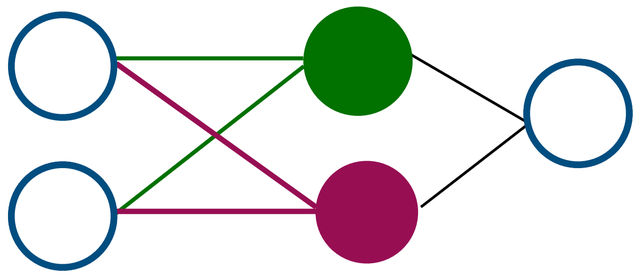
Neural Networks: A Theory Lab

Lecture 6 Optimization and gradient descent



Hadi Daneshmand

University of Virginia

Thank you so much, Po-Wei

- Po-Wei Chen scribed lectures for curse of dimensionality

https://hackmd.io/D_xMlINMQau_QhfYWnEVag

grow exponentially, $n \geq \exp(d/2)$, to counteract the sparsity of high-dimensional spaces.

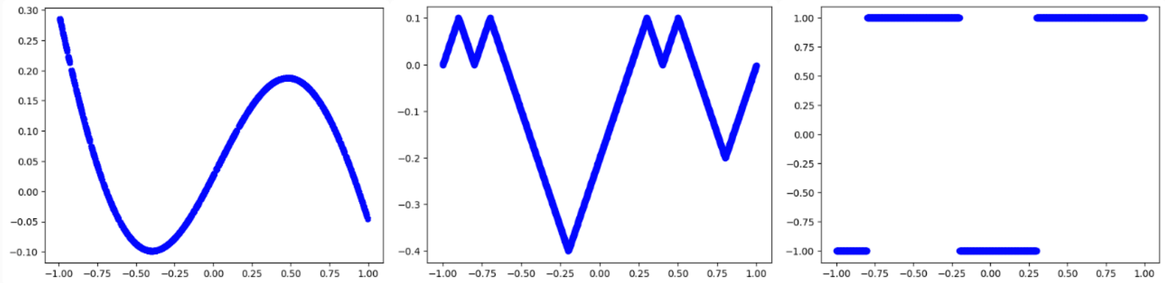
Lab Observation

Recap: Universal Non-Linear Feature Approximation

We want to design universal non-linear features to approximate any function efficiently.

A universal function approximator follows the form:

$$y_i = \sum_j w_j \varphi_j(x_i),$$



We discussed the representer theorem, which allows us to design (n) features for (n) samples to approximately fit all three of the above functions using a linear combination of the same features. However, a key challenge is feature selection: how do we construct a function approximation without requiring an exponential number of basis functions?

Now, we aim to reduce the number of features from (n) to 20. To approximate a target function efficiently, we utilize random features:

$$\phi(x) = \cos(\langle v, x \rangle + b).$$

Homework 1

- Good news: you can now practice the first homework
- Webpage: https://hackmd.io/CVEumpNjTOGZYxhLfpnSYA
- Groups: 1-5
- Deadline: Feb 20
- Questions about HW?
  - post comments online
  - Ask at the beginning of the next lecture
  - Office hours: Monday 1:30-2:30pm and Tuesdays 4-5pm

