

---

---

# Deep Learning

---

---

## — 1. Introduction to Deep Learning —

---

---

# Course Team



Florin  
Brad



Elena  
Burceanu



Marius  
Drăgoi



Florin  
Gogianu



Dragoș  
Boldișor



Ioana  
Pintilie



Andrei  
Manolache



Elisabeta  
Oneață



Ștefan  
Smeu



Antonio  
Bărbălău



Dragoș  
Tânțaru



Cristian  
Păduraru

# Our Team

Team page:  
[bit-ml.github.io](https://bit-ml.github.io)



# AI / ML Community

## Bucharest:

- [Bucharest Deep Learning](#)
- [AI Reading Group @ Unibuc](#)
- [Bucharest AI](#)
- [Computer Vision Reading Group @ UPB](#)

## Events

- [Romanian AI Days](#)
  - Late September
- [EEML Summer School](#)
  - 21-26 July 2025 - Sarajevo, Bosnia & Herzegovina
  - Deadline: 31 March

## General:

- [AI Romania](#)

# Eastern European Machine Learning Summer School

21-26 July 2025, Sarajevo, Bosnia &  
Herzegovina

<https://www.eeml.eu/home>

- BONUS 2 points for acceptance

- Deep Learning lectures
- Tutorial sessions
- Paper reading sessions
- Poster sessions
- Top speakers from: DeepMind, Google, Isomorphic Labs, Apple, Cambridge, MILA, UCL, etc.
- Application
  - Extended abstract
  - Deadline: 31 March 2025

# AI / ML Community

## Research Centers:

- **FMI**: Marius Popescu, Bogdan Alexe, Radu Ionescu, Liviu Dinu et al.
- **UPB**: Traian Rebedea, Marius Leordeanu et al.
- **IMAR**: Marius Leordeanu, Ionel Popescu et al.
- **UT (Cluj)**: Lucian Bușoniu et al.
- **Siemens (Brașov)**: Lucian Itu et al.
- **Bitdefender**: [bit-ml.github.io](https://bit-ml.github.io) team
  - We're looking for (applied) ML interns!

# Administrative

# Program

- **Lectures**

- Each Wednesday: 18:00 - 20:00 (Room 216)
- 3 speed tests
  - 5 minutes
  - 1-2 short questions

- **Labs**

- Each Wednesday: 16:00 - 18:00 (Lab 303)

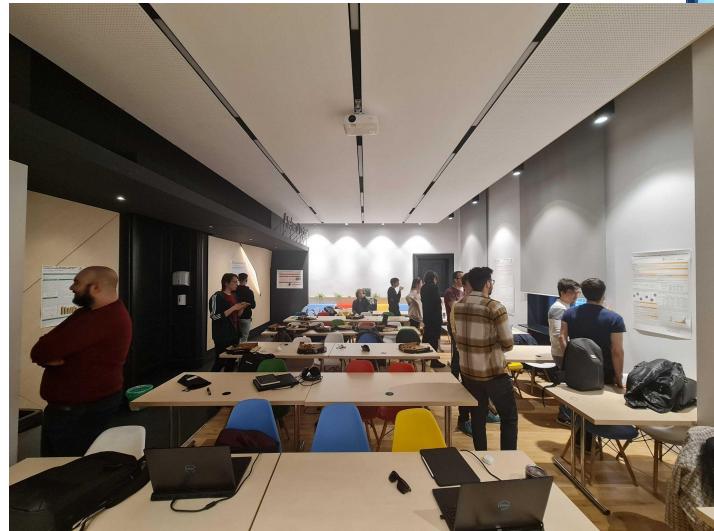
- **Course Materials**

- [Google Drive](#)

# Assignments and Projects

- **2 Assignments (PyTorch)**
  - Related to the lab content
  - Submission deadline: 2 or 3 weeks after publishing
    - penalties for delays
  - Both of them will be presented in Week 9
- **1 Project (PyTorch)**
  - Teams of 2 members
  - Subjects will be proposed in Week 3
  - Each student should choose a subject until Week 6
  - Will be presented in Week 10 - Poster Session

# Poster Session



# Course Structure

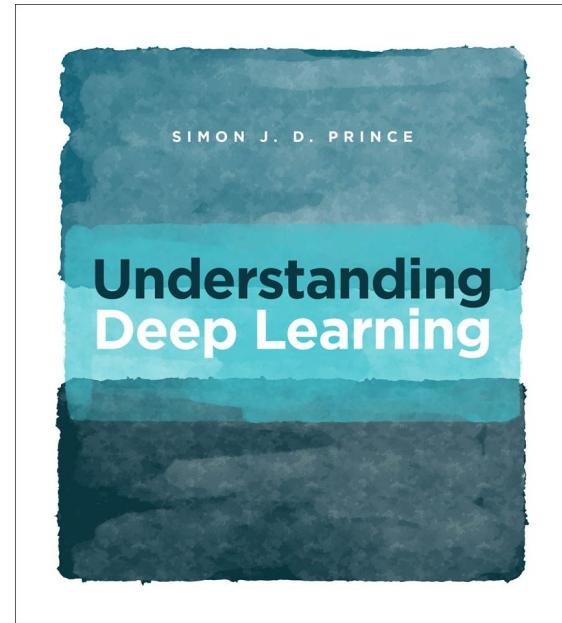
Week	Lecture	Speed Test	Laboratory	Event	Deadline
1	Introduction to Deep Learning		PyTorch and Tensors		
2	Neural Networks		Neural Networks		
3	Optimization		Training and Optimization	Assignment 1 out Project proposals	
4	Debugging Neural Networks	Test 1	Debugging		
5	Convolutional Neural Networks		ConvNets (1)	Assignment 2 out	Assignment 1 EEML- March 31
6	Sequential and set based models (RNN, SSM, TF)	Test 2	Sequential Models		Project selection
7	Natural Language Processing Applications	Test 3	NLP Applications		
8	Computer Vision Applications		ConvNets (2)		Assignment 2
9	Unsupervised Learning		Assignments Evaluation		
10	Project Evaluation (Poster Session)		Project Evaluation		Project

# Grading

		Requirements
Assignment 1	2 points	
Assignment 2	2 points	
3 x Speed Tests	1 point	50% (2.5 points)
[BONUS] Good answers / questions during lectures and labs	max 1 point	
Project	5 points	50% (2.5 points)
[BONUS] Good interventions / questions during the poster session	max 1 point	

# Understanding Deep Learning Book

- A lot of the course and slides is based on “Understanding Deep Learning” by Simon J.D. Prince
- The book is free: <http://udlbook.com/>
- If you want to delve deeper into some concepts, we highly recommend you to read it!
- Special thanks to Emanuela Haller for some of the slides for this lecture



# I. Deep Learning Overview

# Artificial Intelligence

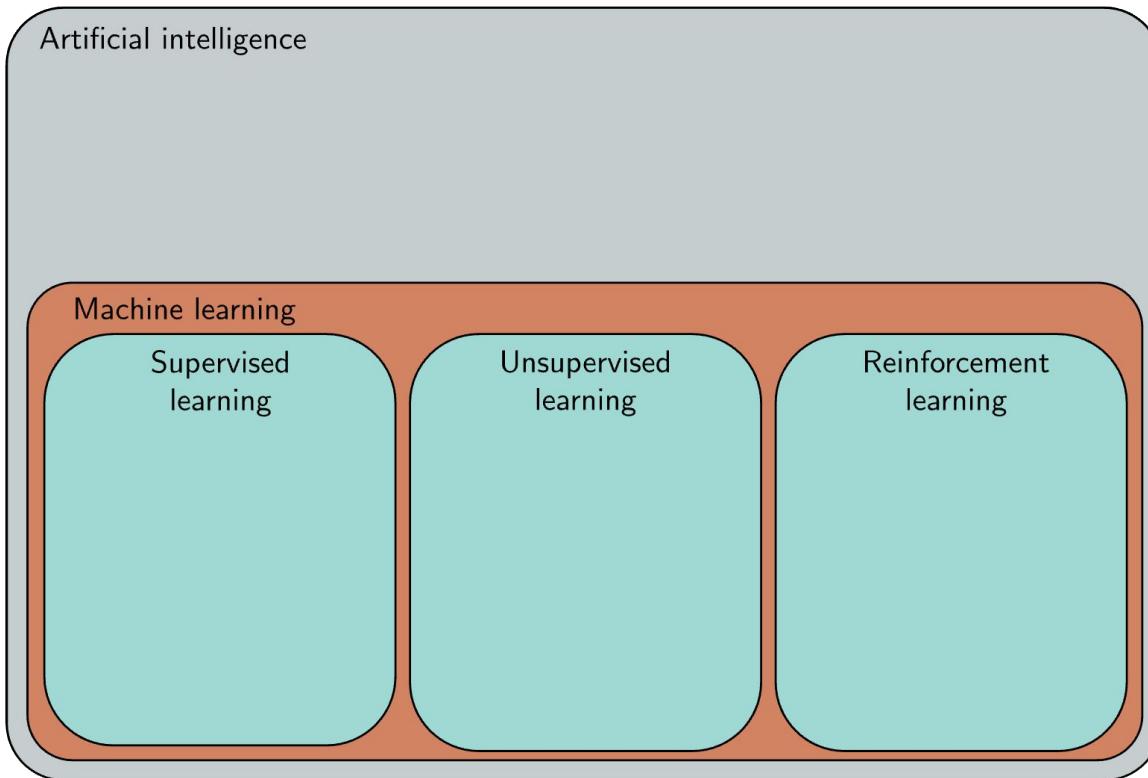
Artificial intelligence

# Artificial Intelligence

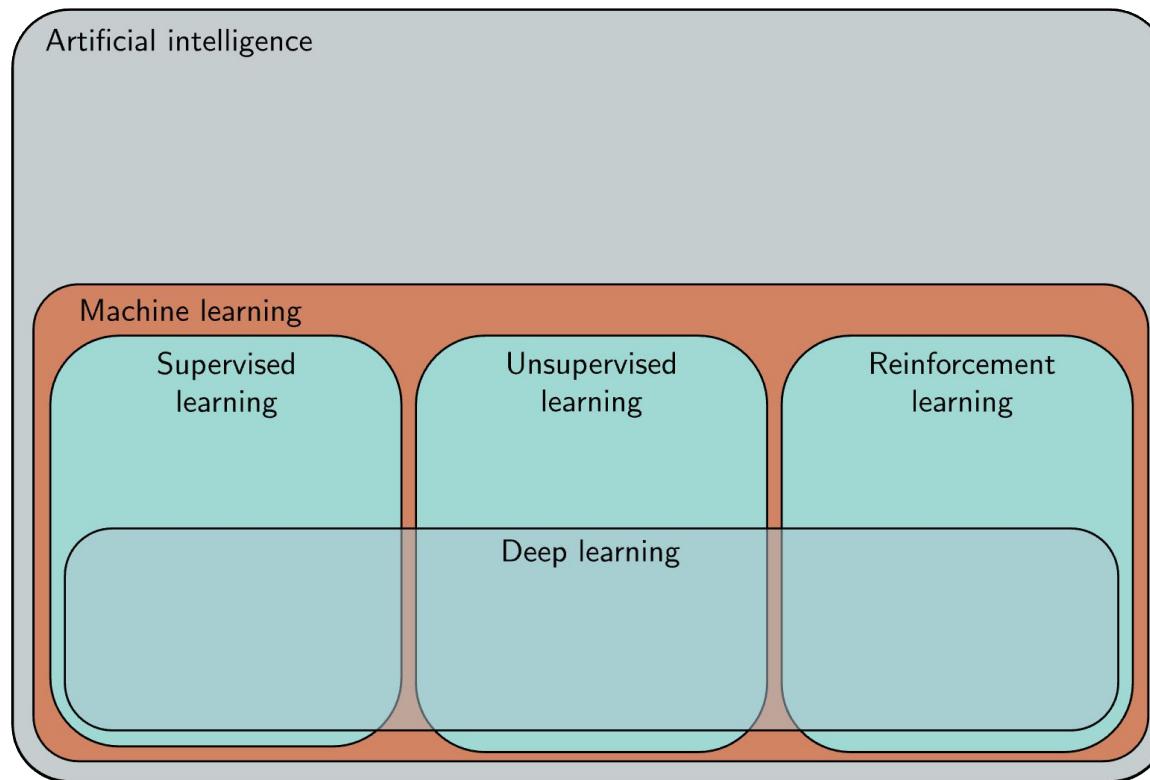
Artificial intelligence

Machine learning

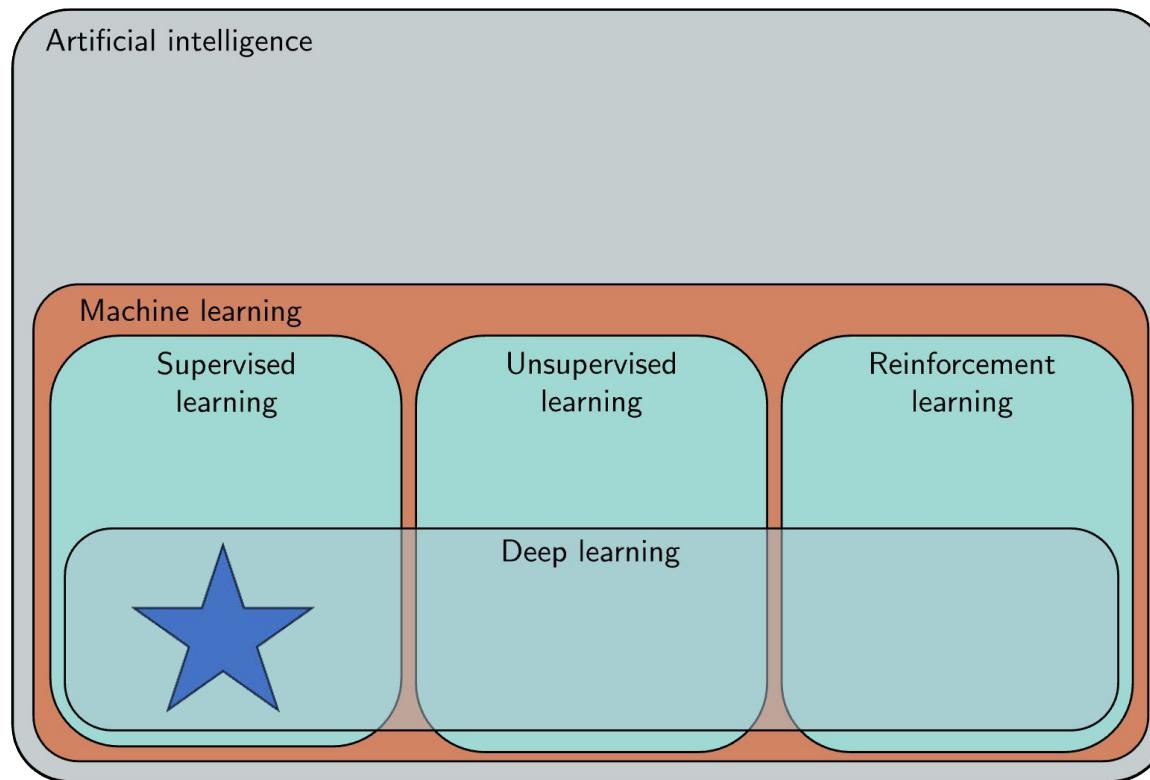
# Artificial Intelligence



# Artificial Intelligence



# Artificial Intelligence



# Supervised Learning

- **Main idea:**
  - Given an input “x” we want to obtain a label “y”
  - We want to “learn” a mapping from x to y

# Supervised Learning

- **Main idea:**
  - Given an input “x” we want to obtain a label “y”
  - We want to “learn” a mapping from x to y
- **Examples:**
  - Does this picture contain a cat or a dog?  $f(\text{picture}) = \text{dog/cat}$
  - Is the sentiment in this text positive or negative?  $f(\text{text}) = \text{positive/negative}$
  - What should be the price of this house?  $f(\text{house}) = [0, 1.000.000] \$$

# Regression

Real world input

6000 square feet,  
4 bedrooms,  
previously sold for  
\$235K in 2005,  
1 parking spot.

