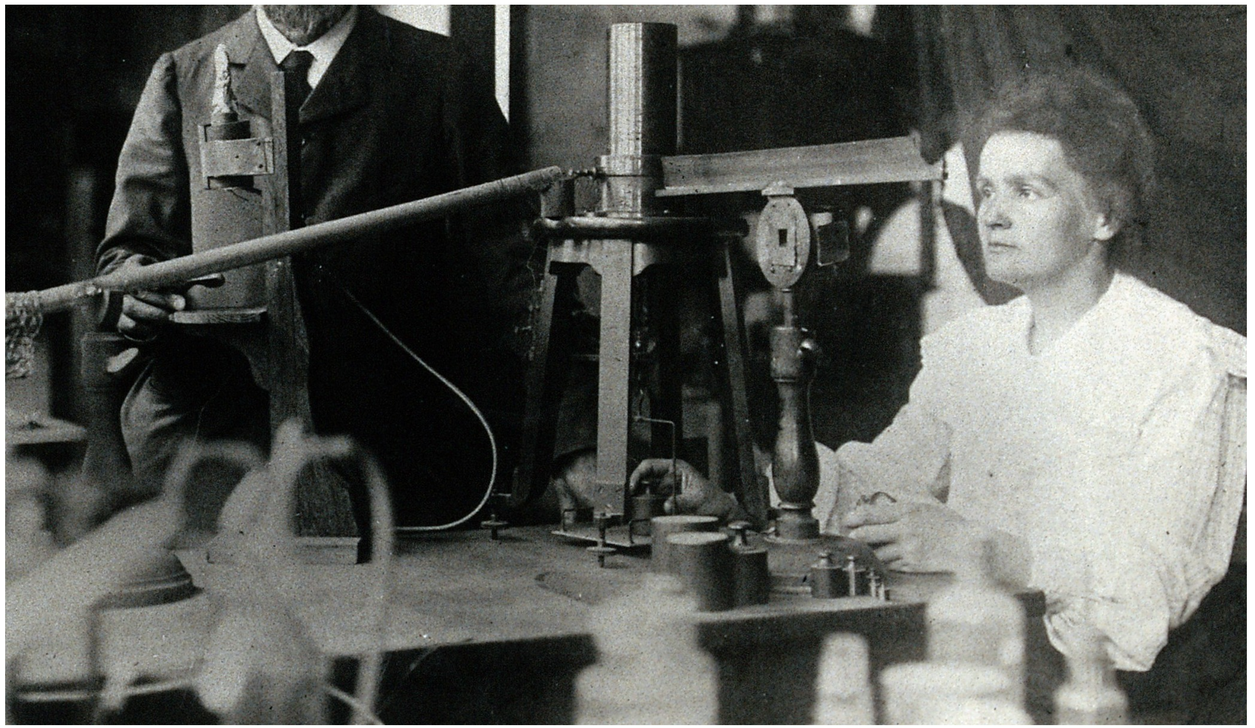
$$\min_{\alpha_i, v_i, b_i} \sum_j (y_j - \sum_i \alpha_i \cos(\langle v_i, x_j \rangle + b_i))^2$$

v_i are optimized since $p(w)$ is unknown

Goal and outline

- Experiments in groups
  - Observing issues with random features
- Theoretical background
  - Neural networks do not suffer from the curse of dimensionality