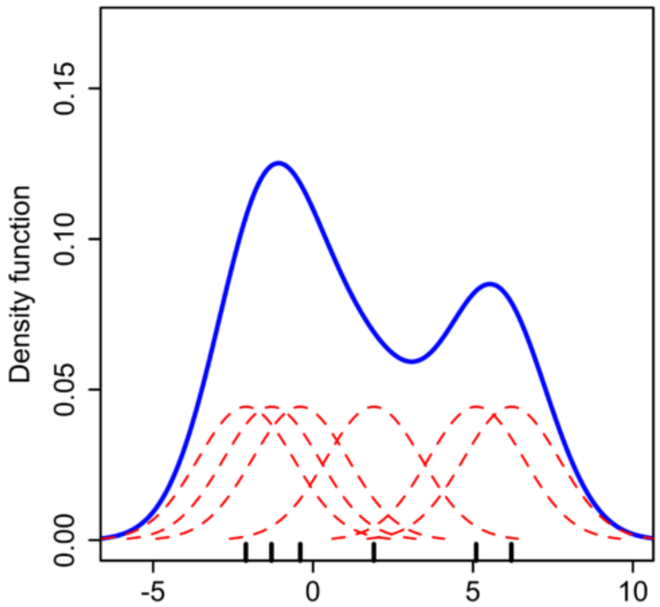
Recap

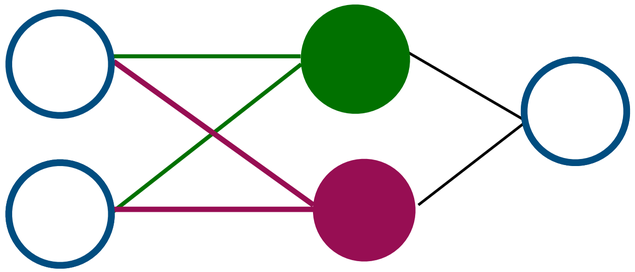
Machine learning universal tools

Kernel methods

Random features

Neural Nets





Kernel methods vs random features

Random Features

$$y \approx \sum_{i=1}^{m} \alpha_i \cos(\langle w^{(i)}, x \rangle + b_i)$$

$O(mn^2)$ computation

Approximate functions obeying reproducing property with an additional condition

Kernel Methods

$$y \approx \sum_{i=1}^{n} \alpha_i k(x, x_i)$$

$O(n^3)$ computation

Approximate functions obeying reproducing property

Barrons' comparison

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

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

Random Features

w_i independent of $f(x)$

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

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

Neural Networks

w_i are optimized for $f(x)$

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

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

Barrons' comparison

Random Features

$w^{(i)}$ independent of y

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

The exponential grow in n with d is called curse of dimensionality

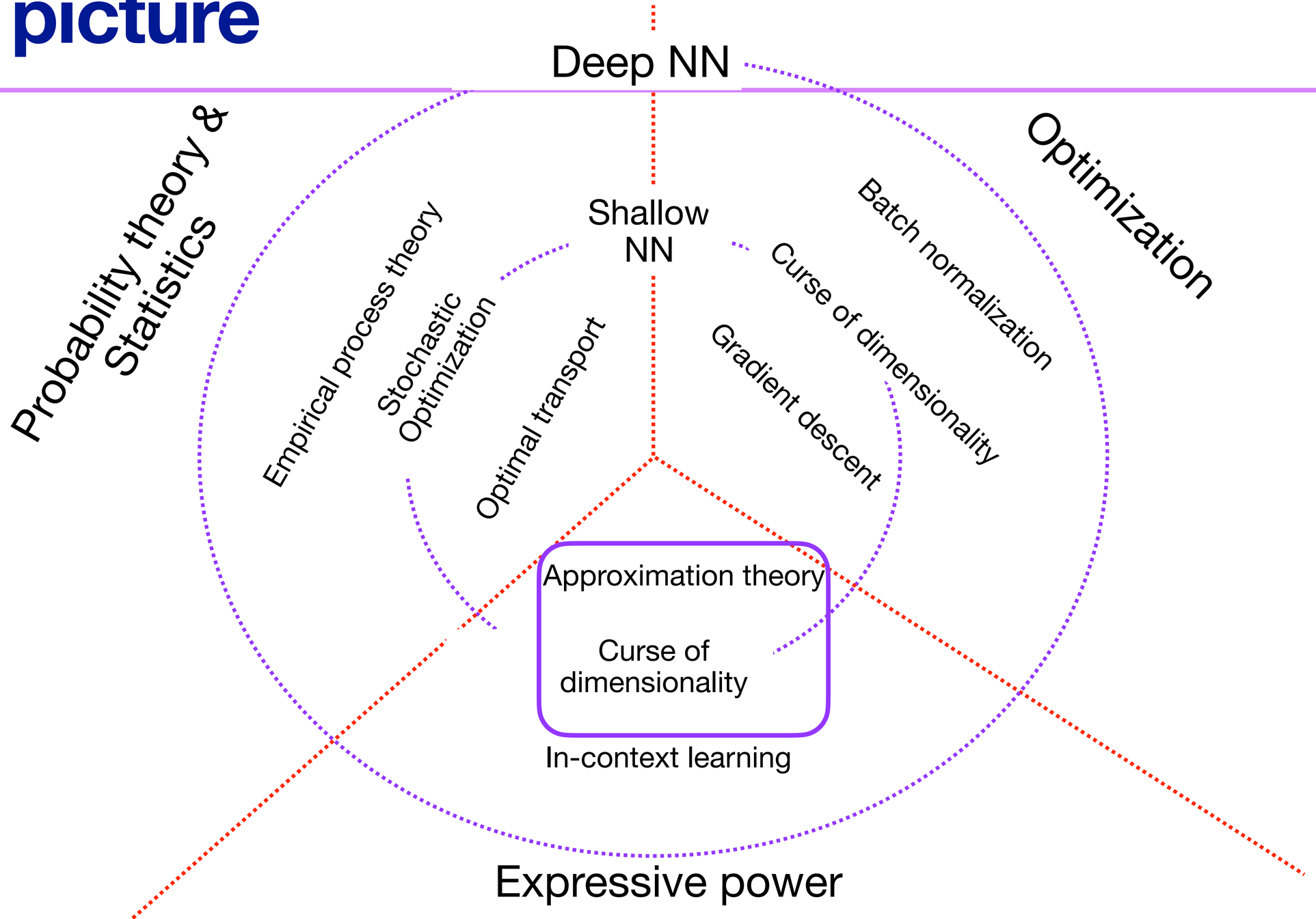
$n = \Omega((\frac{1}{\epsilon})^d)$ for ϵ accuracy

Neural networks can break curse of dimensionality

Poll 1

Do you want to know the proof of Barron’s theorem?