Model  
input

$$\begin{bmatrix} 6000 \\ 4 \\ 235 \\ 2005 \\ 1 \end{bmatrix}$$

Model



Model  
output

$$[340]$$

Real world output

Predicted price  
is \$340k

- Given some characteristics, predict one real number
- Linear, fully-connected neural network

# Text classification

Real world input

"The steak was terrible,  
the salad was rotten, and  
the soup tasted like socks"

Model  
input

$$\begin{bmatrix} 8672 \\ 8194 \\ 9804 \\ 8634 \\ 8672 \\ \vdots \end{bmatrix}$$

Model



Model  
output

$$\begin{bmatrix} 0.02 \\ 0.98 \end{bmatrix}$$

Real world output

Positive  
Negative

- Given some text, output the sentiment
- Transformers, Recurrent neural networks

# Image classification

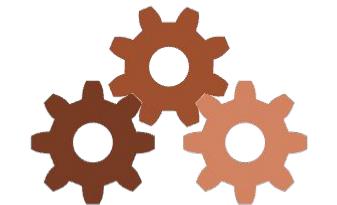
Real world input



Model  
input

$$\begin{bmatrix} 124 \\ 140 \\ 156 \\ 128 \\ 142 \\ 157 \\ \vdots \end{bmatrix}$$

Model



Model  
output

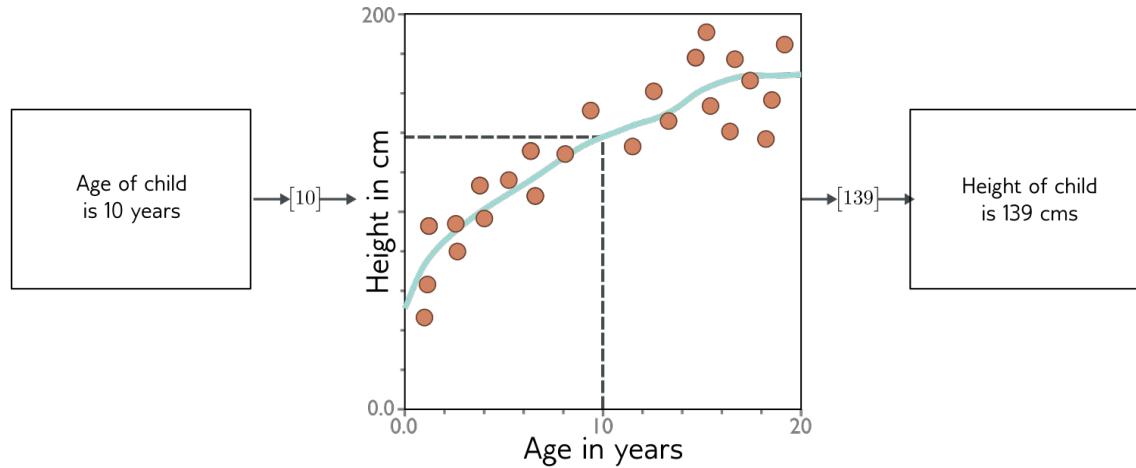
$$\begin{bmatrix} 0.00 \\ 0.00 \\ 0.01 \\ 0.89 \\ 0.05 \\ 0.00 \\ \vdots \\ 0.01 \end{bmatrix}$$

Real world output

Aardvark  
Apple  
Bee  
Bicycle  
Bridge  
Clown  
⋮

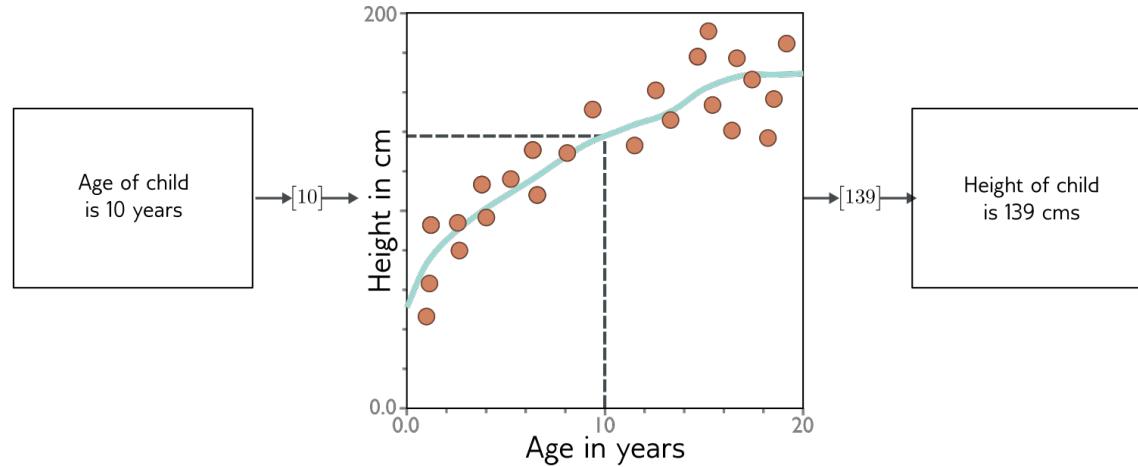
- Given some image, output the sentiment
- Visual Transformers, Convolutional neural networks

# What is the “model”?



- The “model” is an equation relating the input (age) with the output (height)
- We want to search for the equation that best fits our data (learning)

# What is the “model”?



- Deep neural networks are just a family of very flexible equations
- Finding a “good” neural network for the given data = “Deep Learning”

# Image segmentation

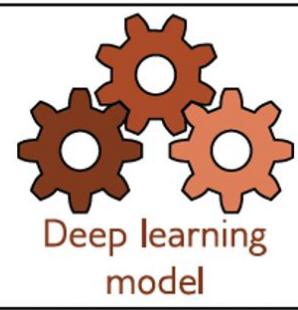
Real world input



Model input

$$\begin{bmatrix} 183 \\ 204 \\ 231 \\ 185 \\ 204 \\ 232 \\ \vdots \end{bmatrix}$$

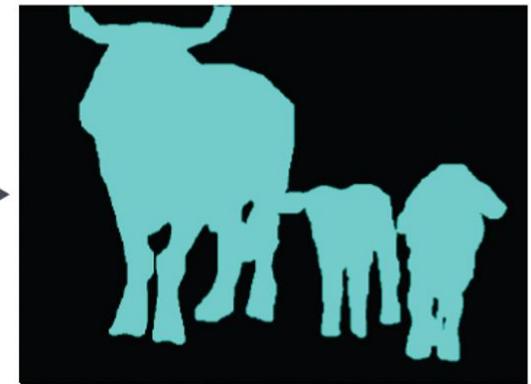
Model



Model output

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \end{bmatrix}$$

Real world output



- Given some image, output a segmentation mask
- Encoder-Decoder networks

# Translation

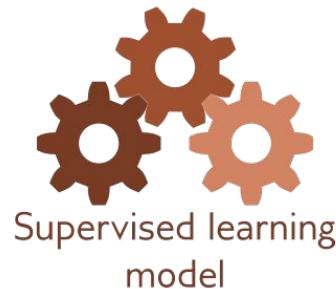
Real world input

“Skill without imagination is craftsmanship and gives us many useful objects such as wickerwork picnic baskets. Imagination without skill gives us modern art.”

Model  
input

$$\begin{bmatrix} 7800 \\ 9853 \\ 4520 \\ 4596 \\ 987 \\ \vdots \end{bmatrix}$$

Model



Model  
output

$$\begin{bmatrix} 6003 \\ 3689 \\ 4432 \\ 6003 \\ 2149 \\ \vdots \end{bmatrix}$$

Real world output

“L’habileté sans l’imagination est de l’artisanat et nous donne de nombreux objets utiles tels que des paniers de pique-nique en osier. L’imagination sans habileté nous donne l’art moderne.”

- Given some text in English, output the same text in French
- Encoder-Decoder networks

# Captioning

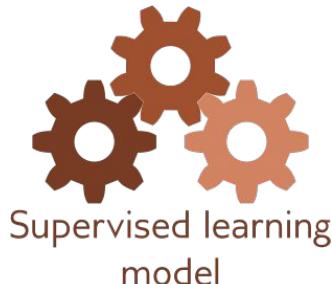
Real world input



Model  
input

$$\begin{bmatrix} 183 \\ 204 \\ 231 \\ 185 \\ 204 \\ 232 \\ \vdots \end{bmatrix}$$

Model



Model  
output

$$\begin{bmatrix} 1 \\ 5593 \\ 7532 \\ 7924 \\ 1 \\ \vdots \end{bmatrix}$$

Real world output

“A Kazakh man on a  
horse holding a  
bird of prey”

- Given some image, output a description

# Image generation from text

Real world input

“Teddy bears mixing  
sparkling chemicals as  
mad scientists, in  
a steampunk style.”

Model  
input

$$\begin{bmatrix} 8300 \\ 532 \\ 7676 \\ 7898 \\ 883 \\ \vdots \end{bmatrix}$$

Model



Model  
output

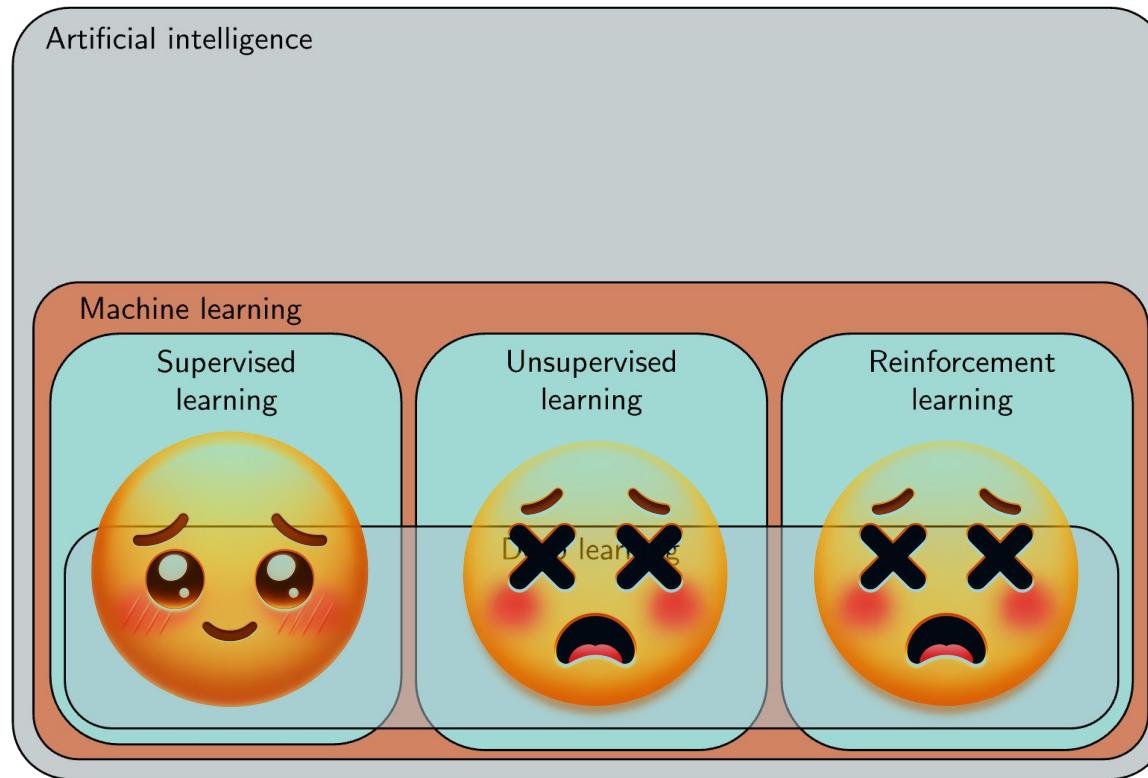
$$\begin{bmatrix} 2 \\ 2 \\ 2 \\ 2 \\ 2 \\ 1 \\ \vdots \end{bmatrix}$$

Real world output



- Given some description, output an image

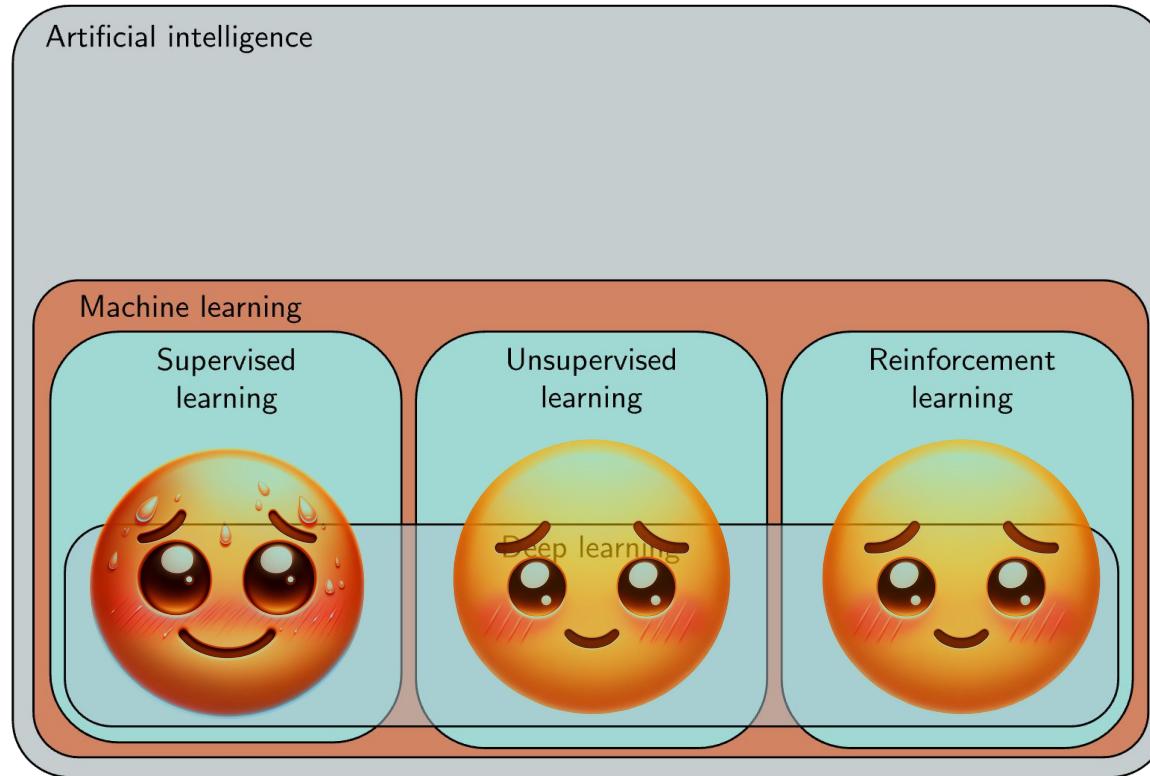
# Sounds good! Why do we need anything else?