- A brief recap
- Experiments
- Theory

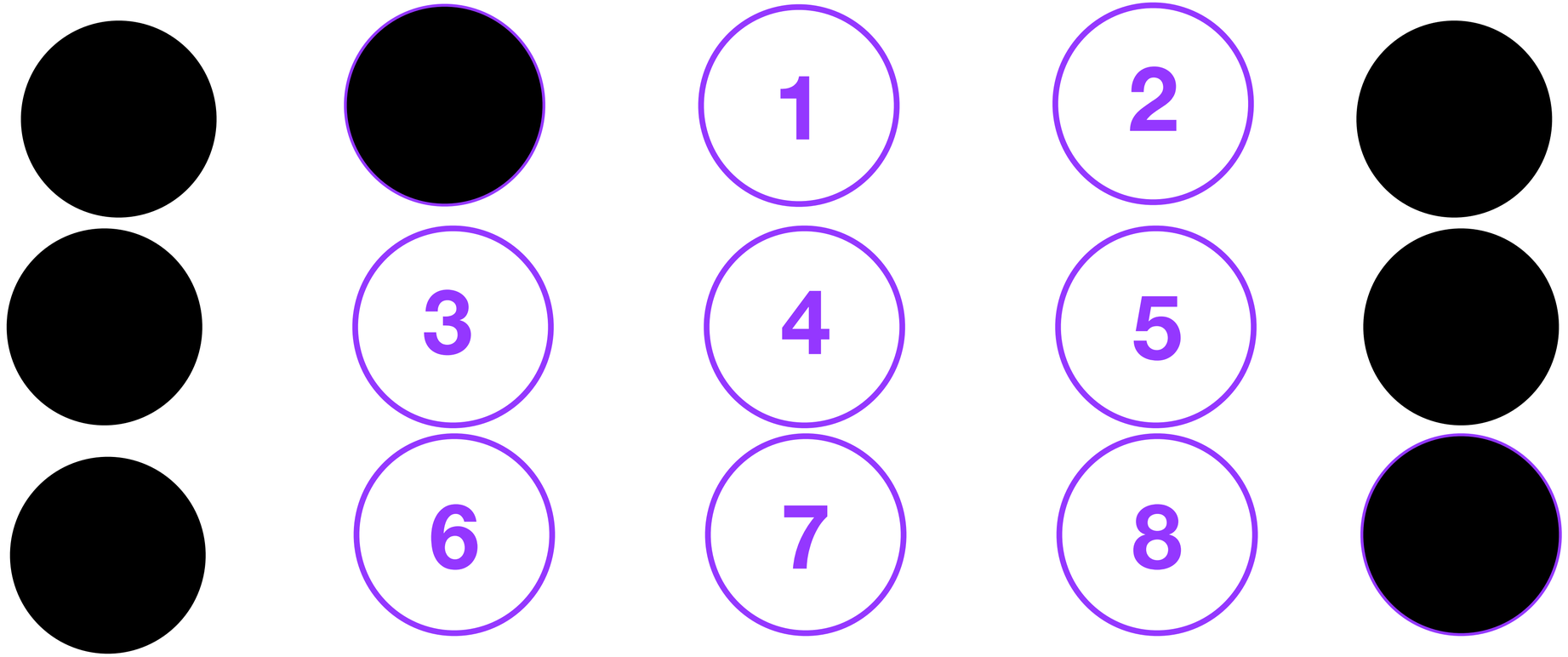


wikipedia: Marie Curie

Experiments

Get ready for hands-on group activity

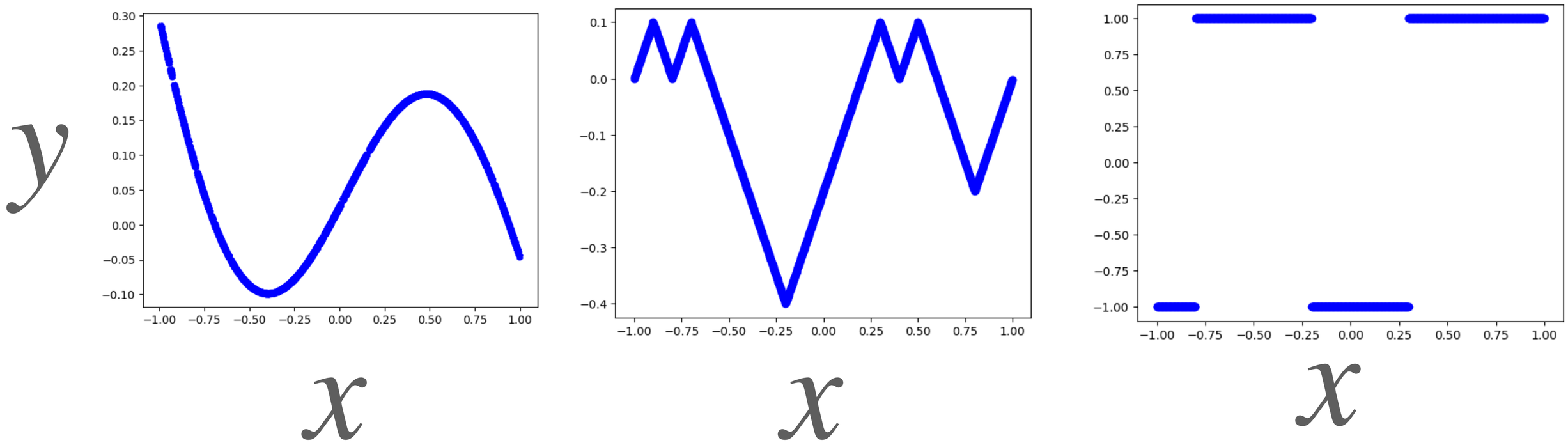
Group assignment



I am here

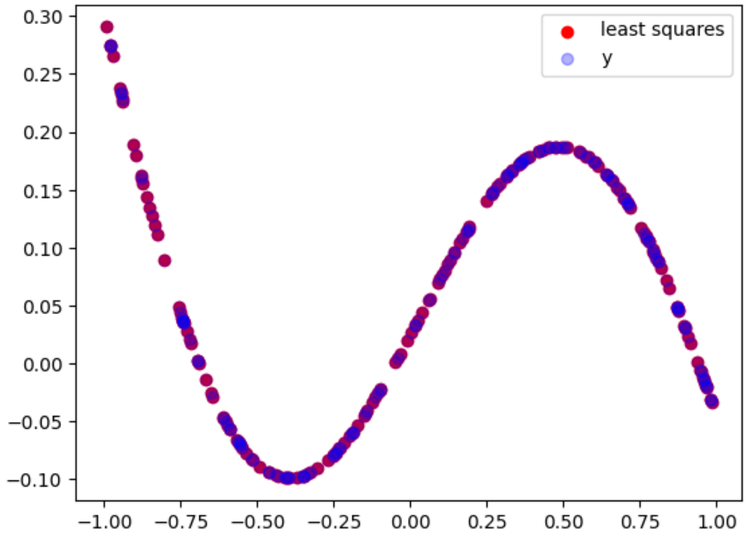
Recap

- Goal: We want to design **universal** non-linear features for 3 functions

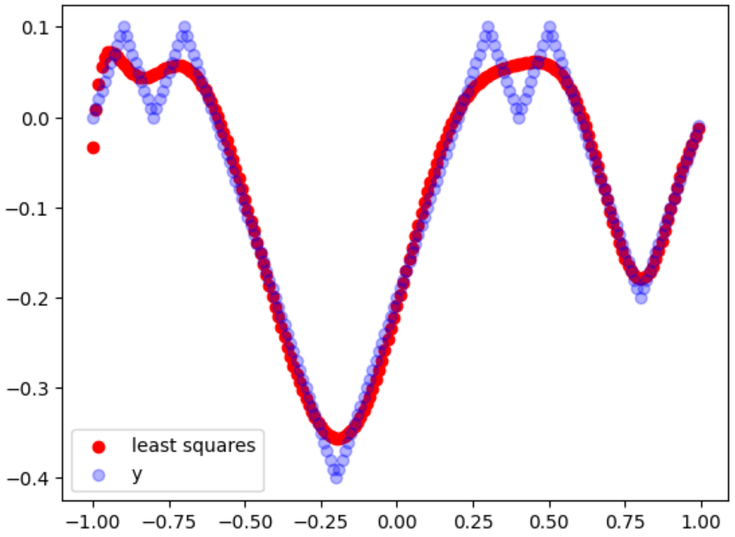


Performance of random gaussian features

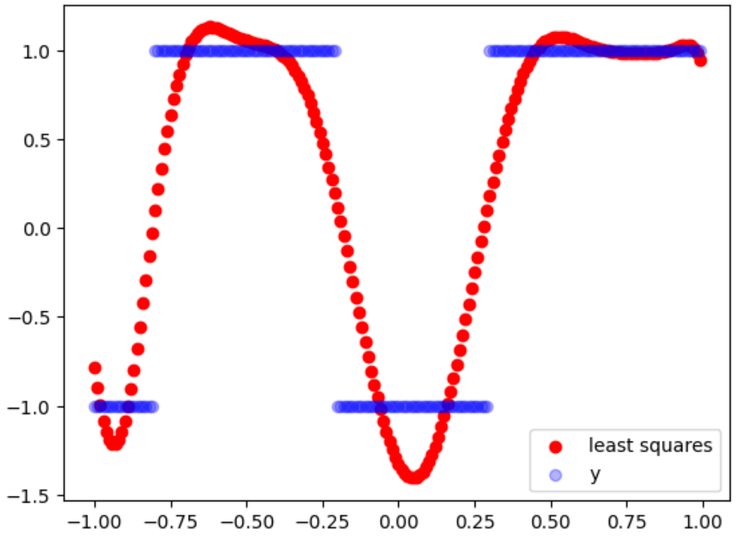
- Blue: y and red is the solution of regression on random features