Big picture


Deep NN
Probability theory & Statistics
Optimization
Shallow NN
Empirical process theory
Stochastic Optimization
Optimal transport
Gradient descent
Curse of dimensionality
Batch normalization
Approximation theory
Curse of dimensionality
In-context learning
Expressive power

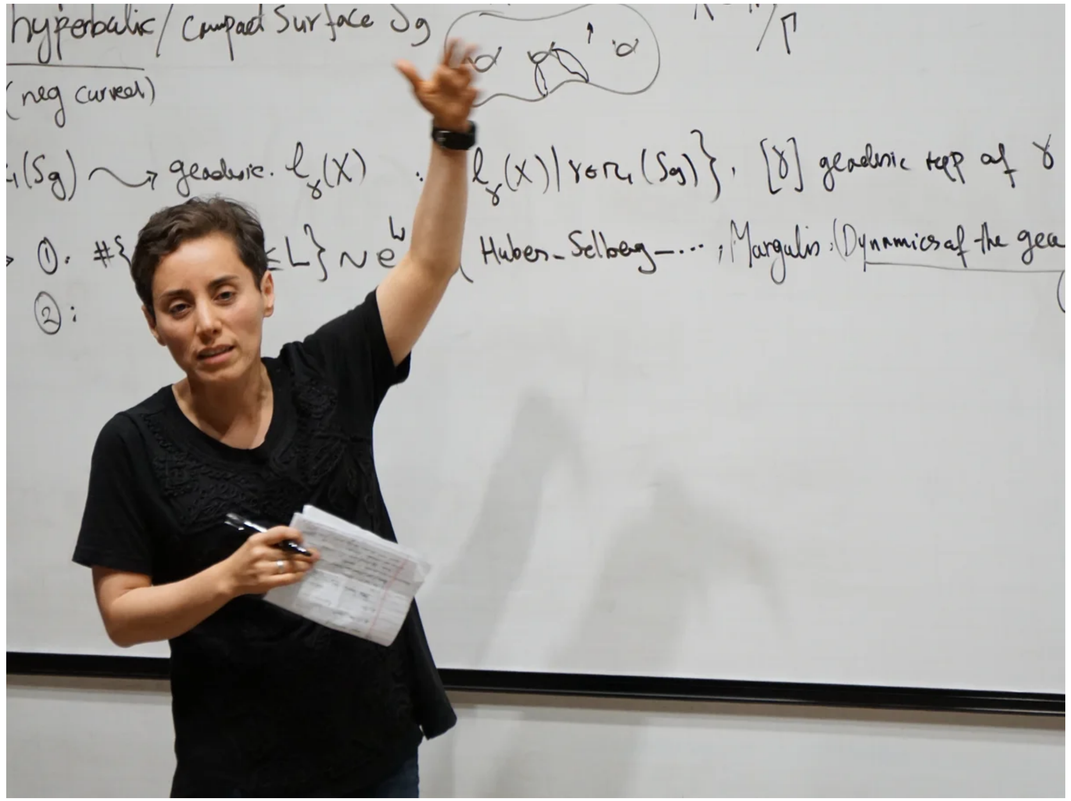
How to optimize?

- Neural Networks: $\min_{\alpha_i, w^{(i)}, b_i} \mathbb{E}\left(\sum_i \alpha_i \cos(\langle w^{(i)}, x \rangle + b_i) - y \right)^2$
- General problem: $\min_{x \in \mathbb{R}^d} f(x), \qquad f : \mathbb{R}^d \to \mathbb{R}$
- Reference: Introductory lecture on convex optimization by Nesterov (Chapter 1)

Plan: Optimization and Neural Nets

- Curse of Dimensionality & Gradient Descent
- Convex Optimization and **Language Models**
- The non-convexity challenge
- Constrained Optimization and Optimal Transport
- Provable sorting with **Transformers**
- **In-context Learning**
- Implicit Convexity and Optimal Transport
- Batch normalization
- Saddle point optimization and generative models