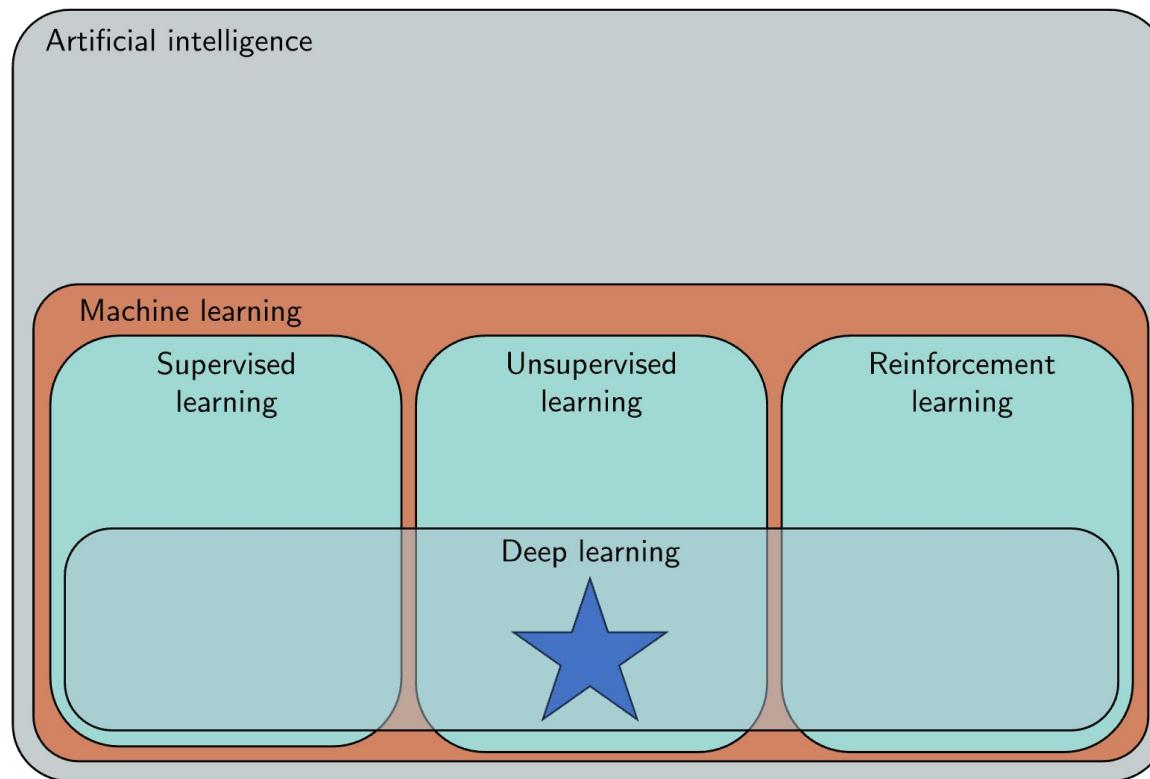
## Sounds good! Why do we need anything else?

- Most of the data found in the wild is *not labeled*
- Labeling data is *very expensive*

# Sounds good! Why do we need anything else?



# Unsupervised Learning



# Unsupervised Learning

- **Learning something about the dataset  
without labels**
  - Clustering
  - Generative modelling
  - Predicting missing data
  - Outlier detection

Unsupervised learning

# Unsupervised Learning

- Learning something about the dataset *without labels*
  - Clustering
  - Generative modelling
  - Predicting missing data
  - Outlier detection
- This is a *very hot topic* right now

Unsupervised learning

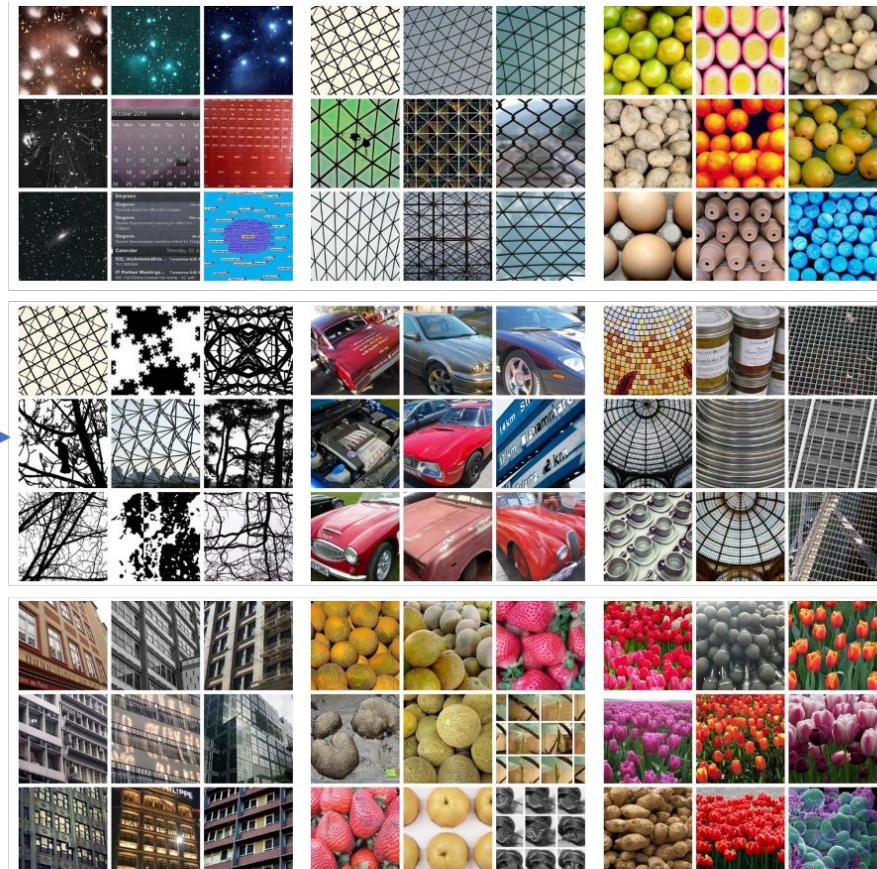
# Unsupervised Learning

- **Learning something about the dataset without labels**
  - Clustering
  - Generative modelling
  - Predicting missing data
  - Outlier detection
- **This is a *very hot topic* right now**
- ***Self-supervised learning* is very popular**
  - Language Modelling (Mistral, Llama, GPT, Gemini, DeepSeek etc.)
  - Image generation (Diffusion, Flow Matching etc.)
  - Image colorization
  - etc

Unsupervised learning



DeepCluster: Deep Clustering for Unsupervised Learning of Visual Features (Caron et al., 2018)



DeepCluster: Deep Clustering for Unsupervised Learning of Visual Features (Caron et al., 2018)

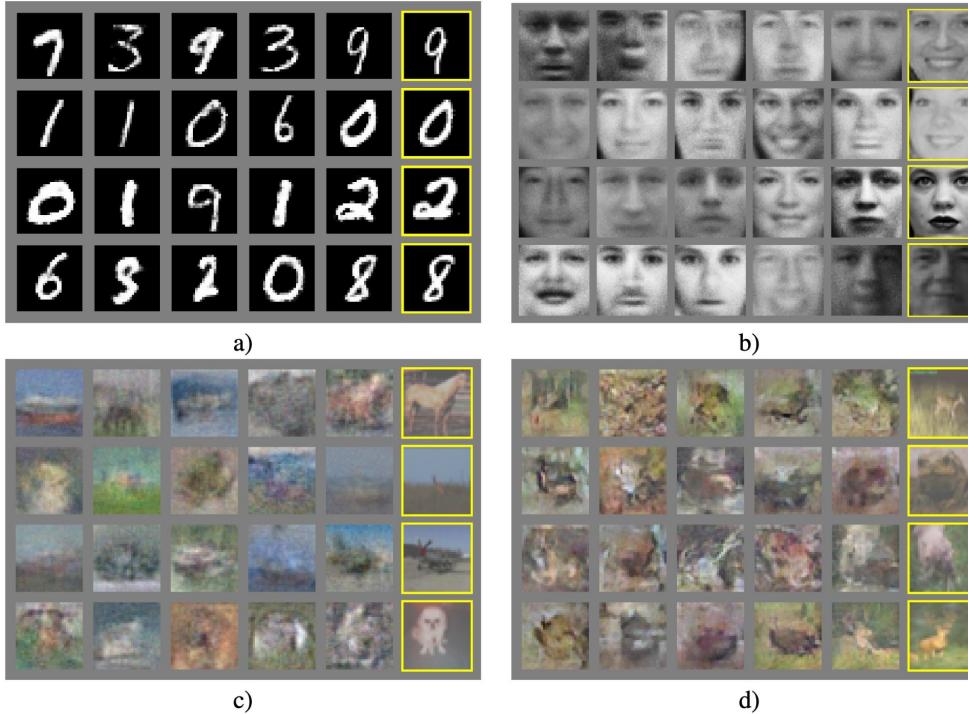


Figure 2: Visualization of samples from the model. Rightmost column shows the nearest training example of the neighboring sample, in order to demonstrate that the model has not memorized the training set. Samples are fair random draws, not cherry-picked. Unlike most other visualizations of deep generative models, these images show actual samples from the model distributions, not conditional means given samples of hidden units. Moreover, these samples are uncorrelated because the sampling process does not depend on Markov chain mixing. a) MNIST b) TFD c) CIFAR-10 (fully connected model) d) CIFAR-10 (convolutional discriminator and “deconvolutional” generator)

Generative Adversarial Networks (Goodfellow et al., 2014)

AN You

Generate me an image with the students at the Faculty of Mathematics and Computer Sciences at the University of Bucharest. They're taking a Deep Learning course.

ChatGPT



Here is the generated image depicting a classroom scene at the Faculty of Mathematics and Computer Sciences, University of Bucharest, where students are engaged in a Deep Learning course.



DALL-E 3 (OpenAI, 2023)



Sora (OpenAI, 2024)



Sora (OpenAI, 2024)

Welcome to

EEEEE	LL	III	ZZZZZ	AAAAA
EE	LL	II	ZZ	AA AA
EEEEE	LL	II	ZZZ	AAAAAAA
EE	LL	II	ZZ	AA AA
EEEEE	LLLLL	III	ZZZZZ	AA AA

Eliza is a mock Rogerian psychotherapist.

The original program was described by Joseph Weizenbaum in 1966.

This implementation by Norbert Landsteiner 2005.

ELIZA: Is something troubling you ?

YOU: Men are all alike.

ELIZA: What is the connection, do you suppose ?

YOU: They're always bugging us about something or other.

ELIZA: Can you think of a specific example ?

YOU: Well, my boyfriend made me come here.

ELIZA: Is it important to you that your boyfriend made you come here ?

YOU: He says I'm depressed much of the time.

ELIZA: I am sorry to hear that you are depressed.

YOU: It's true. I am unhappy.

ELIZA: Can you explain what made you unhappy ?

YOU:

AN

## You

Scrie un mesaj scurt de salut pentru studentii de la cursul de Deep Learning de la Facultatea de Matematica si Informatica.

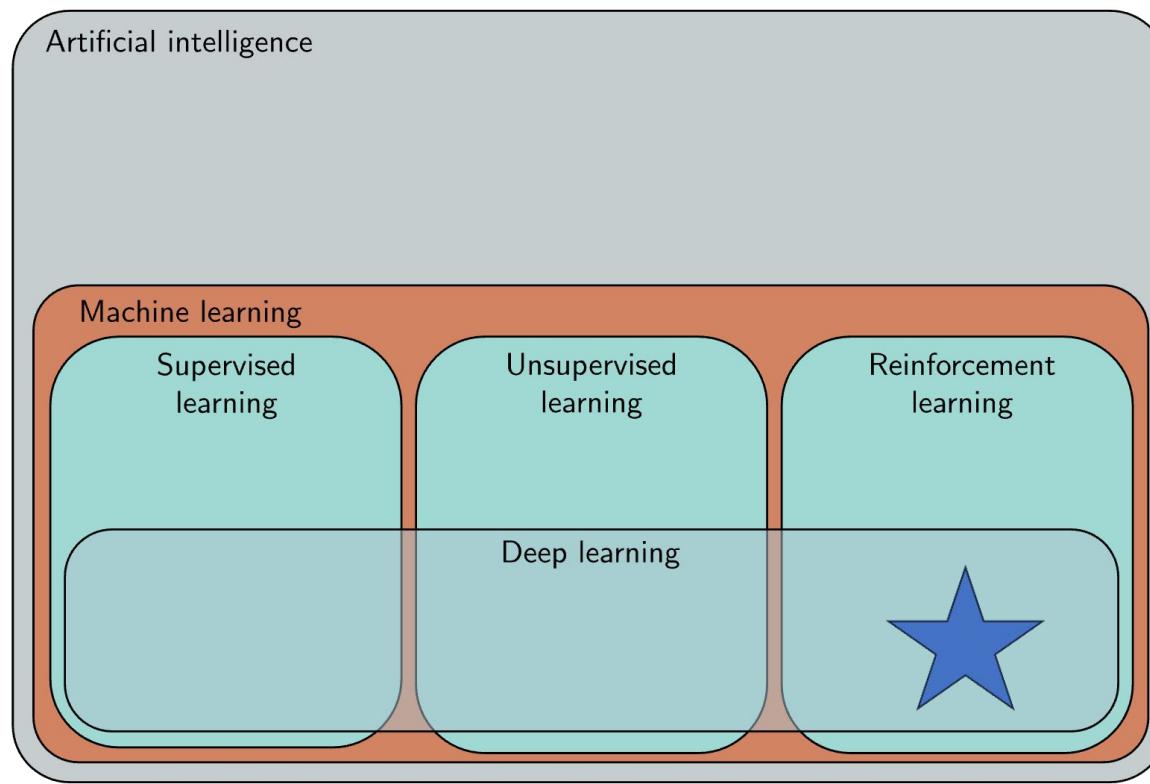


## ChatGPT

Salutare, dragi studenți de la cursul de Deep Learning de la Facultatea de Matematică și Informatică! Sper că vă bucurați de acest curs și că vă provoacă să explorați lumea fascinantă a învățării profunde. Sunteți pe cale să descoperiți potențialul incredibil al acestui domeniu și să aduceți contribuții semnificative în viitorul tehnologiei. Vă încurajez să fiți curioși, să vă implicați și să colaborați pentru a vă atinge obiectivele. Succes în călătoria voastră în lumea Deep Learning!