Good approximation



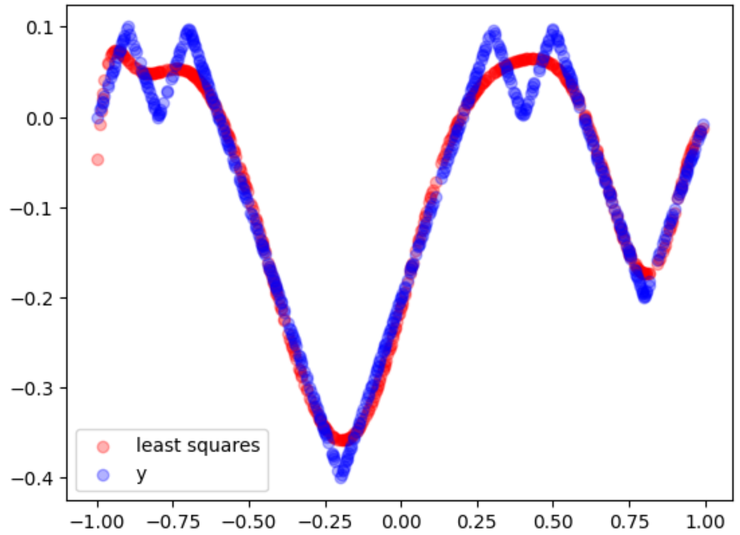
Reasonable approximation



Poor approximation

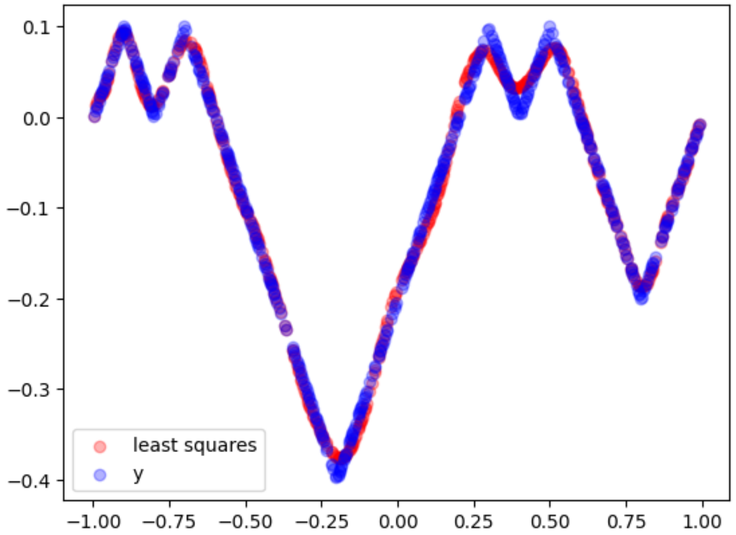
Discuss: How to improve random features?

I used Gaussian features: $y \approx \sum_i \alpha_i \cos(\langle v_i, x \rangle + b_i), v_i \sim N(0, I_d)$



Reasonable approximation

How?



Ideal approximation

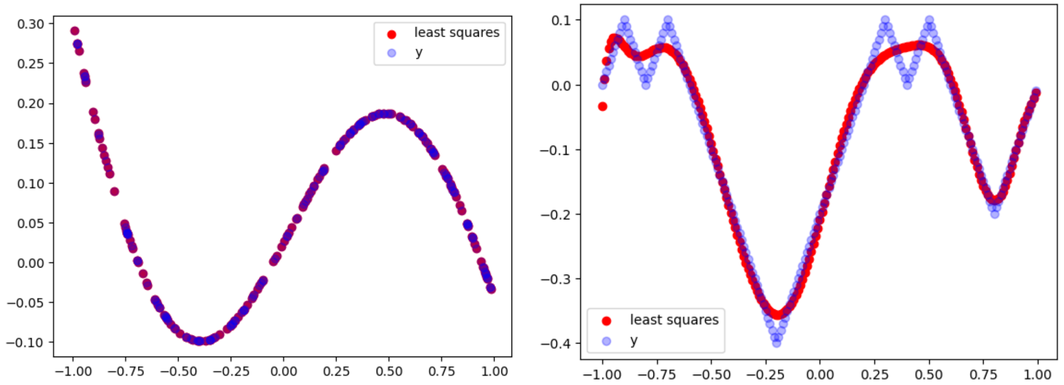
Hint: use theoretical insights from last lecture

Theory

$$k(x-y) = \mathbb{E}\left[\cos(\langle w, x\rangle + b)\cos(\langle w, y\rangle + b)\right]$$

Random features approximate a kernel

Observation



Random features can fit various functions

Distribution choice for random features

$$k(x-y) = 2\mathbb{E}_{w\sim p}\left[\cos(\langle w, x\rangle + b)\cos(\langle w, y\rangle + b)\right]$$

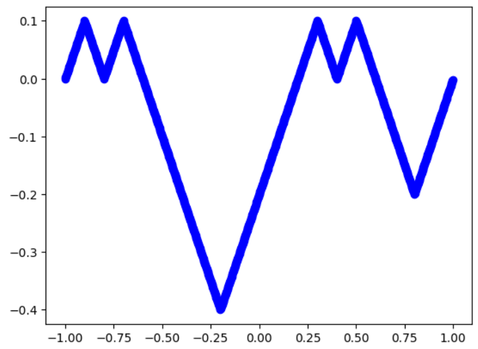
Kernel name	$k(\Delta)$	$p(w)$
Gaussian	$e^{-\frac{1}{2}\|\Delta\|_2^2}$	$(2\pi)^{-d/2}e^{-\frac{1}{2}\|w\|_2^2}$
Cauchy	$e^{-\|\Delta\|_1}$	$\prod_i \frac{1}{\pi(1+w_i^2)}$
Laplacian	$\prod_i \frac{1}{\pi(1+\Delta_i^2)}$	$e^{-\|w\|_1}$

[Rahimi & Recht 2007

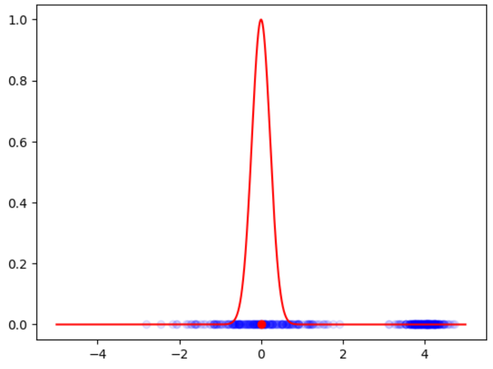
A solution

- Kernel estimation works well for functions obeying **reproducing property**

Function $f(x)$



Kernels $k(x, y)$



Random feature $\phi_i(x)$

Kernel name	$k(\Delta)$	$p(w)$
Gaussian	$e^{-\frac{1}{2}\|\Delta\|_2^2}$	$(2\pi)^{-d/2}e^{-\frac{1}{2}\|\Delta\|_2^2}$
Cauchy	$e^{-\|\Delta\|_1}$	$\prod_i \frac{1}{\pi(1+w_i^2)}$
Laplacian	$\prod_i \frac{1}{\pi(1+\Delta_i^2)}$	$e^{-\|w\|_1}$