- Recap
- Theory
- Experiments

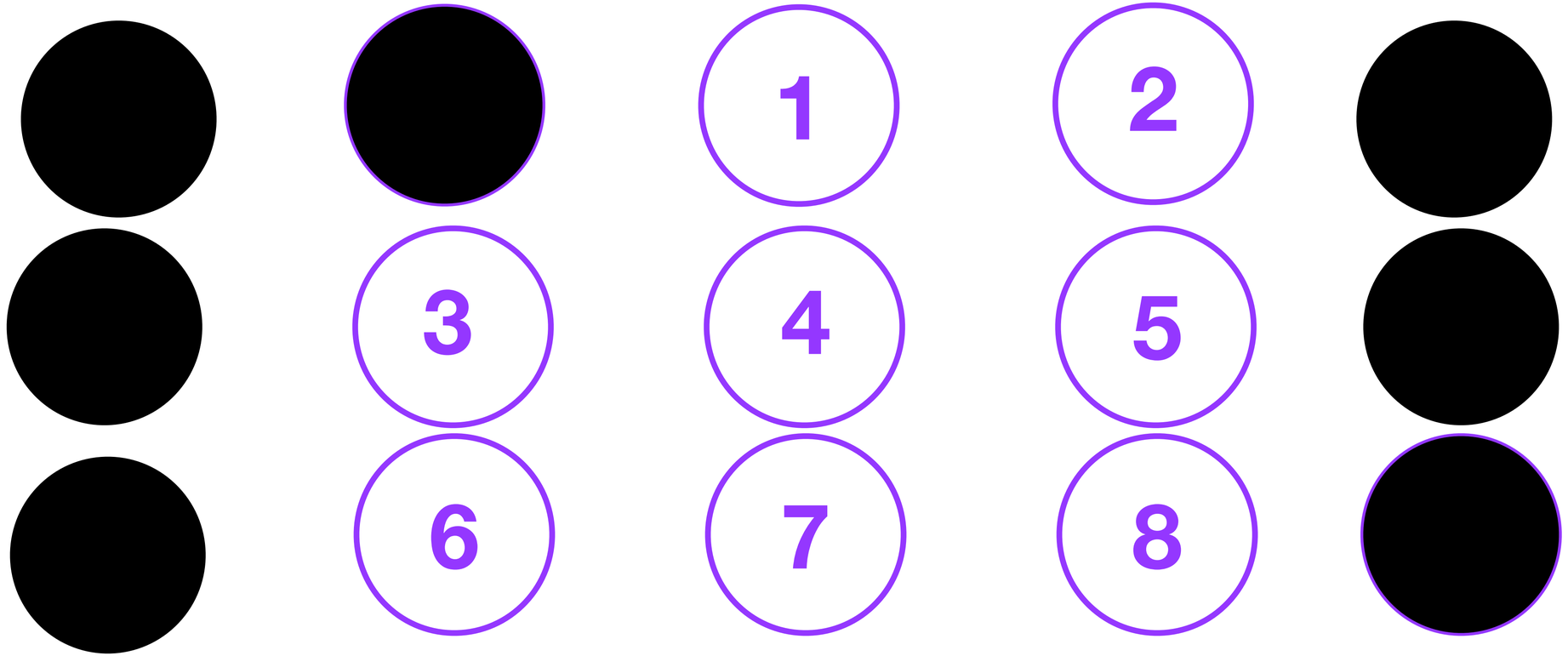


theguardian: Maryam Mirzakhani

Theory

Get ready for math

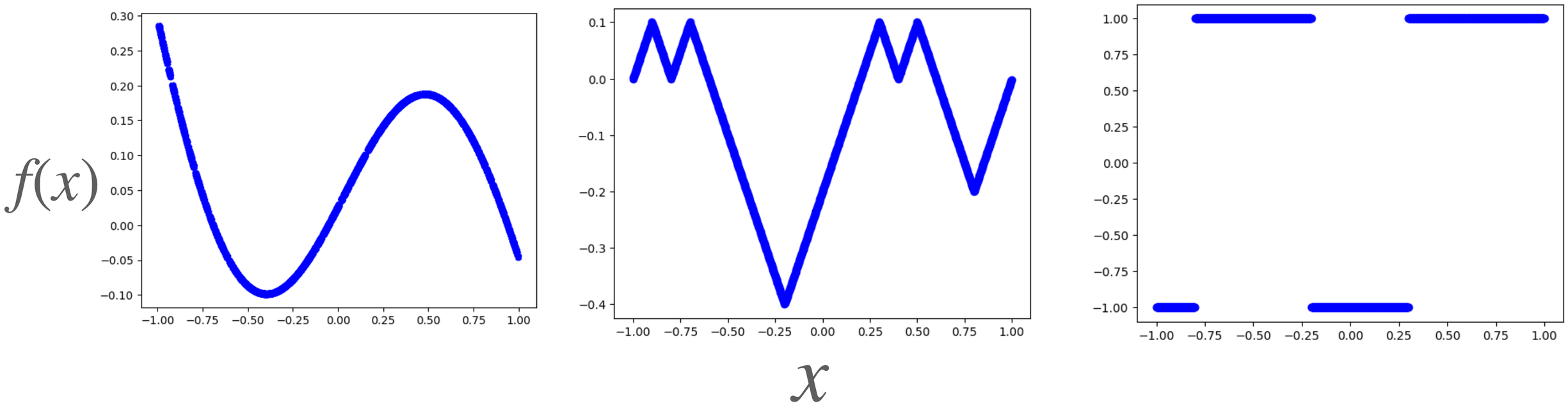
Group assignment



I am here

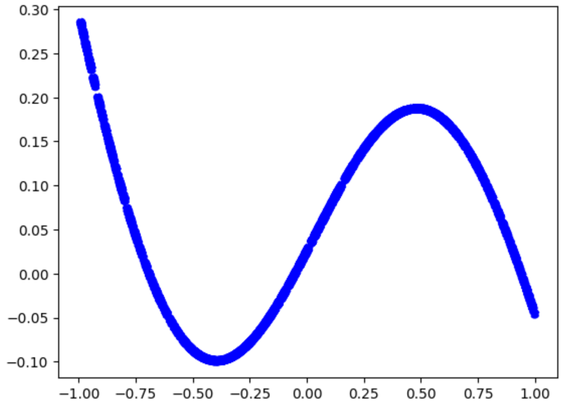
Background: Lipschitz continuity

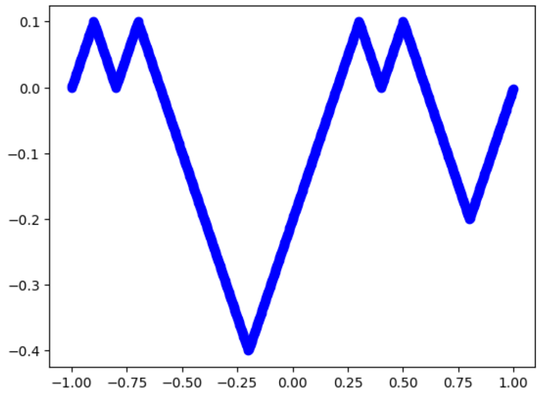
- Lipschitz continuity: $|f(x) - f(y)| \leq L\|x - y\|$
- Question: Which function is Lipschitz?