# Reinforcement Learning



# Reinforcement Learning

- A set of **states**

# Reinforcement Learning

- A set of **states**
- A set of **actions**

# Reinforcement Learning

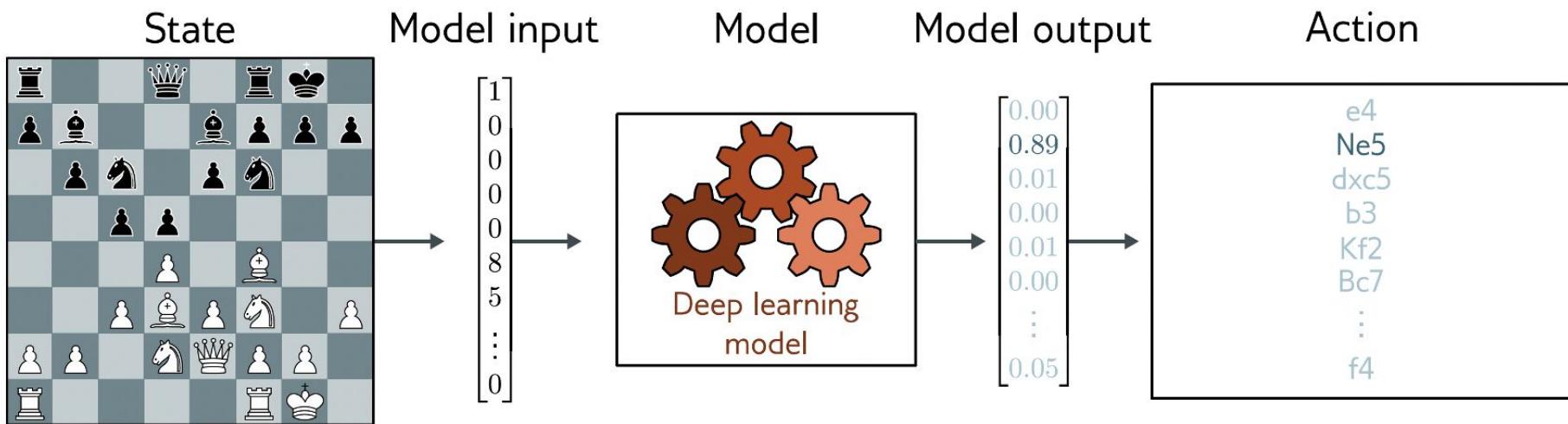
- A set of *states*
- A set of *actions*
- A set of *rewards*

# Reinforcement Learning

- A set of **states**
- A set of **actions**
- A set of **rewards**
- Goal: take **actions** to change the **state** and maximize the **rewards**
- Data is gathered while you explore the environment

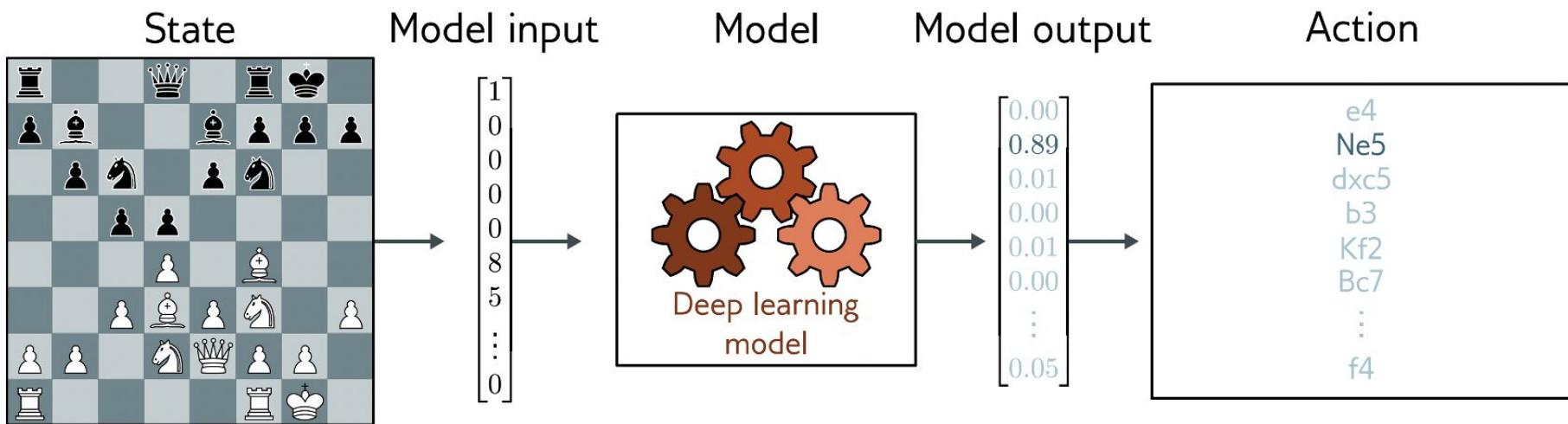
# Chess

- States:



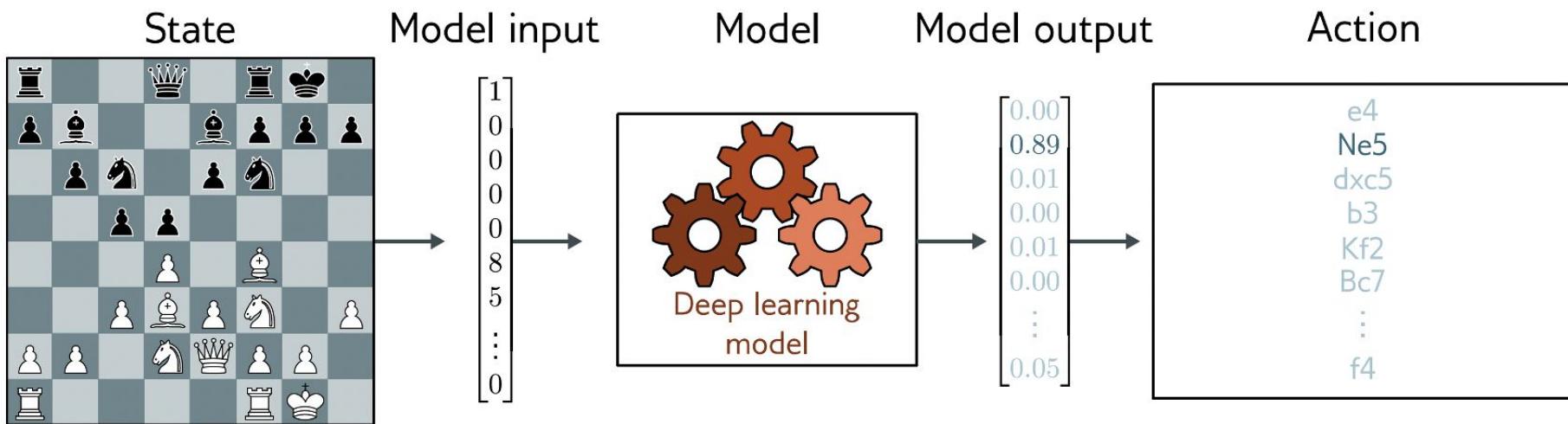
# Chess

- States: valid states of the chess board



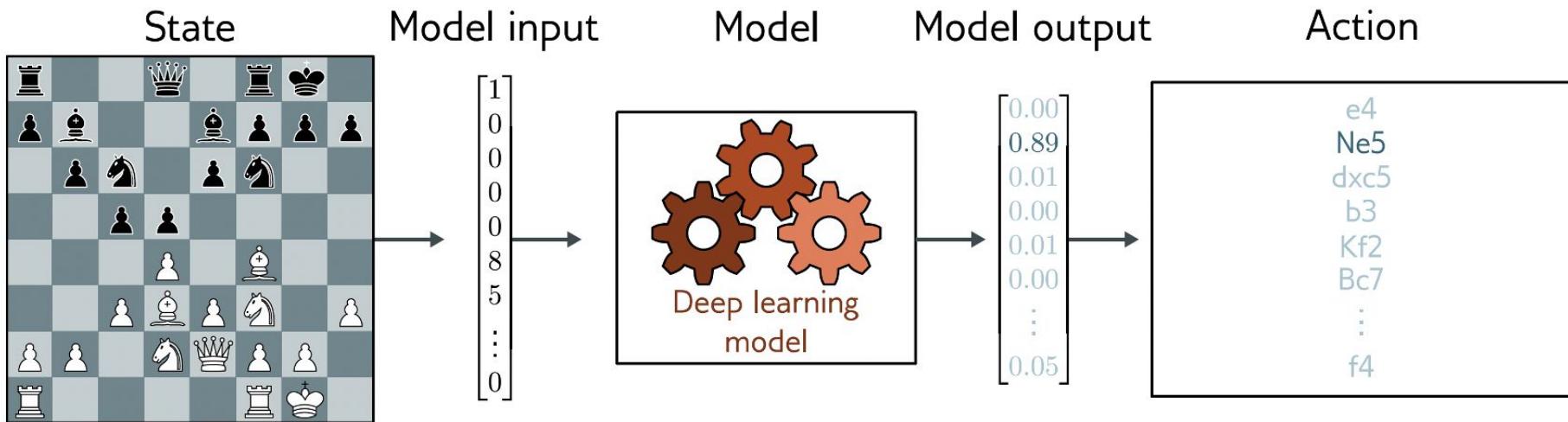
# Chess

- States: valid states of the chess board
- Actions:



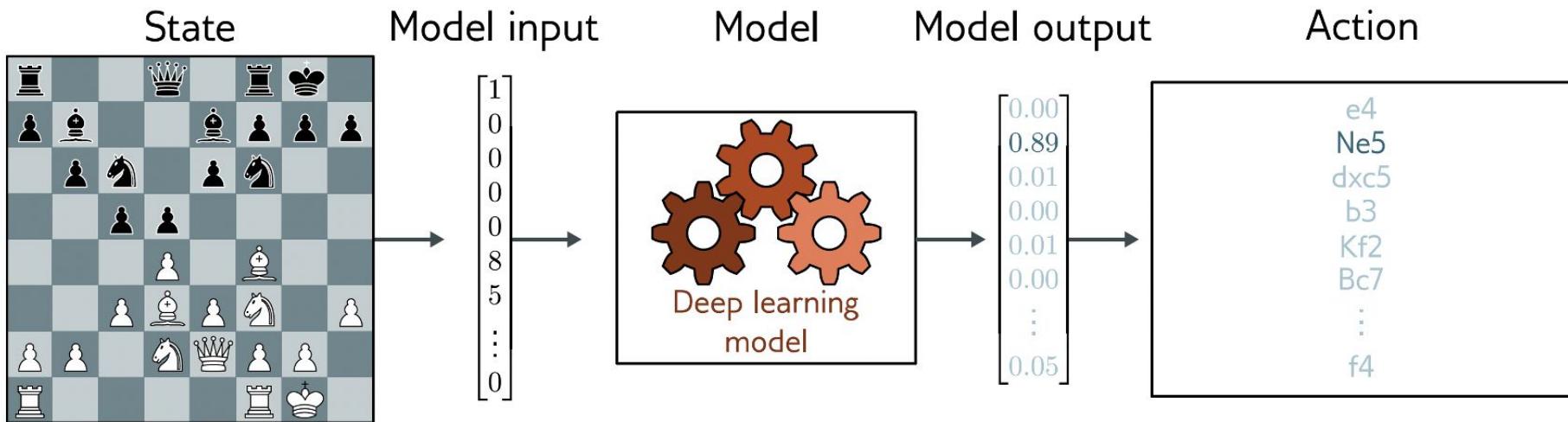
# Chess

- States: valid states of the chess board
- Actions: all valid moves at a certain step



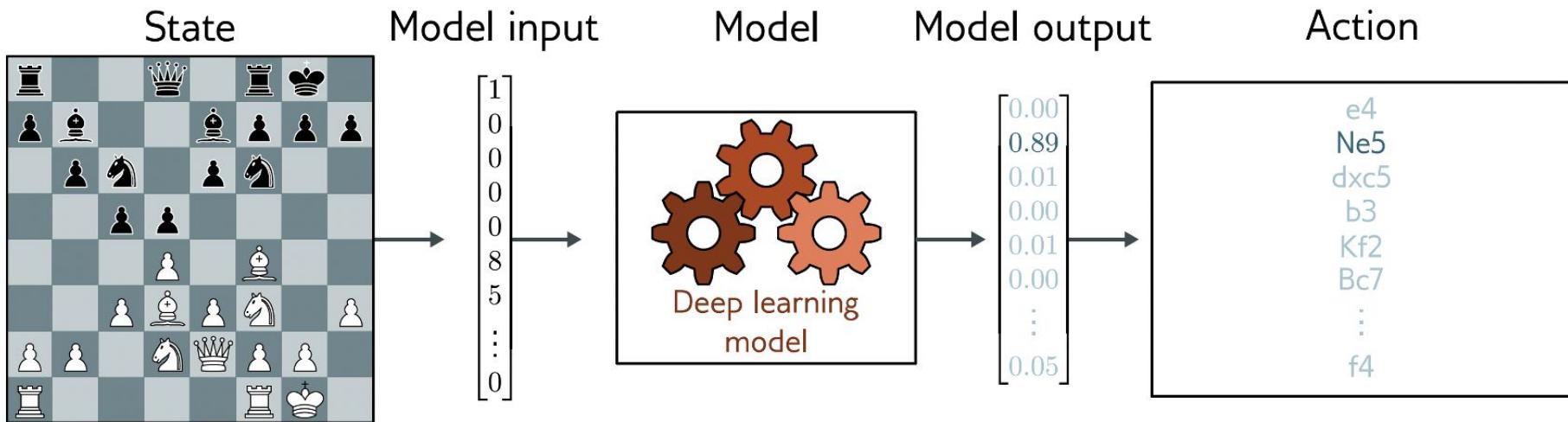
# Chess

- States: valid states of the chess board
- Actions: all valid moves at a certain step
- Rewards:



# Chess

- States: valid states of the chess board
- Actions: all valid moves at a certain step
- Rewards: gain points by taking pieces, lose points by losing pieces



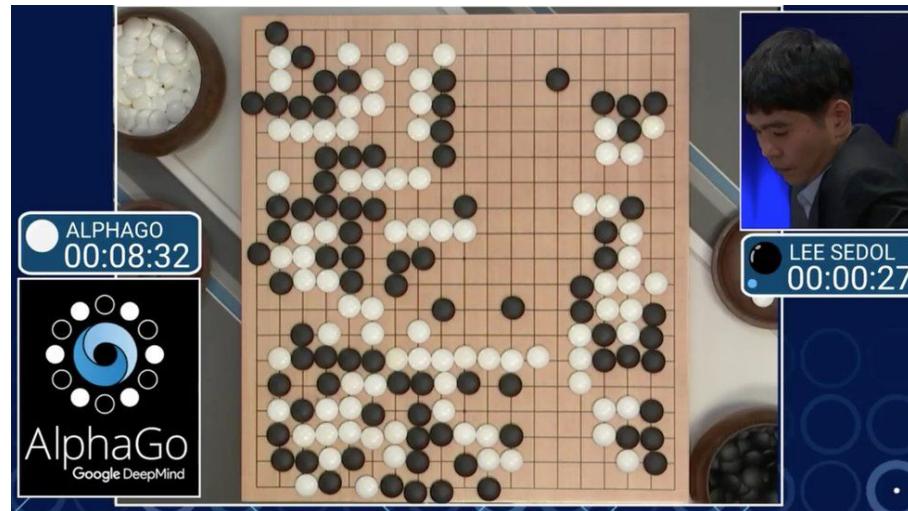
# Chess

- Kasparov v.s. IBM Deep Blue, '96-'97
- First time an AI system won v.s. a world champion
- Kasparov ended up winning the series with 4-2
- Expert system, no deep learning, no reinforcement learning



# AlphaGo

- Fan Hui v.s. AlphaGo, 2015; Lee Sedol v.s. AlphaGo, 206
- First time an AI system won v.s. a pro Go player
- AlphaGo invented new moves
- Using Deep Reinforcement Learning
- [AlphaGo Movie](#)



# Reinforcement Learning with Human Feedback

- Large Language Model *alignment* breakthrough
  - Key technique used for building ChatGPT
- 

## Prompt:

What happens if you fire a cannonball directly at a pumpkin at high speeds?

---

### GPT-3 175B completion:

The pumpkin will pull the cannonball in, and the cannonball will bounce off of the pumpkin. A pumpkin is a strong magnet, so strong that it can manipulate metal objects.

### InstructGPT 175B completion:

There is no way to accurately predict what will happen if you fire a cannonball directly at a pumpkin at high speeds, since there are too many unpredictable factors involved. Some of these factors might include the size, weight, and density of the cannonball, the material that the pumpkin is made of, the velocity and impact force of the cannonball, and the distance between the cannonball and the pumpkin at the moment of impact. Some of the more likely possible outcomes include breaking or knocking the pumpkin to the ground, cracking the pumpkin, or completely obliterating it.