[Rahimi &

$$\phi_i(x) = \cos(\langle w^{(i)}, x\rangle + b_i), w_i \sim p(w)$$

A solution

- Kernel estimation works well for functions obeying **reproducing property**
- Idea: adapting the distribution of random features

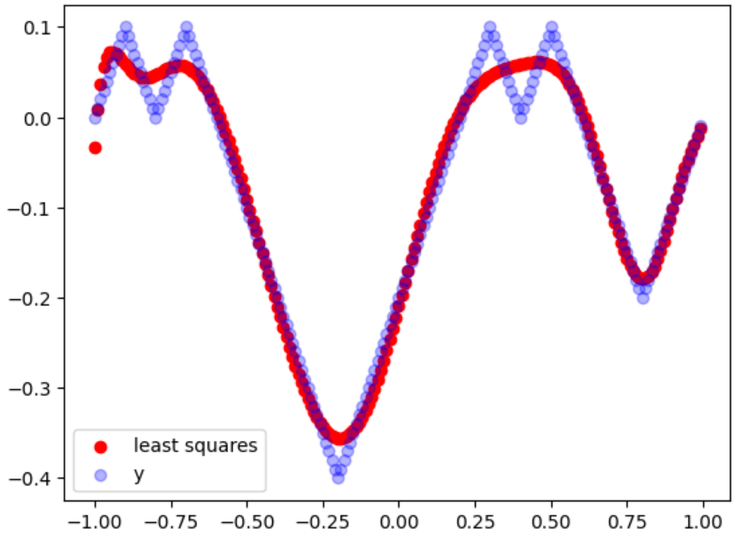
Random feature $\phi_i(x)$

Kernel name	$k(\Delta)$	$p(w)$
Gaussian	$e^{-\frac{1}{2}\|\Delta\|_2^2}$	$(2\pi)^{-d/2}e^{-\frac{1}{2}\|w\|_2^2}$
Cauchy	$e^{-\|\Delta\|_1}$	$\prod_i \frac{1}{\pi(1+w_i^2)}$
Laplacian [Rahimi &	$\prod_i \frac{1}{\pi(1+\Delta_i^2)}$	$e^{-\|w\|_1}$

$$\phi_i(x) = \cos(\langle w^{(i)}, x\rangle + b_i), w_i \sim p(w)$$

Group activity: implement Cauchy random features

Gaussian random feature



$$p(w) = (2\pi)^{-d/2} e^{-\frac{1}{2}\|\Delta\|_2^2}$$

Cauchy random feature

?

$$p(w) = \prod_i \frac{1}{\pi(1 + w_i^2)}$$

Group activity: implementation

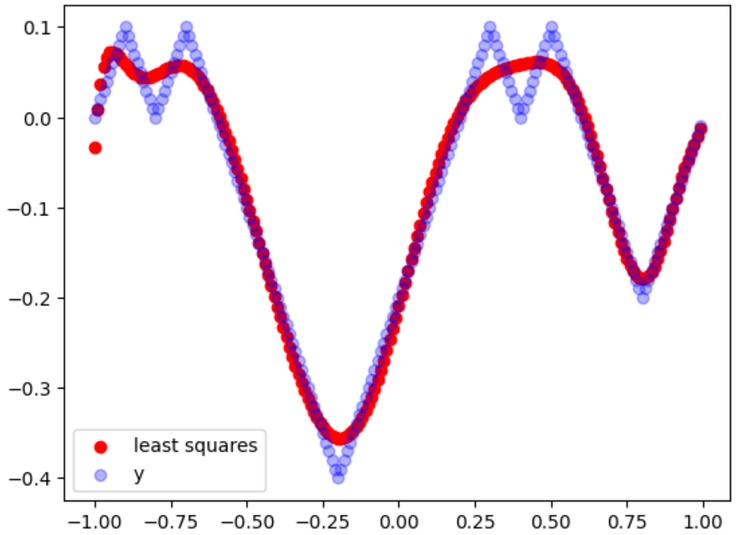
- $y \approx \sum_i \alpha_i \cos(\langle v_i, x \rangle + b_i), v_i \sim$ cauchy

Task 3 in colab: https://shorturl.at/ABb6H

https://colab.research.google.com/drive/1M-nRBdhg1XJiV8sy4wMkUsfPJLKKSCB0?usp=sharing