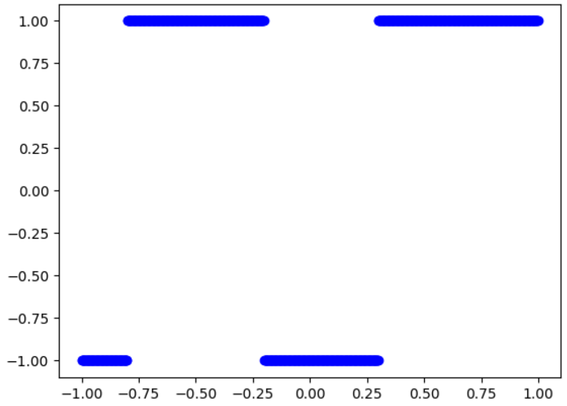


Smooth functions

- Gradient: $\nabla f(x) = \begin{bmatrix} \frac{\partial}{\partial x_1} f(x) \\ \vdots \\ \frac{\partial}{\partial x_d} f(x) \end{bmatrix}$, Hessian: $\nabla^2 f(x) = \begin{bmatrix} \frac{\partial^2}{\partial x_1 \partial x_1} f(x) & \dots & \frac{\partial^2}{\partial x_1 \partial x_d} f(x) \\ \vdots & \vdots & \vdots \\ \frac{\partial^2}{\partial x_d \partial x_1} f(x) & \dots & \frac{\partial^2}{\partial x_d \partial x_d} f(x) \end{bmatrix}$
- $f(x)$ is β-smooth if twice differentiable and $\|\nabla^2 f(x)\| \leq \beta$
- Question: Which function is smooth?