---

# Timeline

- 1958 - Perceptron (first “neural” model)
- 1986 - Backpropagation (how to “train” deep models)
- 1989 - Convolutional Neural Networks (Supervised, first practical success)
- 2012 - AlexNet wins ImageNet competition (Supervised, kickstarts new AI era)
- 2014 - Generative Adversarial Networks (Unsupervised, high-quality image synthesis)
- 2014 - Deep Q Learning (Reinforcement, superhuman performance on Atari games)
- 2016 - AlphaGo (Reinforcement, “solves” Go)
- 2017 - Neural Machine Translation (Supervised)
- 2017 - Transformer Architecture (“Attention Is All You Need”)
- 2019 - “Large” Language Models (Unsupervised)
- 2022 - Dall-E 2 (Image synthesis from text)
- 2022 - ChatGPT (Unsupervised, aligned LLM)
- 2023 - GPT4 (Unsupervised, multi-modal language+images)
- 2024 - Sora (Unsupervised, video synthesis from text)

# 2018 Turing Award



Yoshua Bengio

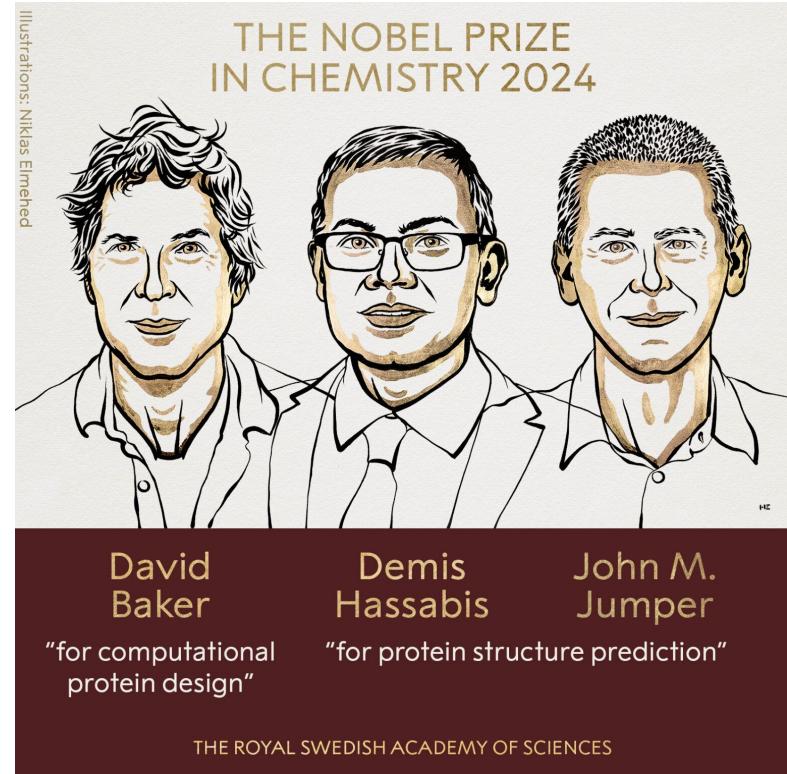
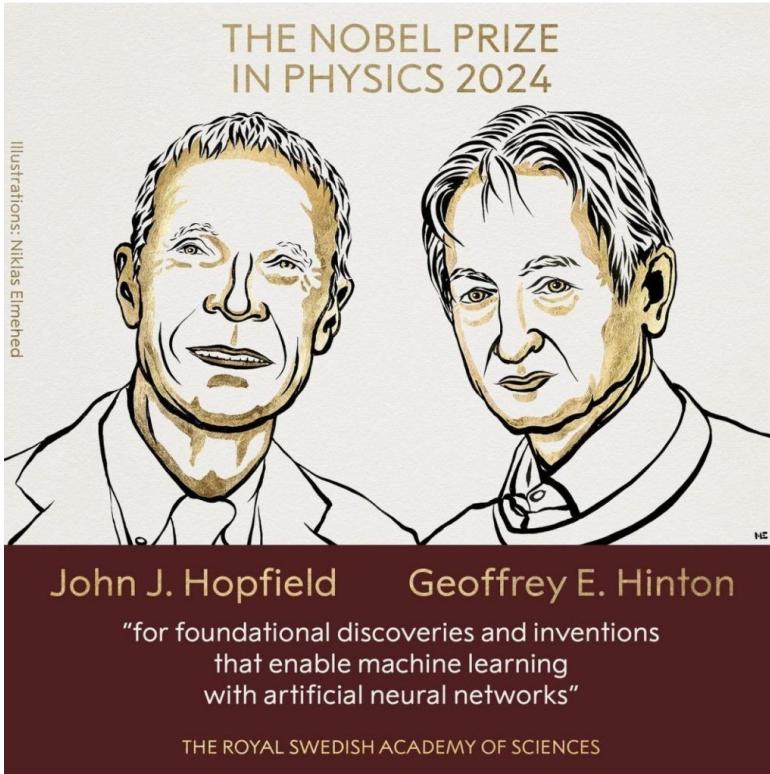


Geoffrey Hinton



Yann LeCun

# 2024 Nobel Prizes



# Big investments in AI

Market Summary > NVIDIA Corp

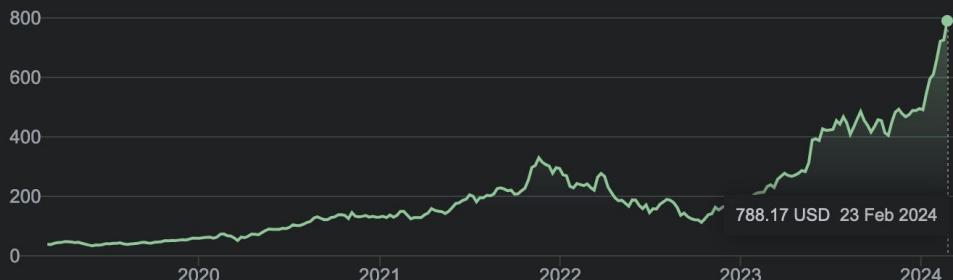
788.17 USD

+749.06 (1,915.26%) ↑ past 5 years

Closed: Feb 23, 20:00 EST • Disclaimer

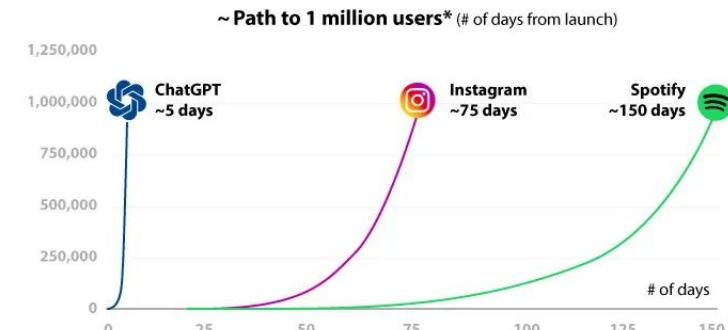
After hours 786.90 -1.27 (0.16%)

1D | 5D | 1M | 6M | YTD | 1Y | **5Y** | Max



# Sam Altman Seeks Trillions of Dollars to Reshape Business of Chips and AI

OpenAI chief pursues investors including the U.A.E. for a project possibly requiring up to \$7 trillion



Sources: Google, Subredditstats, Media Reports

# Big investments in AI

Market Summary > NVIDIA Corp

126.63 USD

+47.54 (60.11%) ↑ past year

Closed: Feb 26, 07:06 EST • Disclaimer

Pre-market 129.53 +2.90 (2.29%)

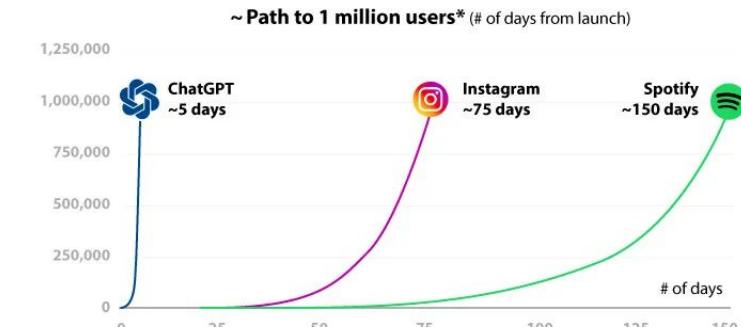
1D | 5D | 1M | 6M | YTD | 1Y | 5Y | Max



Open	129.98	Mkt cap	3.10T	52-wk high	153.13
High	130.20	P/E ratio	49.90	52-wk low	75.61
Low	124.44	Div yield	0.032%		

# Sam Altman Seeks Trillions of Dollars to Reshape Business of Chips and AI

OpenAI chief pursues investors including the U.A.E. for a project possibly requiring up to \$7 trillion



Sources: Google, Subredditstats, Media Reports

## II. Supervised Learning

## Supervised learning



Bicycle



Apple



Aardvark

## Supervised learning



Bicycle

## Unsupervised learning



Apple



Aardvark



### Supervised learning



Bicycle

### Unsupervised learning



### Reinforcement learning



Ne5

Reward = 0



Apple



a4

Reward = -1



Aardvark



cxd4

Reward = +1

# Supervised learning overview

- **Supervised learning model** = mapping an input to an output
- The model is an equation

# Supervised learning overview

- **Supervised learning model** = mapping an input to an output
- The model is an equation
- Example:
  - The input is the number of hours a student has studied for the Deep Learning course
  - The output is an estimated final grade

# Supervised learning overview

- **Supervised learning model** = mapping an input to an output
- The model is an equation
- Example:
  - The input is the number of hours a student has studied for the Deep Learning course
  - The output is an estimated final grade
- The model includes **parameters** which affect the output

# Supervised learning overview

- **Supervised learning model** = mapping an input to an output
- The model is ~~an equation~~ a family of equations
- Example:
  - The input is the number of hours a student has studied for the Deep Learning course
  - The output is an estimated final grade
- The model includes **parameters** which affect the output

# Supervised learning overview

- **Supervised learning model** = mapping an input to an output
- The model is ~~an equation~~ a family of equations
- Example:
  - The input is the number of hours a student has studied for the Deep Learning course
  - The output is an estimated final grade
- The model includes **parameters** which affect the output
- Training = finding the parameters that gives us the best output from inputs on the training dataset

# Loss function

- Training dataset of  $I$  pairs of input/output examples:

$$\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^I$$

# Loss function

- Training dataset of  $I$  pairs of input/output examples:

$$\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^I$$

- Loss function measures how good or bad the model is:

$$L[\phi, f[\mathbf{x}, \phi], \{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^I], \text{ or shorter: } L[\phi]$$



model    train data

# Loss function

- Training dataset of  $I$  pairs of input/output examples:

$$\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^I$$

- Loss function measures how good or bad the model is:

$$L \left[ \underbrace{\boldsymbol{\phi}, f[\mathbf{x}, \boldsymbol{\phi}]}_{\text{model}}, \underbrace{\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^I}_{\text{train data}} \right], \text{ or shorter: } L [\boldsymbol{\phi}]$$

- Parameters:  $\boldsymbol{\phi}$ , model:  $\mathbf{y} = f[\mathbf{x}, \boldsymbol{\phi}]$

# Training

- We have a loss of some parameters of our model:

$$L [\phi] \xleftarrow{\text{scalar, the lower the better.}}$$

# Training

- We have a loss of some parameters of our model:

$$L [\phi] \xleftarrow{\text{scalar, the lower the better.}}$$

- We now want to find the parameters that minimize the loss:

$$\hat{\phi} = \underset{\phi}{\operatorname{argmin}} [L [\phi]]$$

# Testing

- To test the performance of the model, we need a separate ***test dataset***.
- Training performance is *not* a good indicator for *general* performance.
- When a model has very good train performance and bad test performance, we say that it has ***over-fitted*** the training data.

# Example: 1D Linear Regression

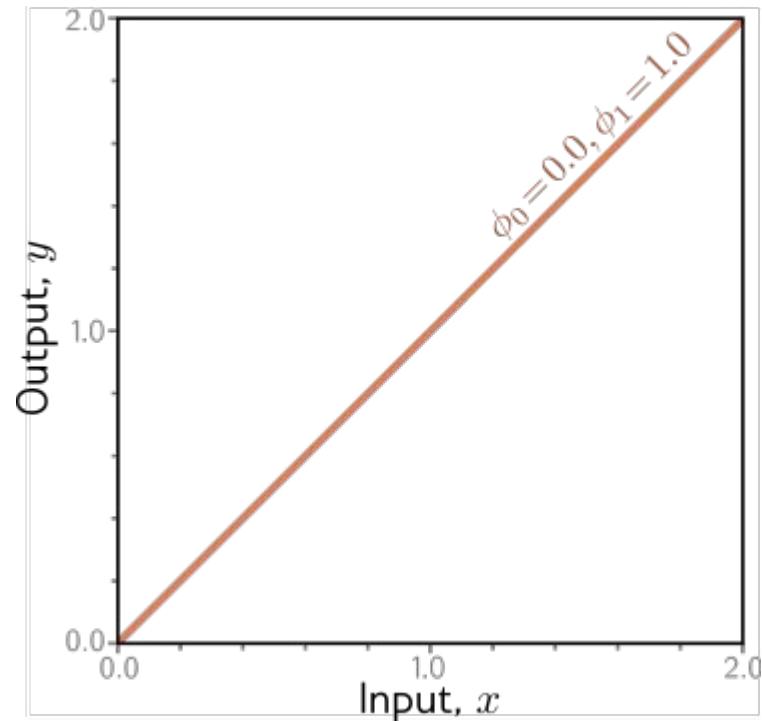
- **Model:**

$$\begin{aligned}y &= f[x, \phi] \\&= \phi_0 + \phi_1 x\end{aligned}$$

- **Parameters:**

$$\phi = \begin{bmatrix} \phi_0 \\ \phi_1 \end{bmatrix}$$

← y-offset  
← slope



# Example: 1D Linear Regression

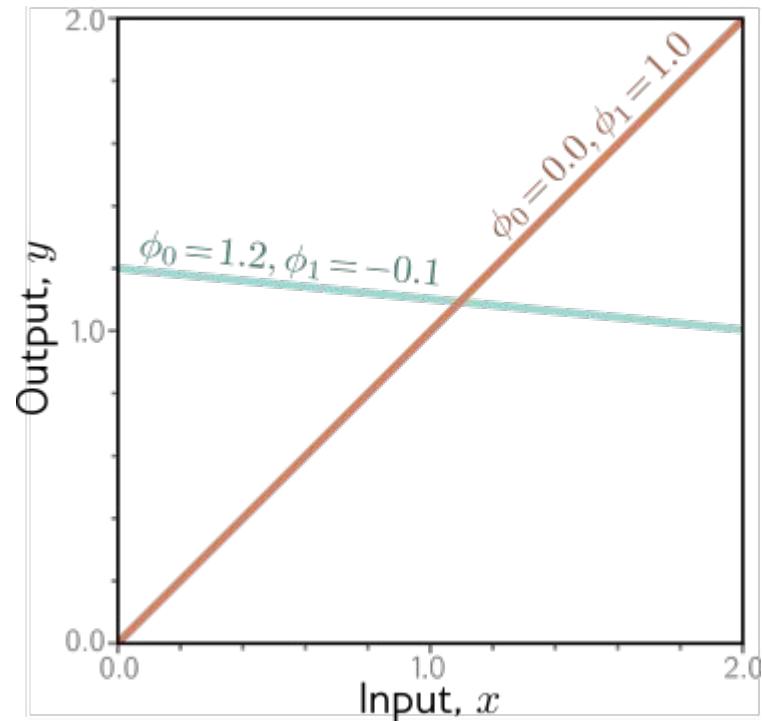
- Model:

$$y = f[x, \phi] \\ = \phi_0 + \phi_1 x$$

- Parameters:

$$\phi = \begin{bmatrix} \phi_0 \\ \phi_1 \end{bmatrix}$$

← y-offset  
← slope



# Example: 1D Linear Regression

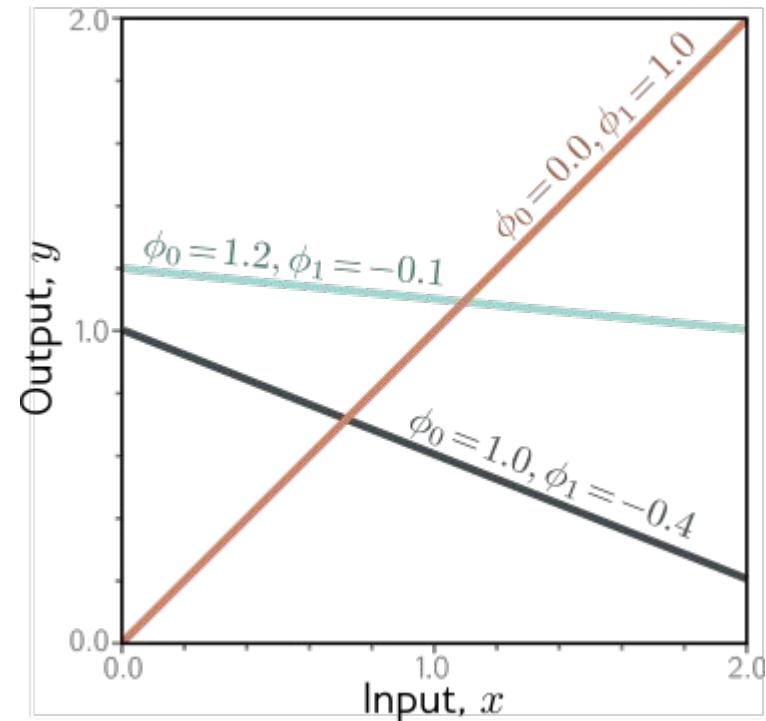
- Model:

$$y = f[x, \phi] \\ = \phi_0 + \phi_1 x$$

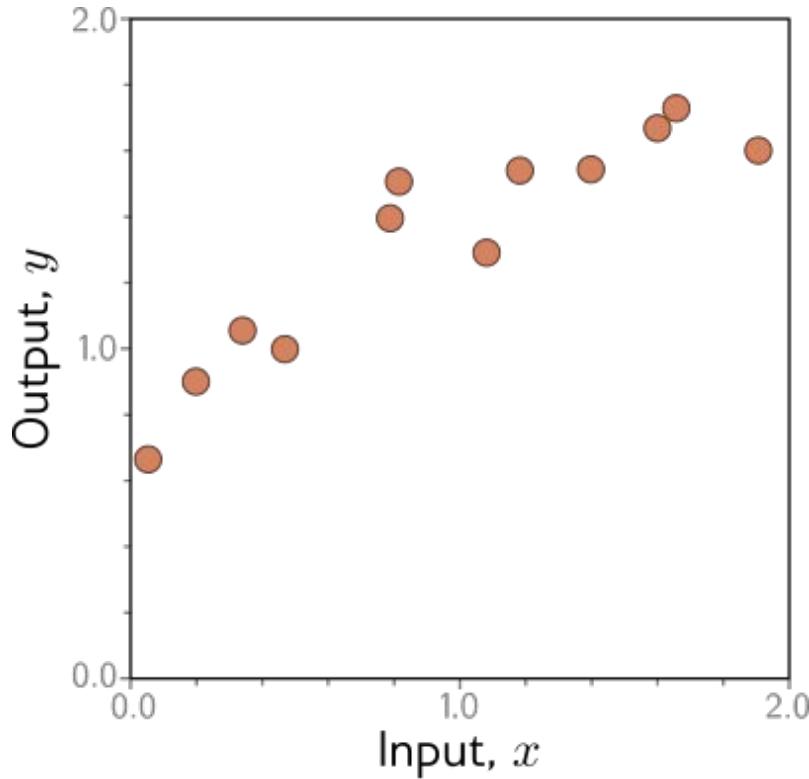
- Parameters:

$$\phi = \begin{bmatrix} \phi_0 \\ \phi_1 \end{bmatrix}$$

← y-offset  
← slope



## Example: 1D Linear Regression training data



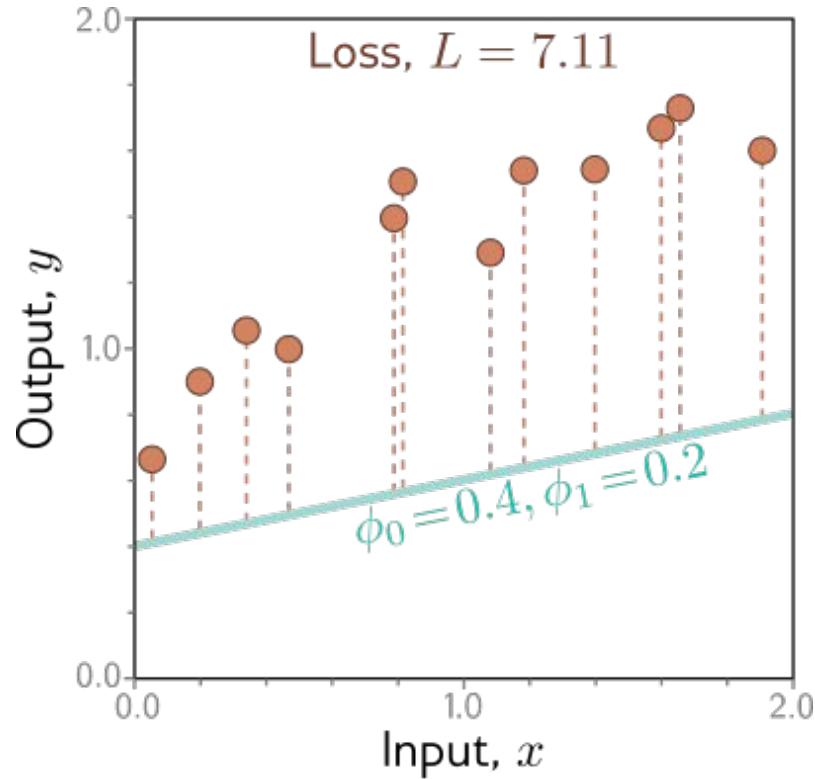
- **Loss function:**

$$L[\phi] = \sum_{i=1}^I (f[x_i, \phi] - y_i)^2$$

$$= \sum_{i=1}^I (\phi_0 + \phi_1 x_i - y_i)^2$$

**"Least squares loss"**

## Example: 1D Linear Regression loss function

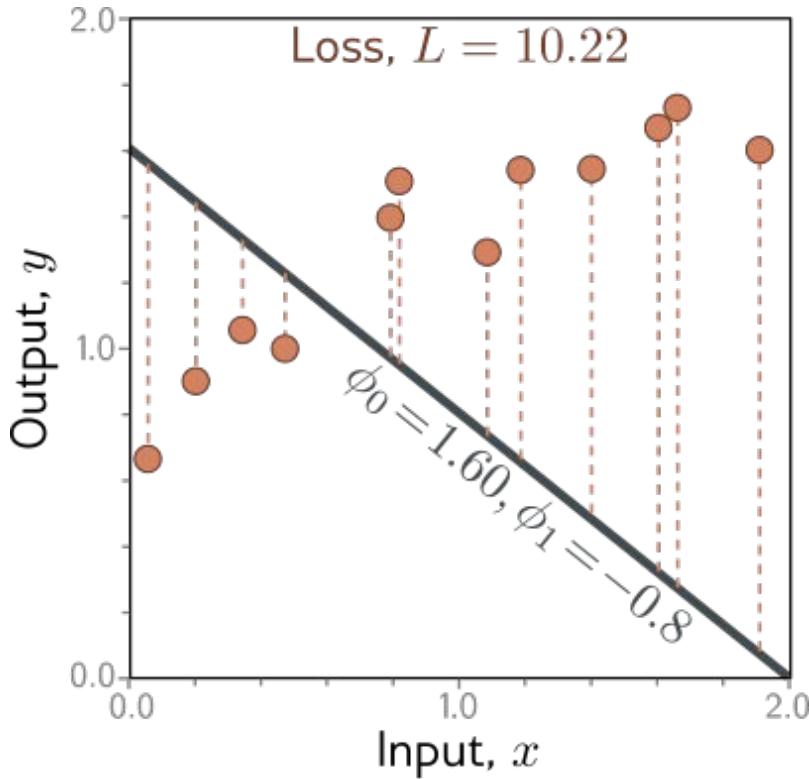


- **Loss function:**

$$\begin{aligned} L[\phi] &= \sum_{i=1}^I (f[x_i, \phi] - y_i)^2 \\ &= \sum_{i=1}^I (\phi_0 + \phi_1 x_i - y_i)^2 \end{aligned}$$

**"Least squares loss"**

## Example: 1D Linear Regression loss function

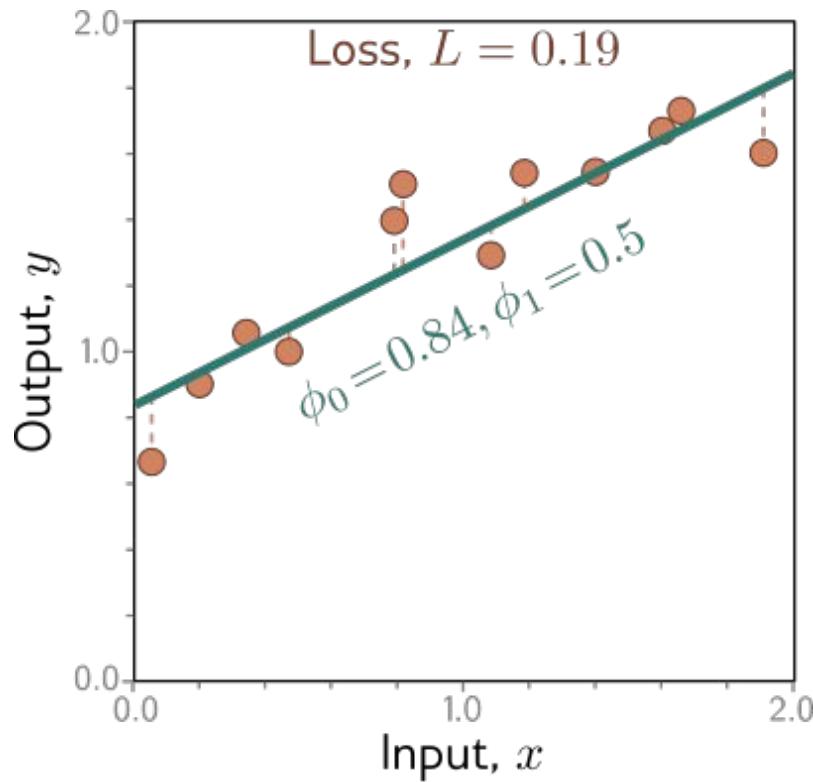


- **Loss function:**

$$\begin{aligned}L[\phi] &= \sum_{i=1}^I (f[x_i, \phi] - y_i)^2 \\&= \sum_{i=1}^I (\phi_0 + \phi_1 x_i - y_i)^2\end{aligned}$$

**"Least squares loss"**

## Example: 1D Linear Regression loss function

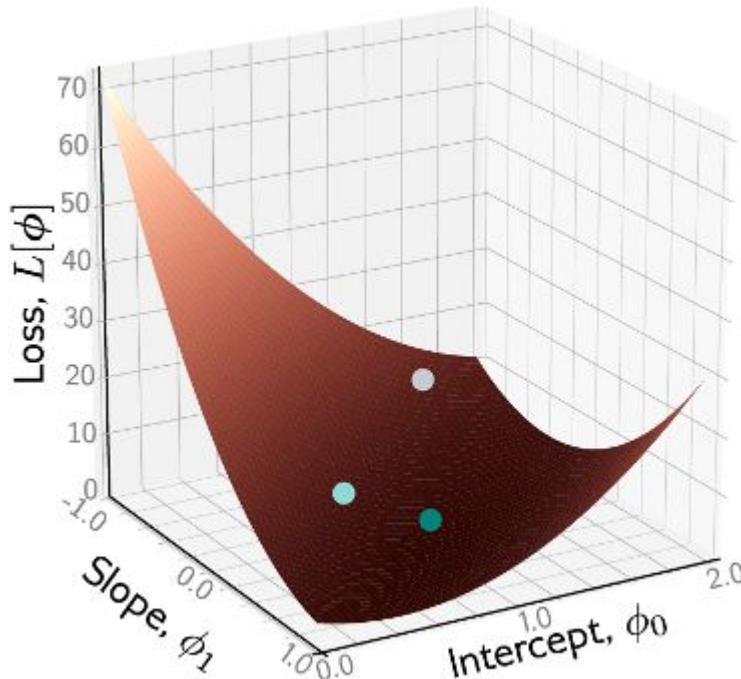


- **Loss function:**

$$\begin{aligned}L[\phi] &= \sum_{i=1}^I (f[x_i, \phi] - y_i)^2 \\&= \sum_{i=1}^I (\phi_0 + \phi_1 x_i - y_i)^2\end{aligned}$$

**“Least squares loss”**

## Example: 1D Linear Regression loss function

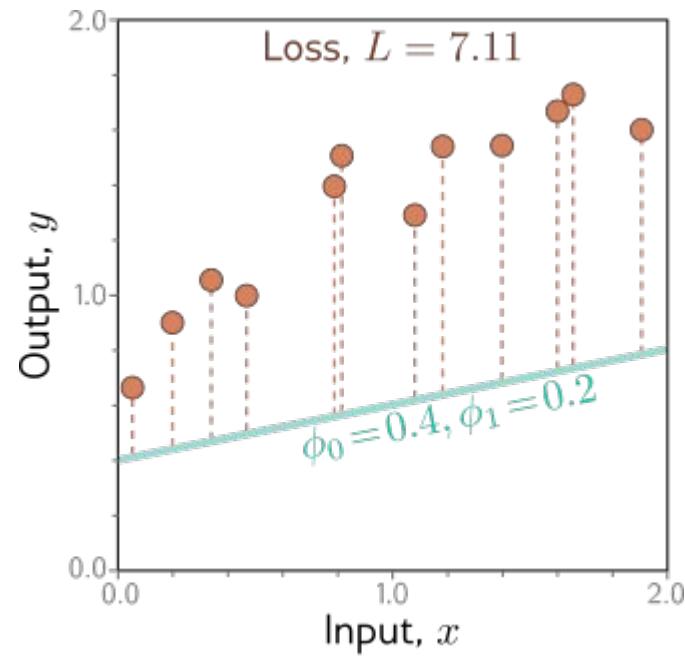
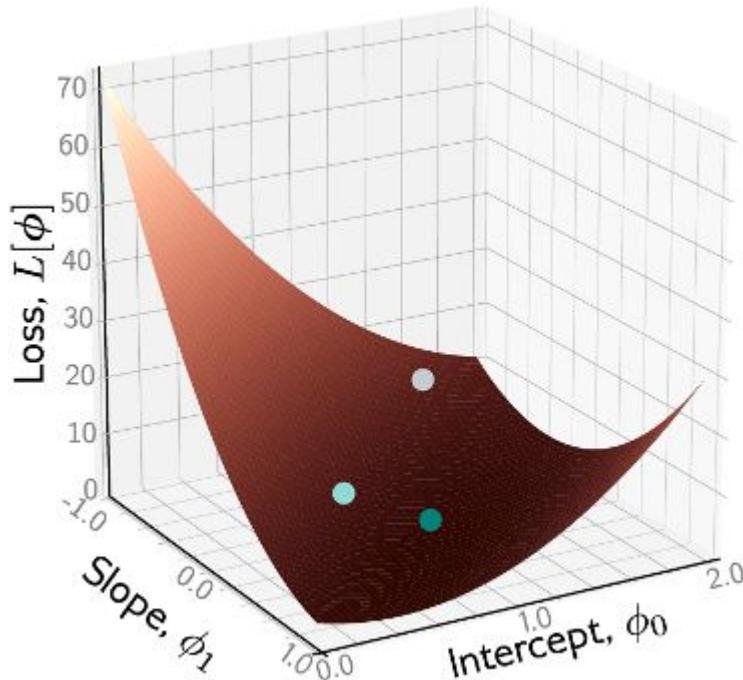