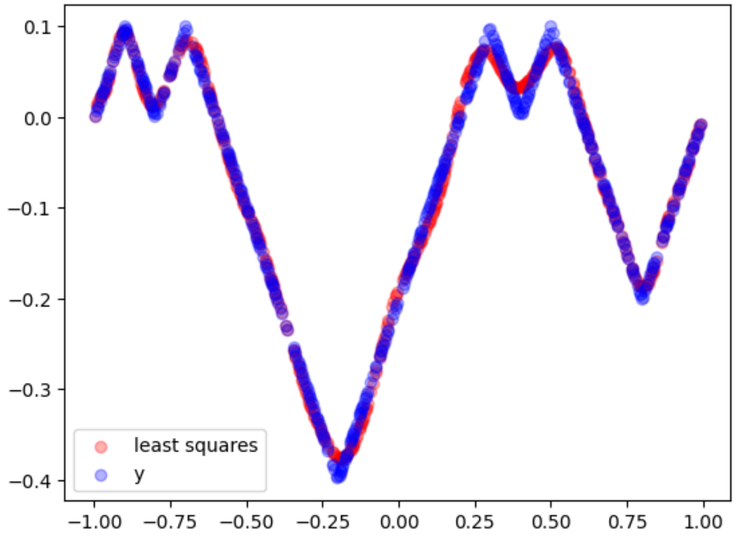
Comparing random features

Gaussian random feature



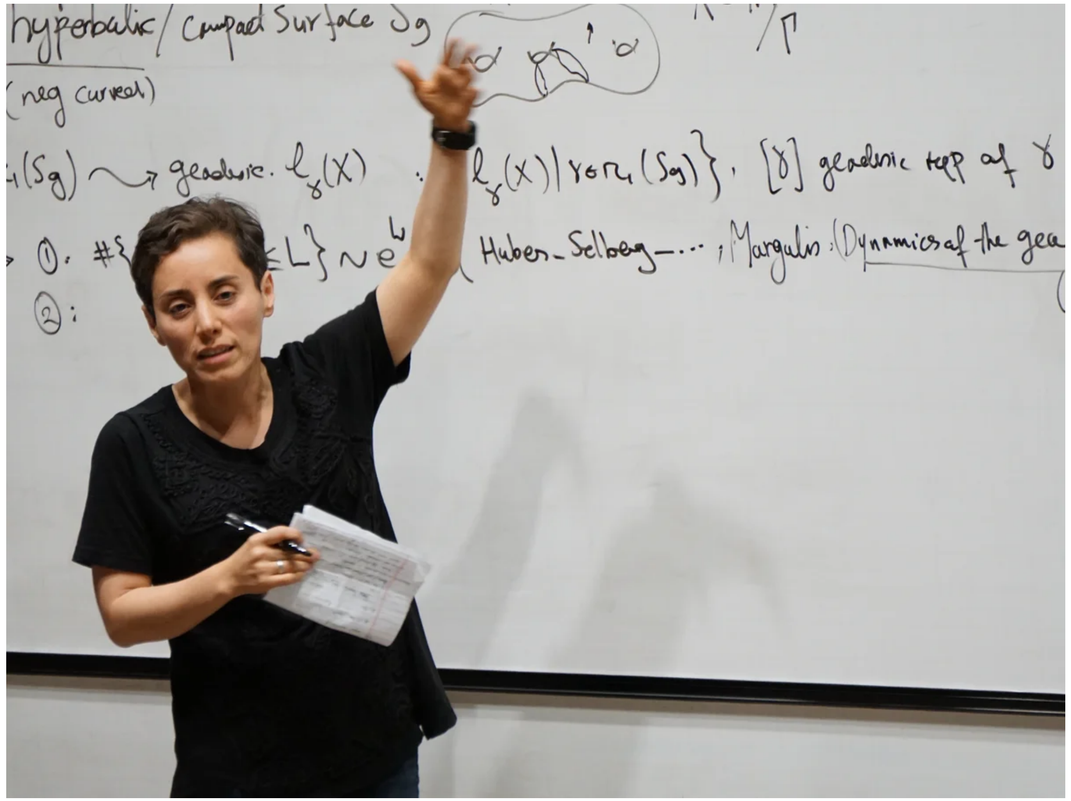
$$p(w) = (2\pi)^{-d/2} e^{-\frac{1}{2}\|\Delta\|_2^2}$$

Cauchy random feature



$$p(w) = \prod_i \frac{1}{\pi(1 + w_i^2)}$$

- Recap
- Experiments
- Theory



theguardian: Maryam Mirzakhani

Theory

Get ready for math

Optimization over distributions

- Random features approximate y by a linear transformation of non-linear features:

$$\min_{\alpha_i} \mathbb{E}\left(\sum_i \alpha_i \cos(\langle v_i, x\rangle + b_i) - y\right)^2, v_i \sim p(w), b_i \sim uniform$$

- For an unknown p: $\min_{\alpha,p,b} \mathbb{E}\left(\int \alpha(v)\cos(\langle v, x\rangle + b(v))p(v)dv - y\right)^2$
  - $p, a, b : \mathbb{R}^d \to \mathbb{R}$ are functions.

Particle optimization

Abstract problem: $\arg\min_{p} \mathscr{L}\left(\int \phi(v)p(v)\right), \quad p$ is a probability measure

How can we optimize with respect to probability measure?

Particle method: $\arg\min_{v_1,\ldots,w_m} \mathscr{L}\left(\frac{1}{m}\sum_{i=1}^{m}\phi(v_i)\right)$

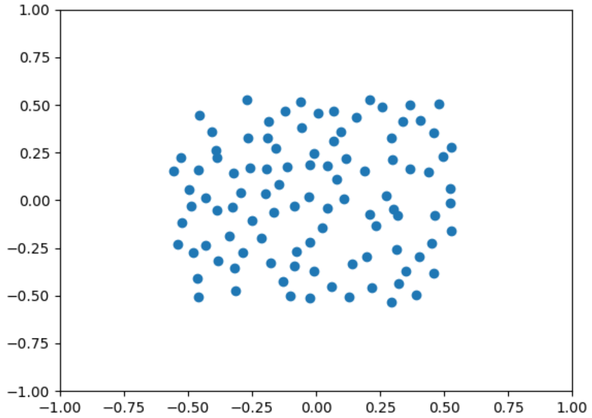
An example of particle optimization

$$\mathscr{L}(p) := \frac{1}{2} \min_{p} \int \mathscr{K}(x-y)dp(x)dp(y), \quad \mathscr{K}(\Delta) = -\log(\|\Delta\|) + \|\Delta\|^2$$

Surprise 1: the solution in 2D is the uniform measure over a circle

[See Global Minimizers of a Large Class of Anisotropic Attractive-Repulsive Interaction Energies in 2D]

Particle Method: $\min_{v_1,\ldots,v_m} \sum_{i=1}^{m} \sum_{j\neq i,1}^{m} \mathscr{K}\left(v_i - v_j\right)$



Particle gradient descent is extensively studied in physics

[see works of José A. Carrillo]

Neural Networks as a particle method

- Random features with an unknown distribution p

$$\min_{\alpha,p,b} \mathbb{E}\left(\underbrace{\int \alpha(v)\cos(\langle v, x\rangle + b(v))p(v)dw}_{f(x)} - y \right)^2$$

  - $p, a, b : \mathbb{R}^d \to \mathbb{R}$ are functions.

- Neural Networks: $\min_{\alpha_i, v_i, b_i} \mathbb{E}\left(\sum_i \alpha_i \cos(\langle v_i, x\rangle + b_i) - y \right)^2$

Barron's theorem

A. R. Barron, "Universal approximation bounds for superpositions of a sigmoidal function," in *IEEE Transactions on Information Theory*, 1993

- Suppose $y = f(x)$ has the Fourier transform

$$\int e^{i\langle w,x\rangle}\hat{f}(w)dw \text{ such that } C_f = \int |w||\hat{f}(w)|dw \text{ is finite.}$$

- Then, there are $v_1, \ldots, v_n \in \mathbb{R}^d$ and $b_1, \ldots, b_n \in \mathbb{R}$ such that

$$\int \left(y - \sum_i \alpha_i \cos(\langle v_i, x\rangle + b_i) \right)^2 p(x)dx \leq \frac{(2rC_f)^2}{n}$$

holds where $p(x)$ is a probability measure over the ball of radius r.