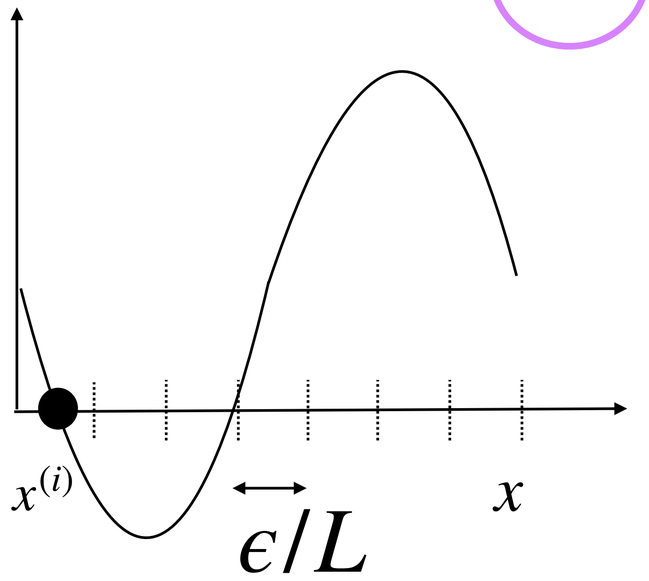






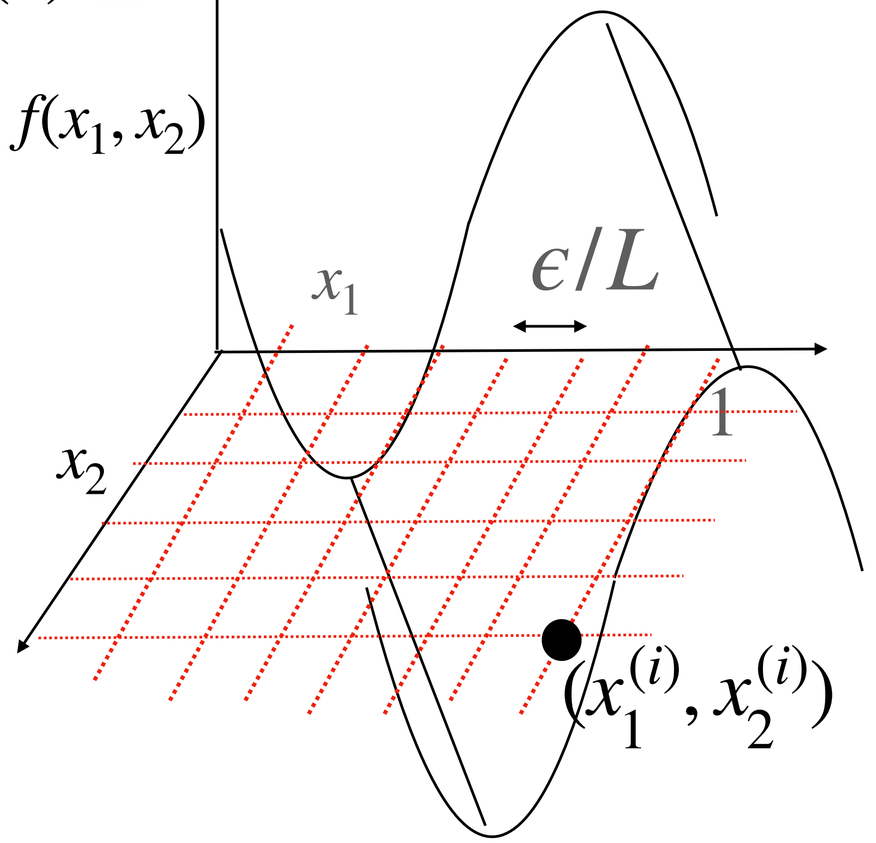
Task 1: optimization of one variable

- General problem: $\min_{x \in [0,1]} f(x)$
- Lipschitz continuity: $|f(x) - f(y)| \leq L\|x - y\|$
- Question: How to find x_0 such that $f(x_0) - \min_{x \in [0,1]} f(x) \leq \epsilon$?
- Solution: Make a grid of size $\frac{L}{\epsilon}$ and return $x_0 = \min_{i \in \{1,\ldots,1/\epsilon\}} f(x_i)$
  - $x_* := \arg\min_x f(x),\ \exists i : x_* \in (x^{(i)}, x^{(i+1)})$
  - $\min f(x^{(i)}) - f(x_*) \leq f(x^{(i)}) - f(x_*) \leq L\|x^{(i)} - x_*\| \leq L\|x^{(i)} - x^{(i+1)}\| \leq \epsilon$



Task 2: Optimization of two variables

- Question: How many query to find $f(x) - \min_{x \in [0,1]^2} f(x) \leq \epsilon$?
- Solution:
  - Make a 2D grid of size $\frac{L}{\epsilon}$
  - Return $\min_{i \leq L^2/\epsilon^2} f(x_1^{(i)}, x_2^{(i)})$
- Conclusion: $\frac{1}{L^2\epsilon^2}$ queries are needed



Thank you very much Wenny Xie's for pointing out to a mistake

Curse of dimensionality for optimization

	1D	2D	d-D
Grid size	$\frac{1}{\epsilon}$	$\frac{1}{\epsilon^2}$	$\frac{1}{\epsilon^d}$

To find x_0 such that$f(x_0) - \min_{x \in [0,1]^d} f(x)$, we need to evaluate $f(x_i)$ $O(\frac{L^d}{\epsilon^d})$ times

Curses of dimensionality

Function approximation

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

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

Optimization

$$f(x_0) - \min f(x) \leq \epsilon$$

$$\frac{1}{\epsilon^d} \text{ for } \epsilon-\text{accuracy}$$

Smarter than grid search

- Assume: f is β-smooth: $\|\nabla^2 f(x)\| \leq \beta$
- **Statement:** Given $x_0 \in \mathbb{R}^d$, there is x_1 in a neighborhood of x_0 such that

$$f(x_1) \leq f(x_0) - \frac{1}{\beta}\|\nabla^2 f(x_0)\|^2$$

- Conclusion: We can iteratively decrease the function

$$f(x_2) \leq f(x_1) - \frac{1}{\beta}\|\nabla^2 f(x_1)\|^2 \leq f(x_0) - \frac{1}{\beta}\left(\|\nabla^2 f(x_1)\|^2 + \|\nabla^2 f(x_0)\|^2\right)$$

A Quadratic Upper-bound

- Assume: f is twice differentiable and $\|\nabla^2 f(x)\| \leq \beta$
- Question: Let $g(x) = f(x_0) + \langle \nabla f(x_0), x - x_0 \rangle + \frac{\beta}{2}\|x - x_0\|^2$, show

$$f(x) \leq g(x)$$

- Hint: Use Taylor remainder theorem:
  $\exists t \in [0,1] : f(y) = f(x) + \nabla f(x)[y - x] + \frac{1}{2}\nabla^2 f(tx + (1-t)y)[y - x, y - x]$

Task 3: Finding the descent direction

- Recall: $g(x) = f(x_0) + \langle \nabla f(x_0), x - x_0 \rangle + \frac{\beta}{2}\|x - x_0\|^2$
- Question: What are $\arg\min_x g(x)$ and $\min g(x)$?
- Solution: $\underbrace{\arg\min g}_{x_1} = x_0 - \frac{1}{\beta}\nabla f(x_0), f(x_1) \leq g(x_1) = f(x_0) - \frac{1}{2\beta}\|\nabla f(x_0)\|$

Gradient descent

- Stitching with problem 1: $\underbrace{f(x_1) \leq g(x_1)}_{prob.1} \underbrace{\leq f(x_0) - \frac{1}{2\beta}\|\nabla f(x_0)\|^2}_{prob.2}$