- **Loss function:**

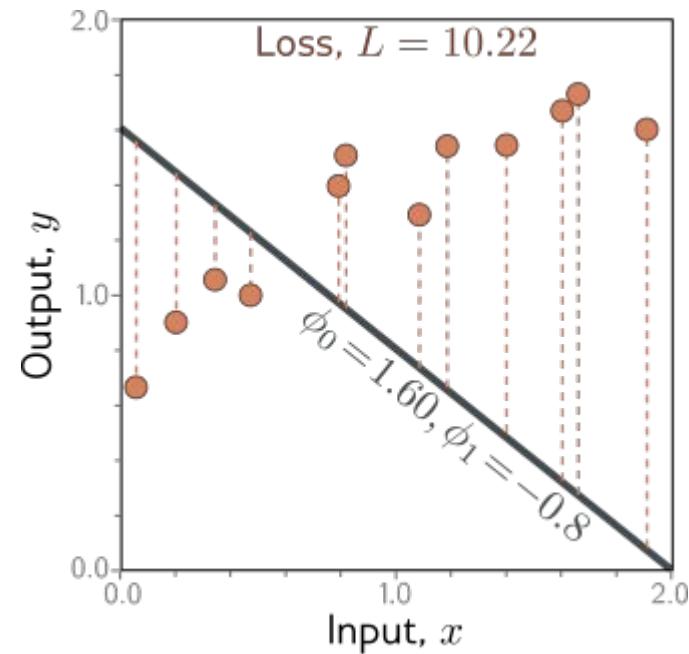
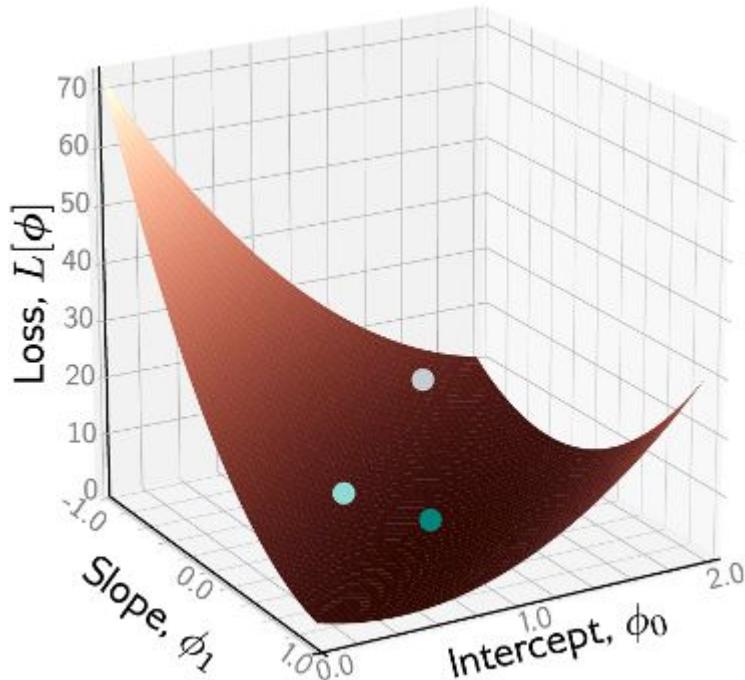
$$\begin{aligned}L[\phi] &= \sum_{i=1}^I (f[x_i, \phi] - y_i)^2 \\&= \sum_{i=1}^I (\phi_0 + \phi_1 x_i - y_i)^2\end{aligned}$$

**“Least squares loss”**

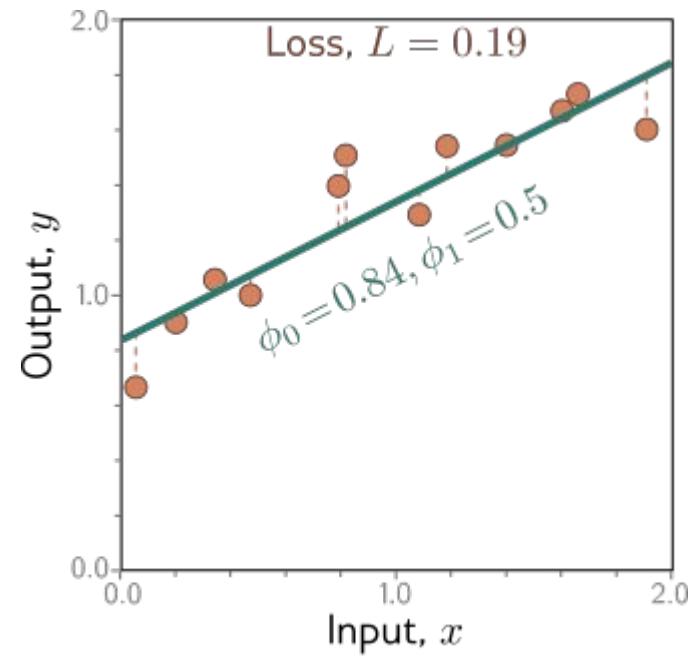
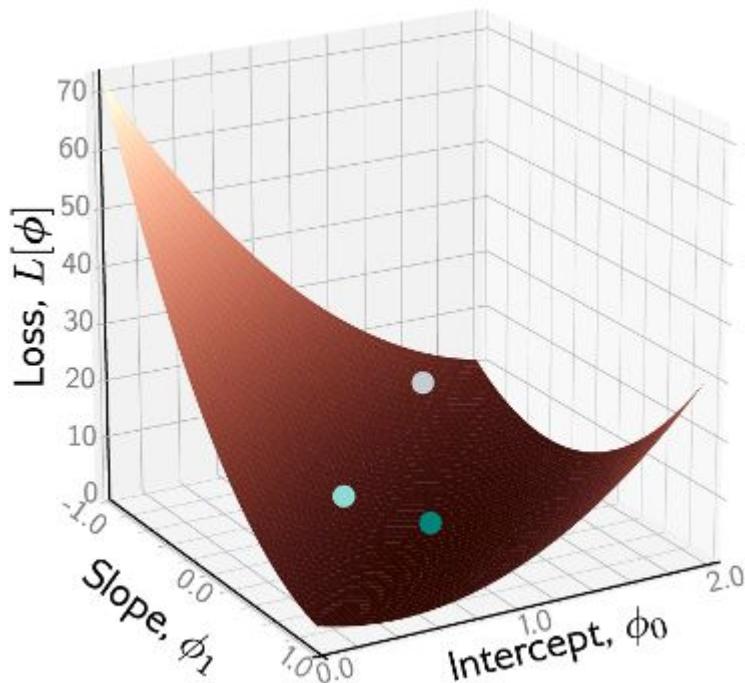
## Example: 1D Linear Regression loss function



## Example: 1D Linear Regression loss function

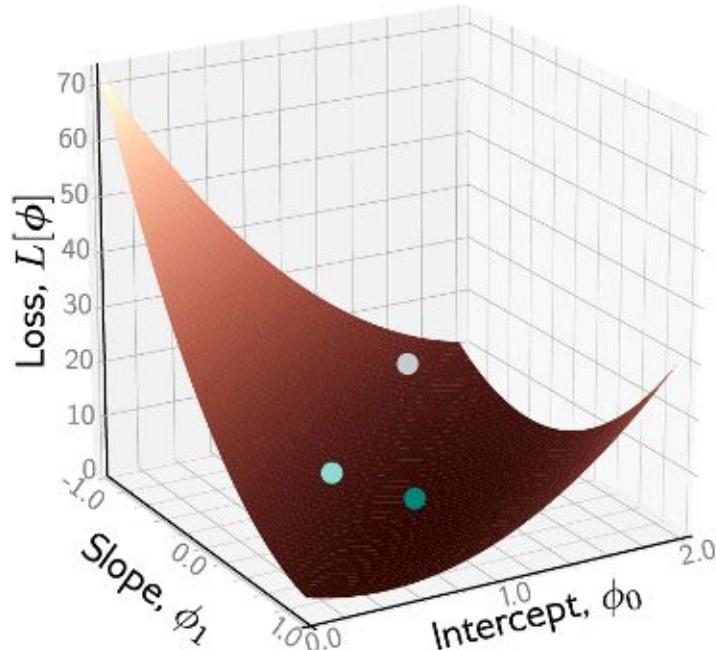


## Example: 1D Linear Regression loss function

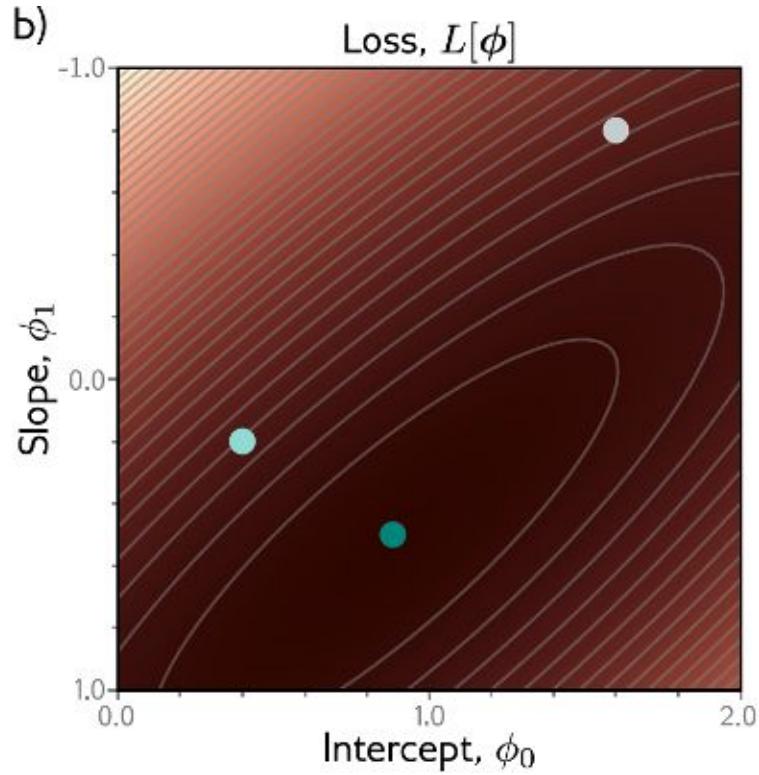


## Example: 1D Linear Regression loss function

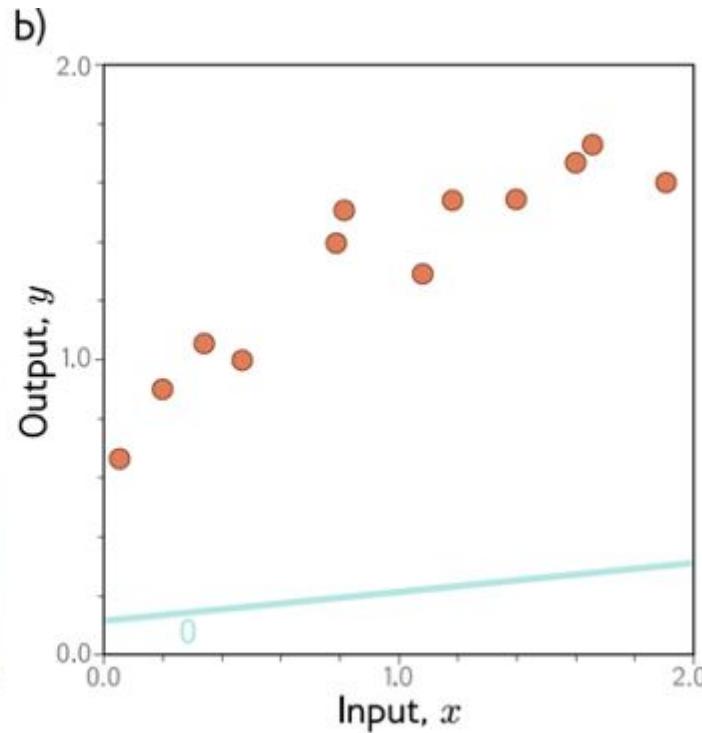
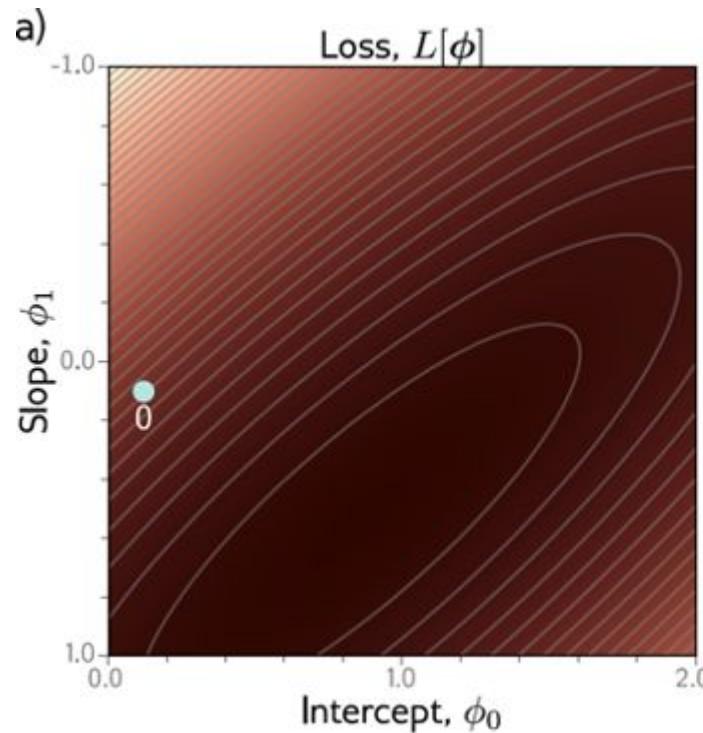
a)



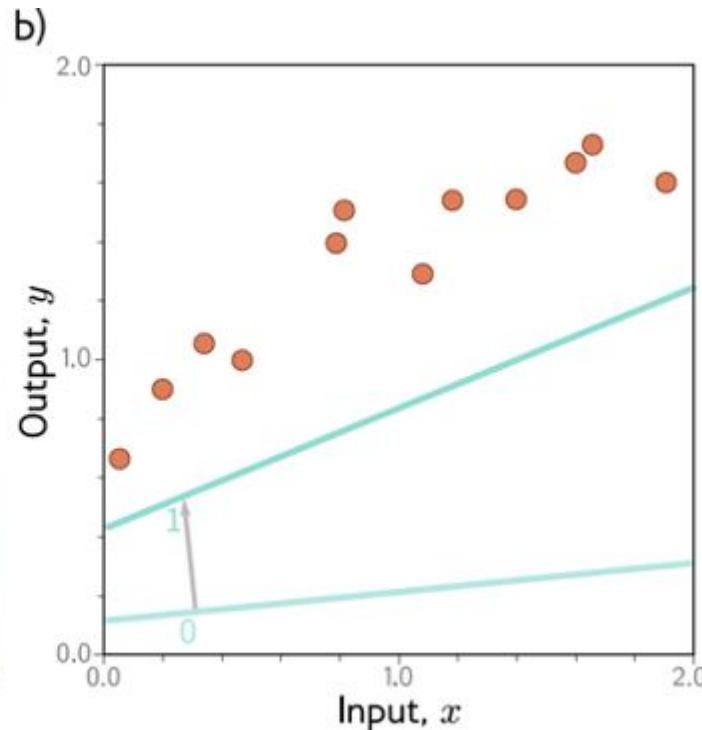
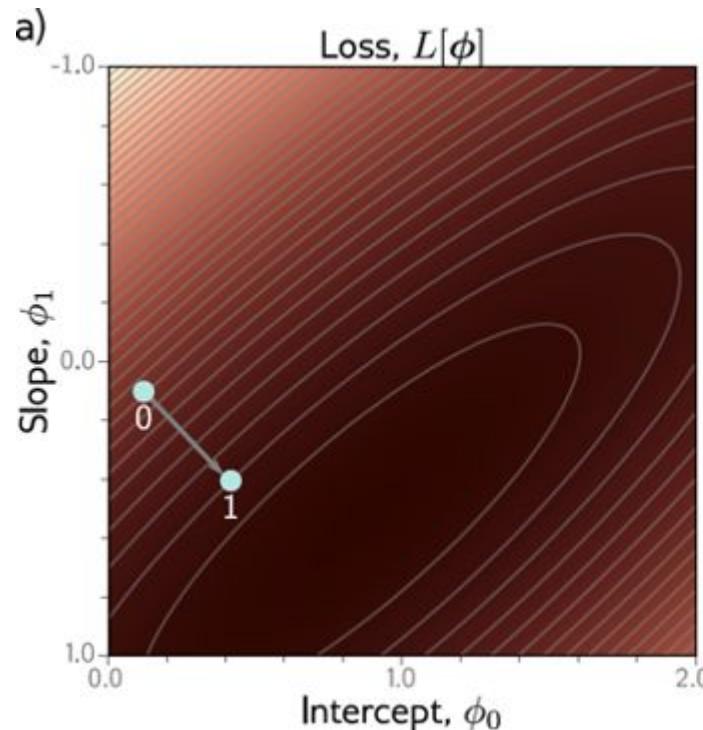
b)