Barrons' comparison

A. R. Barron, "Universal approximation bounds for superpositions of a sigmoidal function," in *IEEE Transactions on Information Theory*, 1993

$$f(x) \approx \underbrace{\sum_i \alpha_i \cos(\langle v_i, x \rangle + b_i)}_{f_n(x)}$$

Random Features

v_i independent of y

$$\mathbb{E}(f(x) - f_n(x))^2 \geq c \frac{C_f}{d} \left(\frac{1}{n}\right)^{1/d}$$

$$n = \Omega((\frac{1}{\epsilon})^d) \text{ for } \epsilon - \text{accuracy}$$

Neural Networks

v_i are optimized for y

$$\mathbb{E}(f(x) - f_n(x))^2 = O\left(\frac{1}{n}\right)$$

$$n = O(\frac{1}{\epsilon}) \text{ for } \epsilon - \text{accuracy}$$

Curse of dimensionality for function approximation

Random Features

v_i independent of y

$$\mathbb{E}(f(x) - f_n(x))^2 \geq c\frac{C_f}{d}\left(\frac{1}{n}\right)^{1/d}$$

$$n = \Omega((\frac{1}{\epsilon})^d) \text{ for } \epsilon\text{-accuracy}$$

Neural networks can break curse of dimensionality

Neural Networks vs Random Features

Random Features

$$y \approx \sum_i \alpha_i \cos(\langle v_i, x \rangle + b_i)$$

$v_i \sim p(w)$ depending on $k(x, y)$

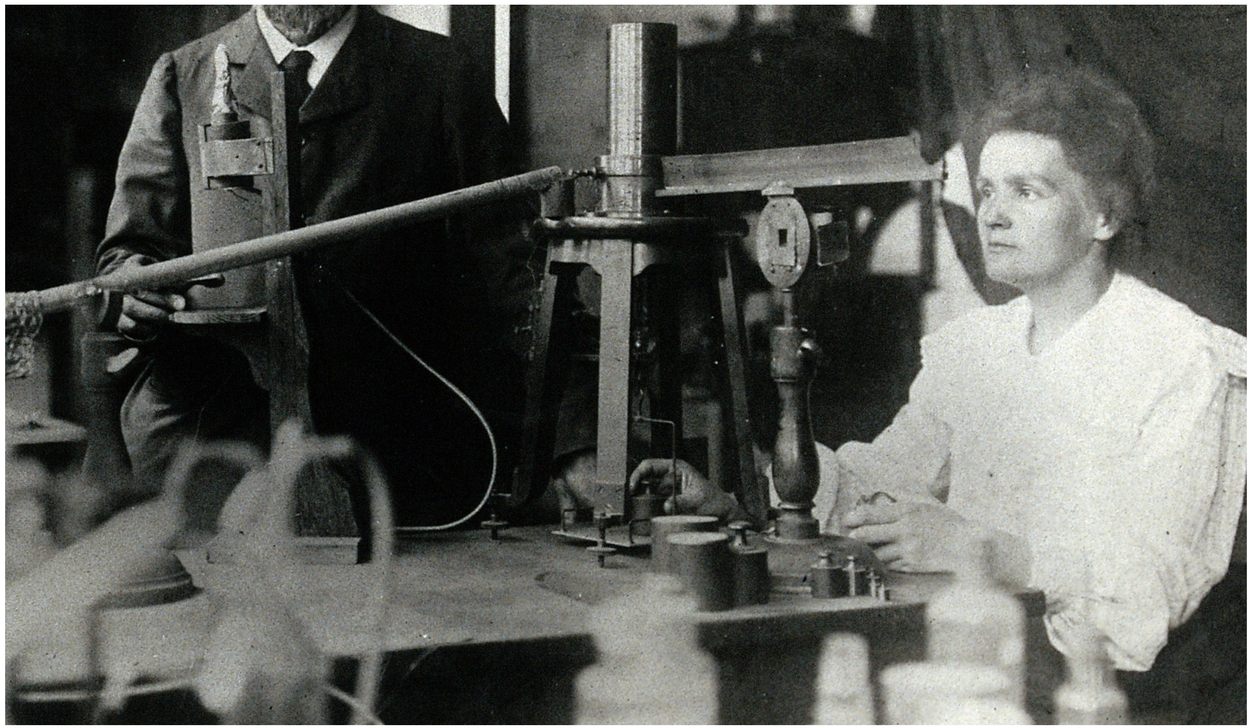
Neural Networks

$$y \approx \sum_i \alpha_i \cos(\langle v_i, x \rangle + b_i)$$

v_i are optimized since $p(w)$ is unknown

Two challenges of random features:

- The choice of p
- Curse of dimensionality



wikipedia: Marie Curie

Experiments

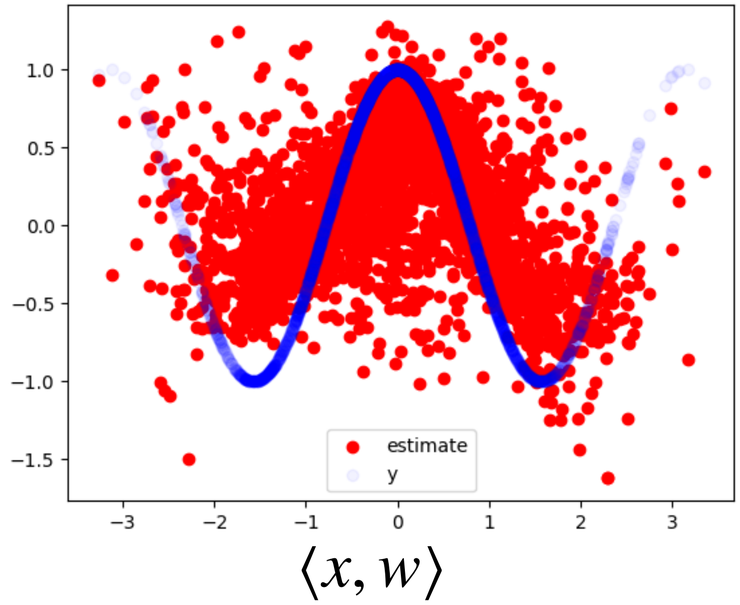
Get ready for hands-on group activity

4-dimensional case

- $y = \cos(\langle w, x \rangle), x, w \in \mathbb{R}^4$
- Estimate with 100 random feature