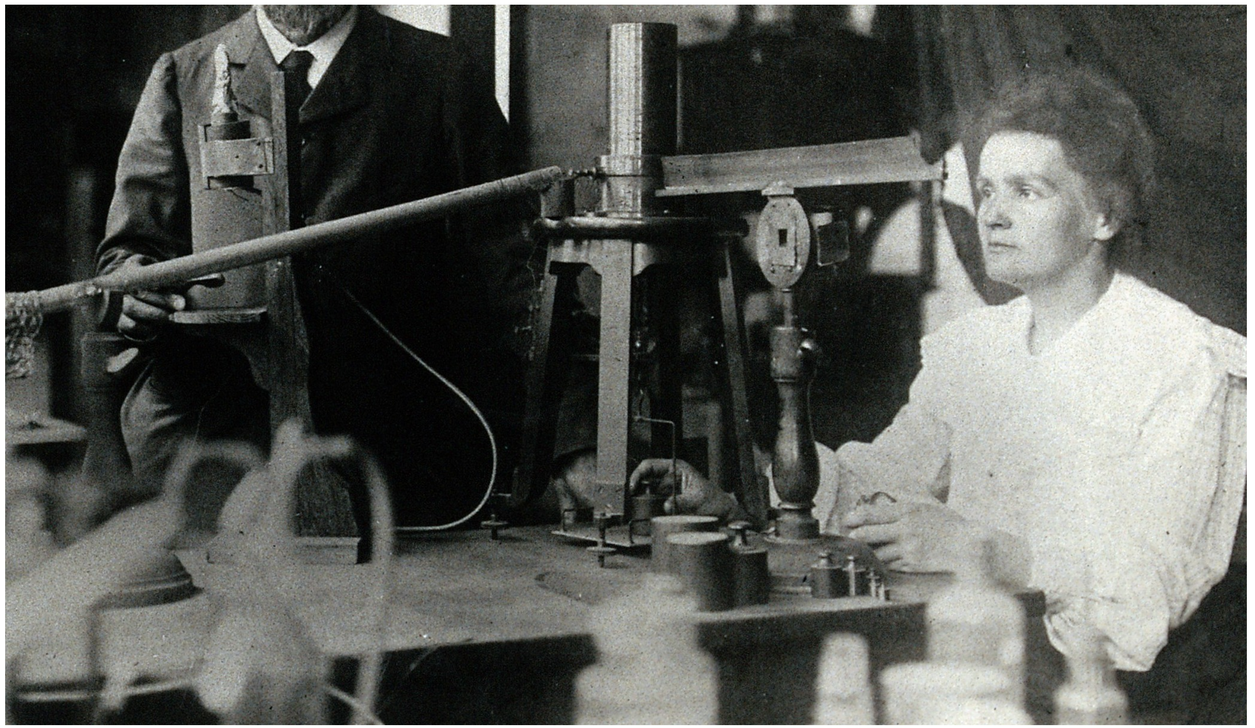
Does GD suffer from curse of dim for optimization?

	1D	2D	d-D
Grid search	$\frac{1}{\epsilon}$	$\frac{1}{\epsilon^2}$	?

GD: $x_{k+1} = x_k - \frac{1}{\beta}\nabla f(x_k)$

To find x_k such that $f(x_k) - \min f(x) \leq \epsilon$, how large k has to be?

- A brief recap
- Theory
- Experiments



wikipedia: Marie Curie

Experiments

Get ready for hands-on group activity

Gradient descent implementation

- $c = a + b$
- $d = b + 1$
- $e = c * d$
- Question: what is de/db?
- Automatic differentiation packages: PyTorch and TensorFlow

Step 1: Variables

- $c = a + b$
- $d = b + 1$
- $e = c * d$
- Question: what is de/db?
- Automatic differentiation packages: PyTorch and TensorFlow

PyTorch Code

```
import torch
a = torch.randn(1,requires_grad=True)
b = torch.randn(1,requires_grad=True)
```

Step 2: Computation

- $c = a + b$
- $d = b + 1$
- $e = c * d$
- Question: what is de/db?
- Automatic differentiation packages: PyTorch and TensorFlow

PyTorch Code

```
import torch
a = torch.randn(1,requires_grad=True)
b = torch.randn(1,requires_grad=True)
c = a + b
d = b+1
e = d*c
```

Step 3: Differentiation

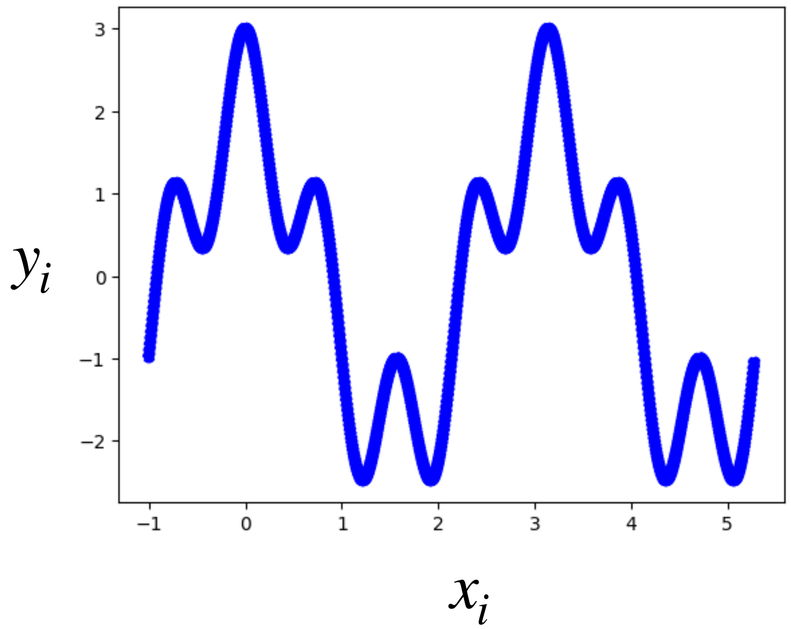
- $c = a + b$
- $d = b + 1$
- $e = c * d$
- Question: what is de/db?
- Automatic differentiation packages: PyTorch and TensorFlow