y

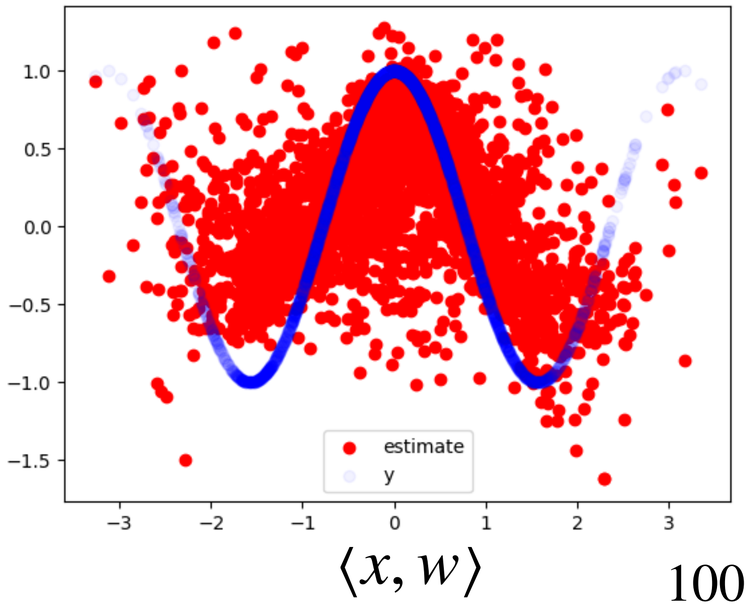
f_{100}



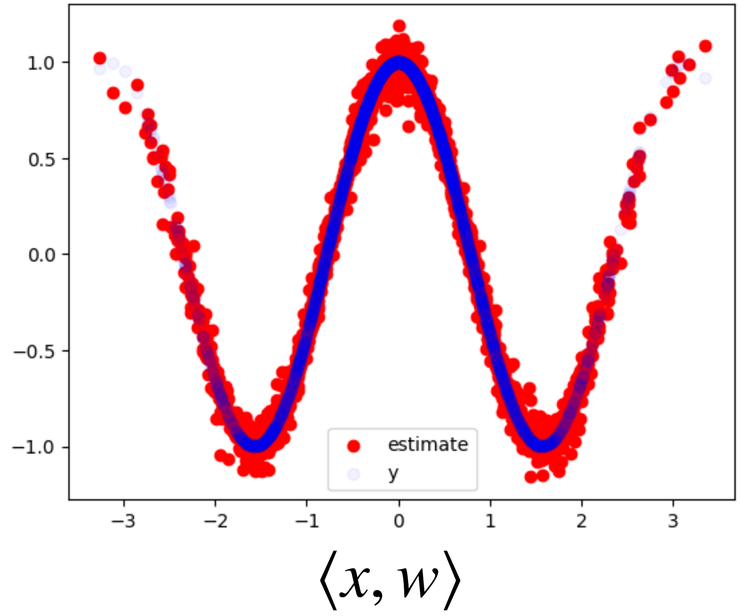
Task: improve random feature estimate

y

f_{100}



How?



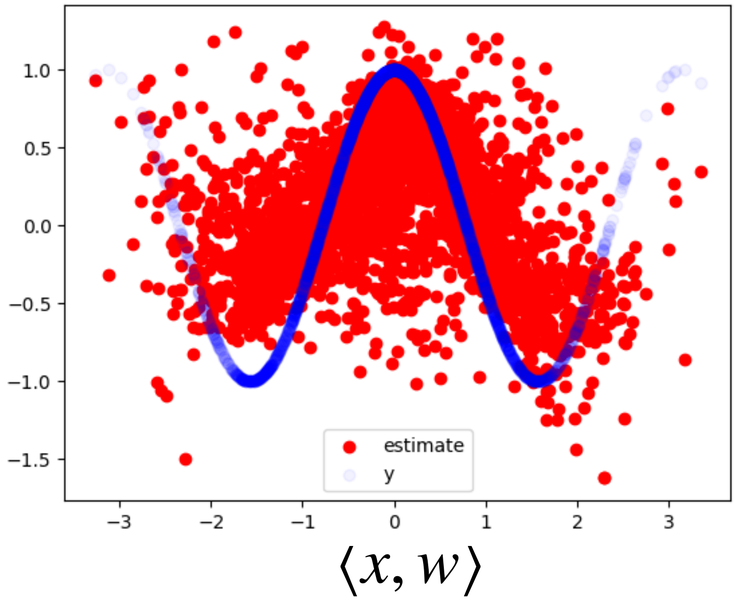
$$y \approx \underbrace{\sum_{i=1}^{100} \alpha_i \cos(\langle v_i, x \rangle + b_i), v_i, x \in \mathbf{R}^d}_{f_{100}(x)}$$

Task: increase the number of features to 1000

Colab: https://shorturl.at/mWTLt

y

f_{100}



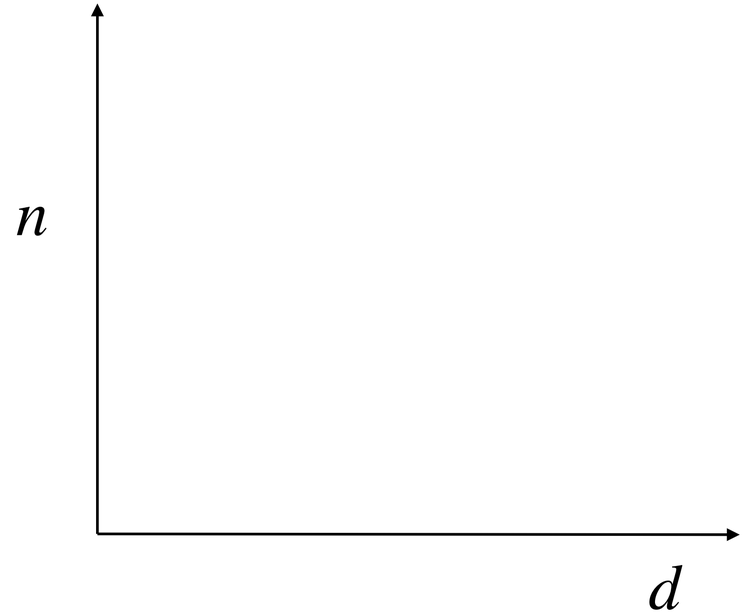
?

https://colab.research.google.com/drive/1SzLEF2EEMhjAXZkx86GuRxIwSkeTKbvs#scrollTo=V5xanwu8fDBt

Task 2: observing the curse of dimensionality

$$f(x) \approx \underbrace{\sum_i \alpha_i \cos(\langle v_i, x \rangle + b_i)}_{f_n(x)}, v_i, x \in \mathbf{R}^d$$

- $\mathbb{E}\left[(f(x) - f_n(x))^2\right] \leq 0.0005$
- How large n for $d = 1,2,3,4,5$

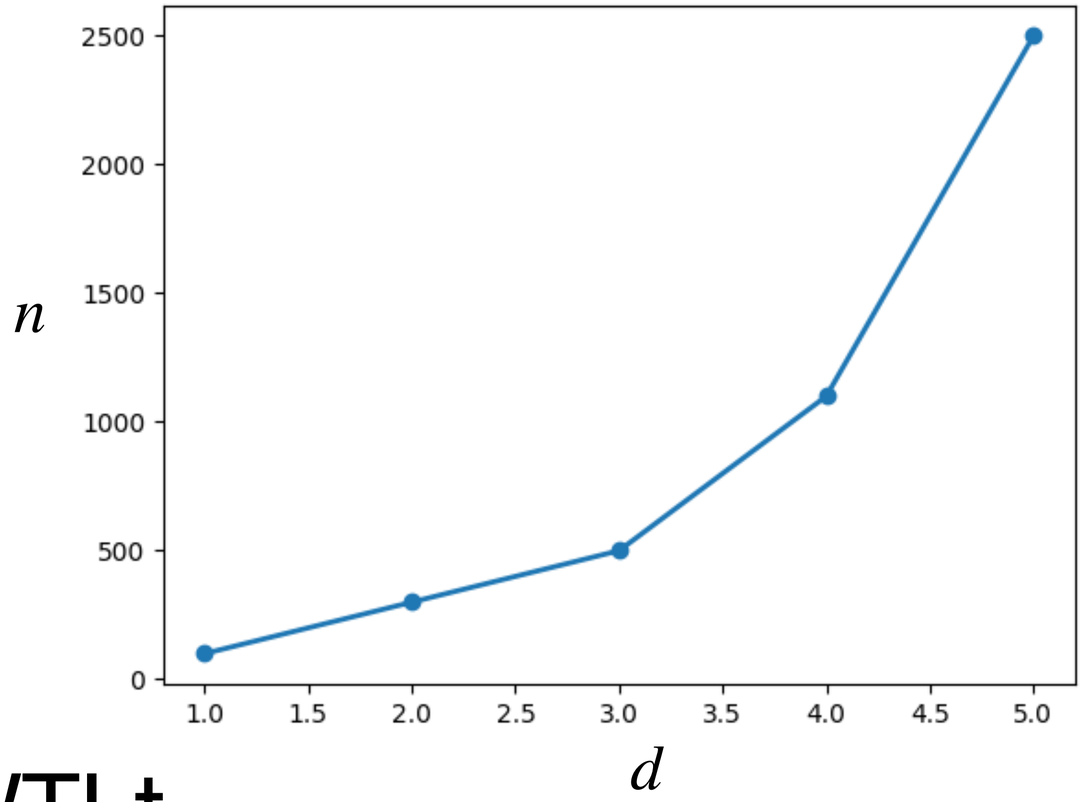


Colab: https://shorturl.at/mWTLt

Solution

$$f(x) \approx \underbrace{\sum_i \alpha_i \cos(\langle v_i, x \rangle + b_i)}_{f_n(x)}, v_i, x \in \mathbf{R}^d$$

- $\mathbb{E}\left[(f(x) - f_n(x))^2\right] \leq 0.0005$
- How large n for $d = 1,2,3,4,5$



Colab: https://shorturl.at/mWTLt

Theory vs Lab

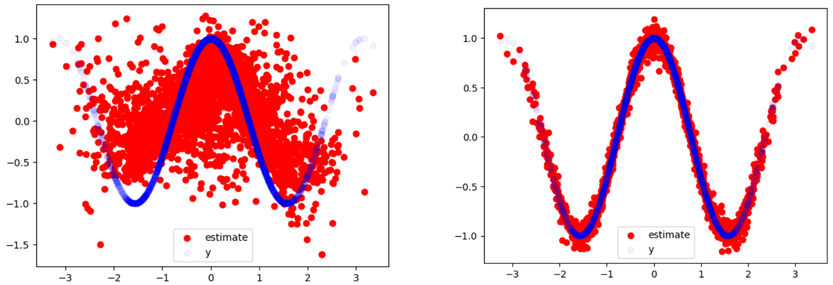
Theory (last lecture)

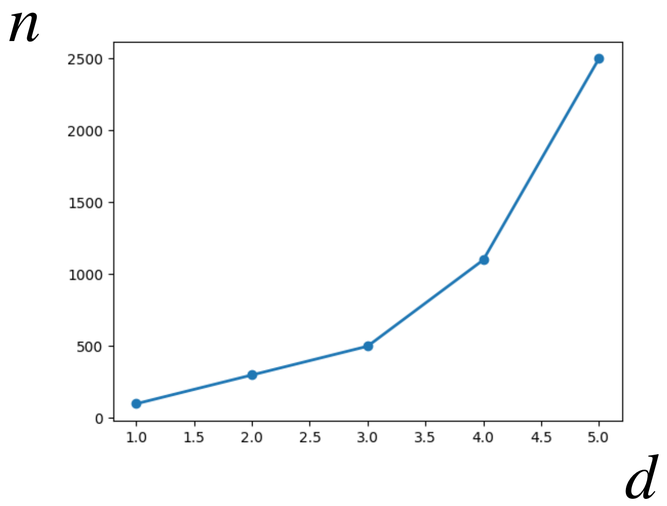
$$\mathbb{E}\left[\max_{i\le n}\langle v_i, w\rangle^2\right] \approx \frac{\log(n)}{d}$$

- Order statistics
- Memoryless property
- Integral bound for $\sum_i \frac{1}{i}$

Lab

$y = \cos(2 * \langle w, x \rangle)$





Connecting theory and lab

- $y = \cos(\langle w, x \rangle) \approx \sum_i \alpha_i \cos(\langle v_i, x \rangle)$
- To approximate y, there must exists a v_i such that $\|w - v_i\| \leq \epsilon$ Why?
  - $\cos(\langle v_i, x \rangle) - \cos(\langle w, x \rangle) \leq -2 \sin\left(\frac{\langle v_i + w, x \rangle}{2}\right) \sin\left(\frac{\langle v_i - w, x \rangle}{2}\right)$
  - $|\cos(\langle v_i, x \rangle) - \cos(\langle w, x \rangle)| \leq \|x\| \max\{\|w - v_i\|, \|w + v_i\|\}$
- $|\langle w, v_i \rangle| \geq 1 - \epsilon$

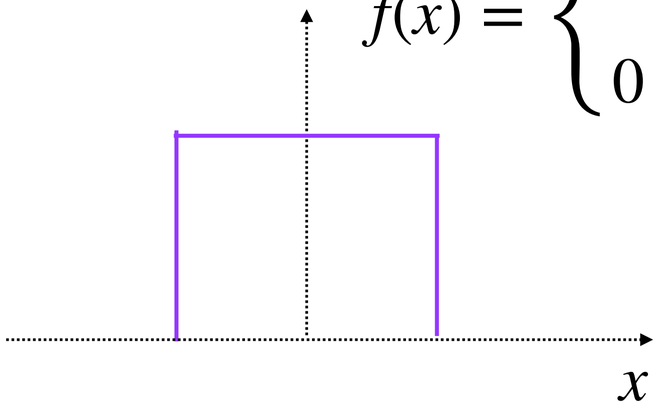
Barron's function class

A. R. Barron, "Universal approximation bounds for superpositions of a sigmoidal function," in *IEEE Transactions on Information Theory*, 1993

Barrons' function: $f(x) = \int e^{i\langle w,x\rangle} \hat{f}(w) dw$ such that $C_f = \int |w| |\hat{f}(w)| dw < \infty$

- For Barrons' functions, neural networks break curse of dimensionality
- Question: Is the step function (see bellow) a Barrons' function?

$$f(x) = \begin{cases} 1 & |x| \leq \frac{1}{2} \\ 0 & \text{otherwise} \end{cases} \implies \hat{f}(w) = \frac{\sin(\pi w)}{\pi w}$$



Thank you very much!