PyTorch Code

```
import torch
a = torch.randn(1,requires_grad=True)
b = torch.randn(1,requires_grad=True)
c = a + b
d = b+1
e = d*c
torch.autograd.grad(e,b)[0]
```

Settings

- $f(x) = 2\cos(2x) + \frac{1}{2}\cos(8x)$ where
- Generating $\{(x_i, y_i = f(x_i))\}_{i=1}^{1000}$



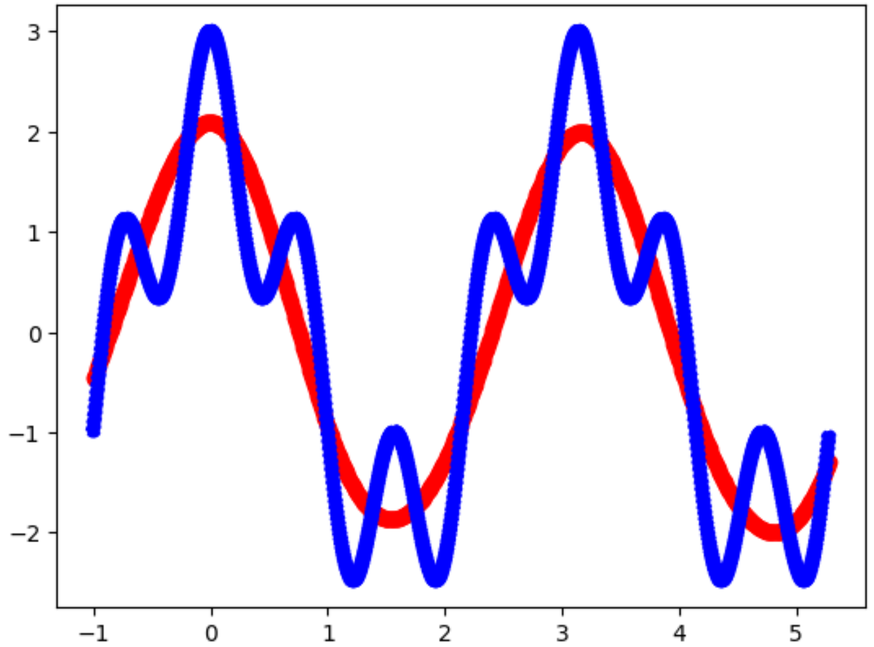
Your task: implement training of neural networks

$$\min_{w_1,w_2,b_1,b_2,\alpha_1,\alpha_2} \frac{1}{1000} \sum_{i=1}^{1000} (y_i - \sum_{i=1}^{m} \alpha_i \cos(w_i x + b_i))^2$$

- Seed Code: **https://shorturl.at/TM3OE**

https://colab.research.google.com/drive/17wq7bsnW8OaJfE0iLsVv2ly_IUFx66UM?usp=sharing

Result: poor estimation

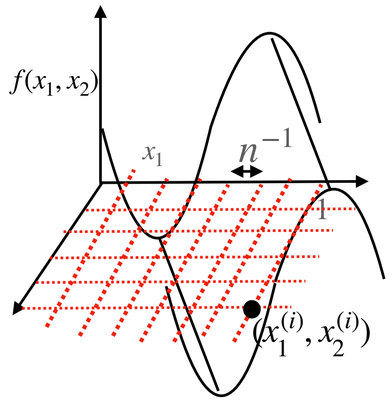


Summary

Theory

Curse of dimensionality for optimization

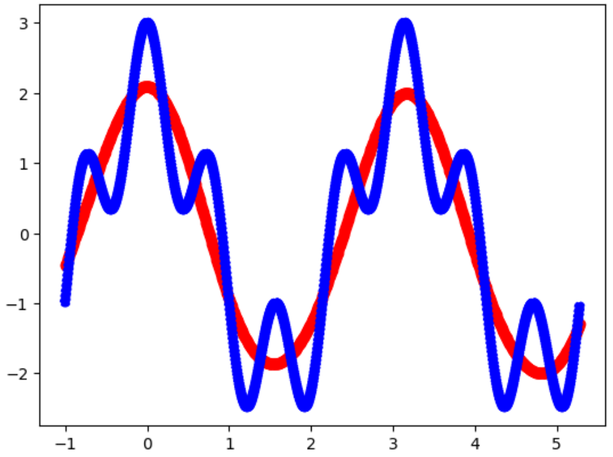
Gradient descent method



Observation

Implementing GD for neural nets

GD do not recover a good estimate



Does GD suffer from curse of dim for optimization?

	1D	2D	d-D
Grid search	$\frac{1}{\epsilon}$	$\frac{1}{\epsilon^2}$	?

GD: $x_{k+1} = x_k - \frac{1}{\beta}\nabla f(x_k)$

Thank you very much!