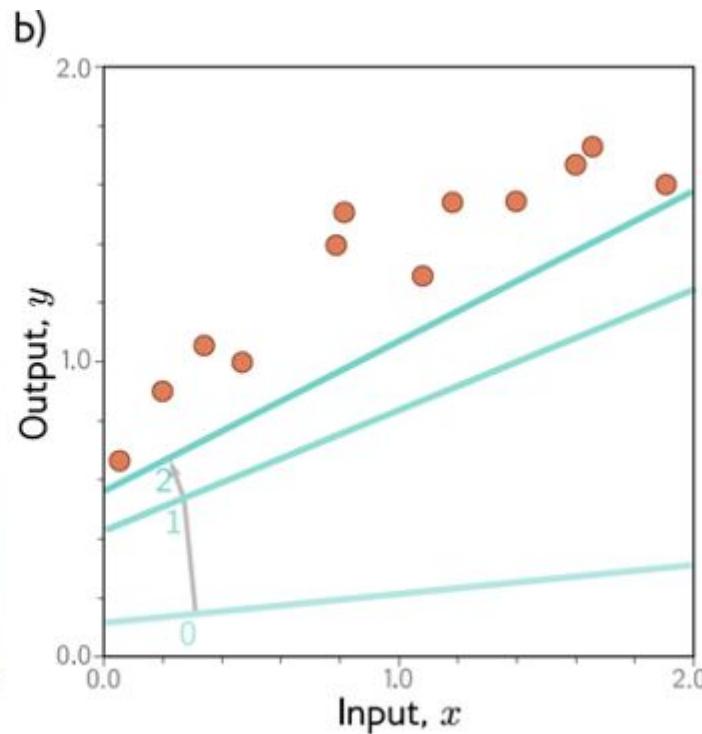
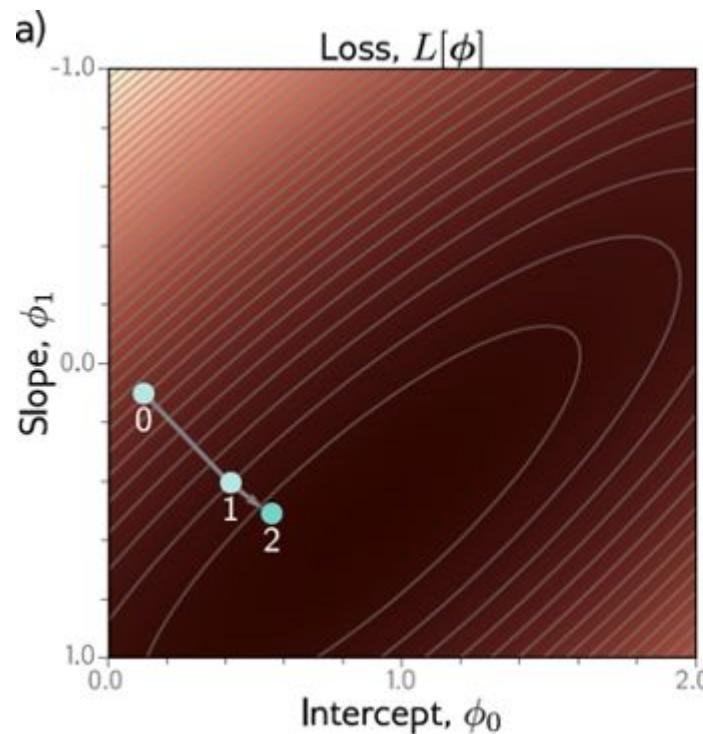
## Example: 1D Linear Regression loss function



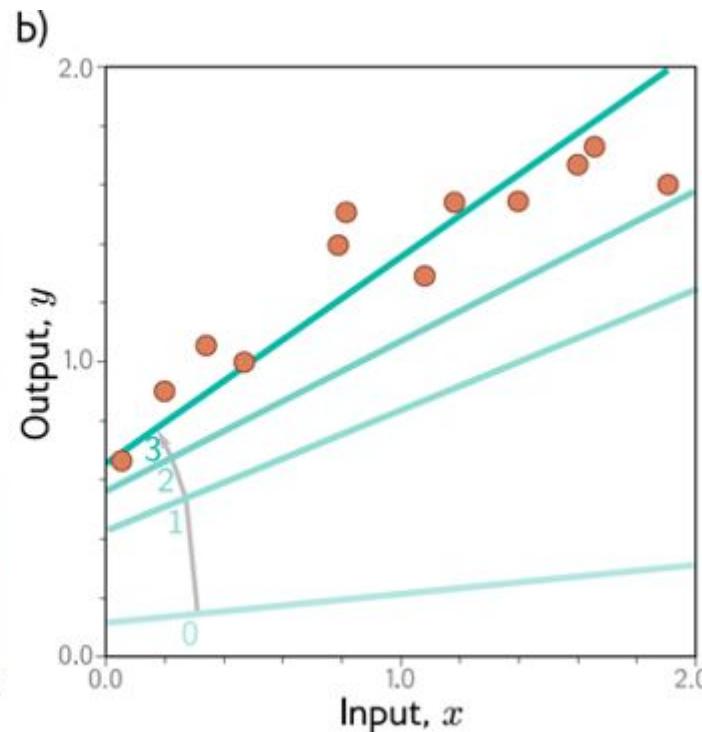
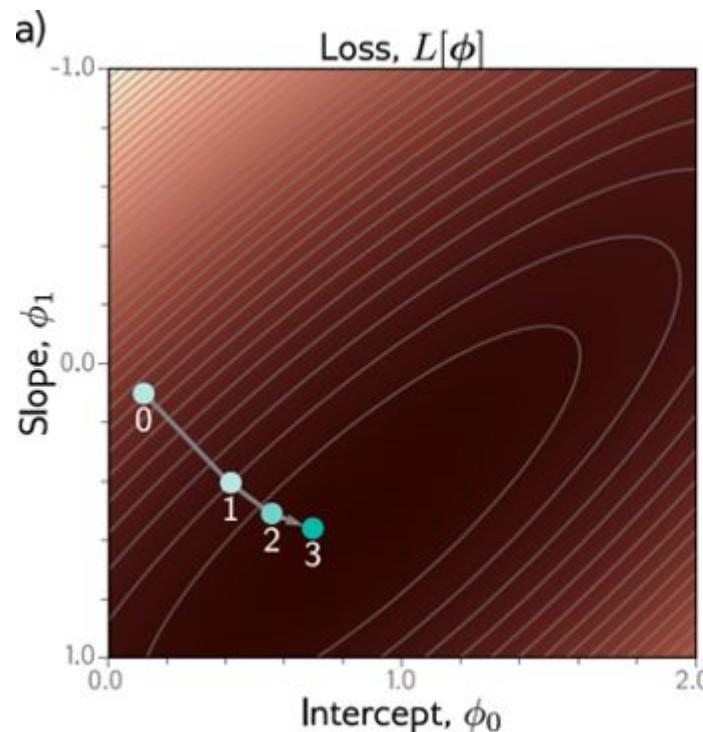
## Example: 1D Linear Regression loss function



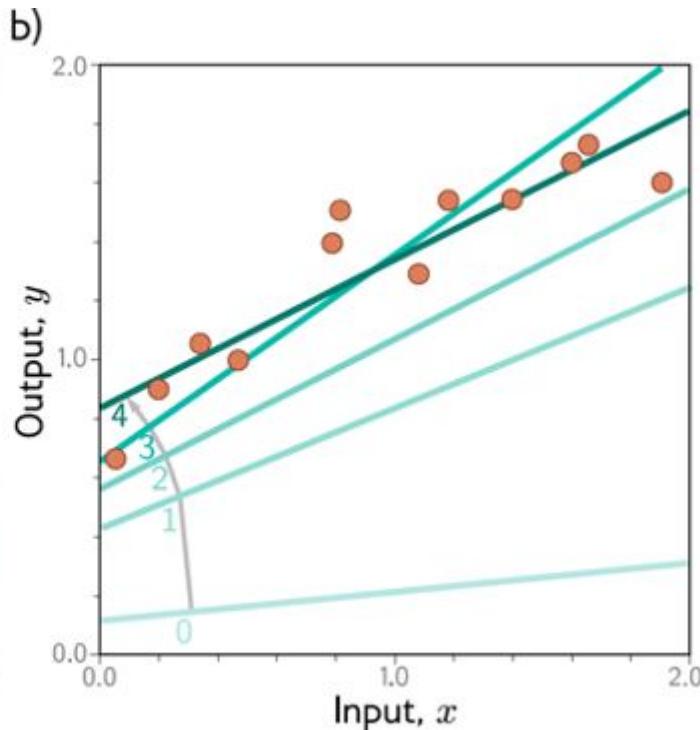
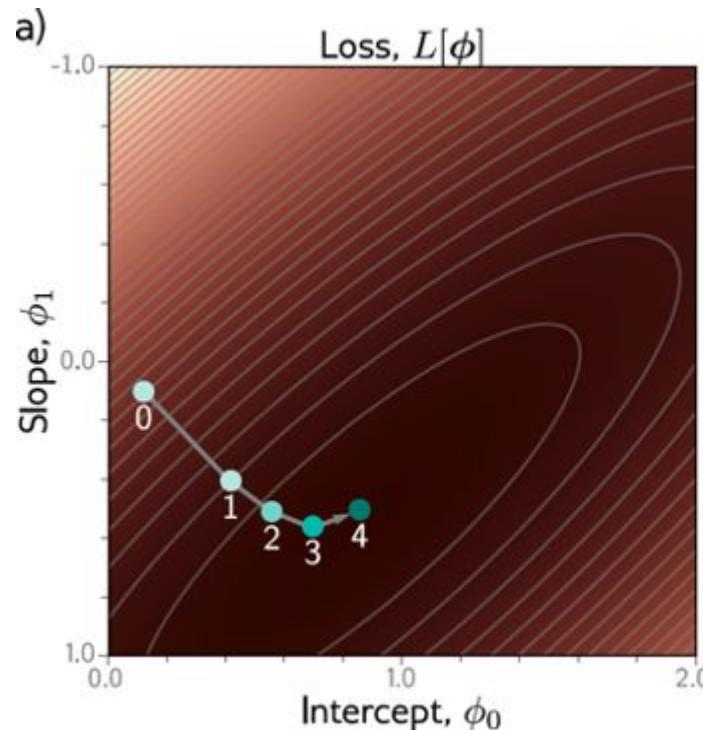
## Example: 1D Linear Regression loss function



## Example: 1D Linear Regression loss function



## Example: 1D Linear Regression loss function



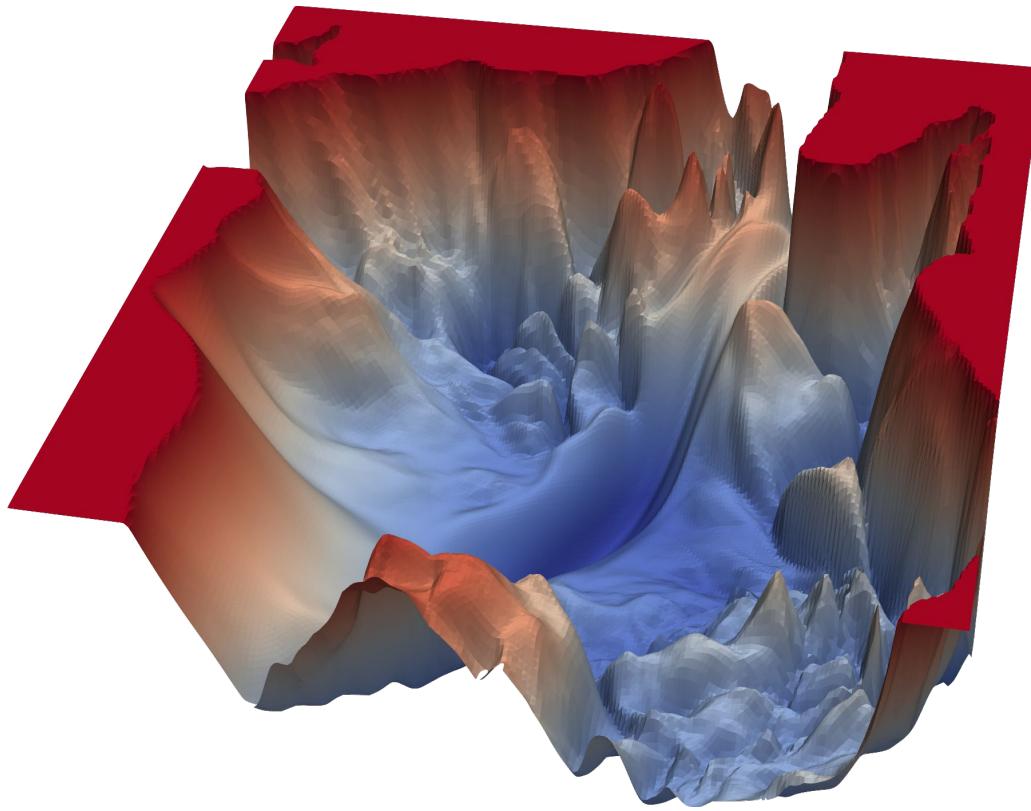
This technique is known as *gradient descent*.

## Why all the hassle?

- I know from analysis that the line has a closed form! We can find a global minima! You even showed this in the lab!

## Why all the hassle?

- I know from analysis that the line has a closed form! We can find a global minima! You even showed this in the lab!
  - Yes, but the line is a very simple model. More complicated models give non-convex loss surfaces.



**“Visualizing the loss landscapes of neural nets”**

<https://www.cs.umd.edu/~tomg/project/landscapes/>

## Why all the hassle?

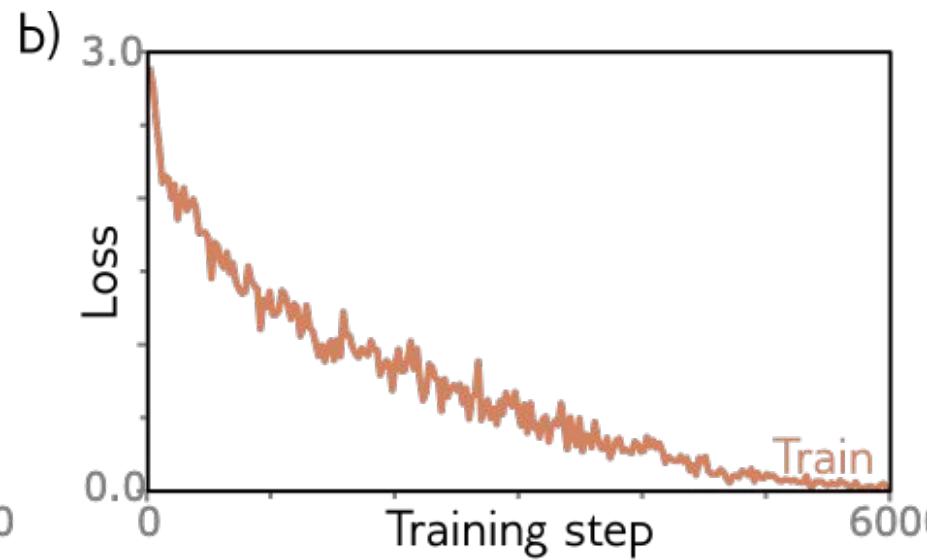
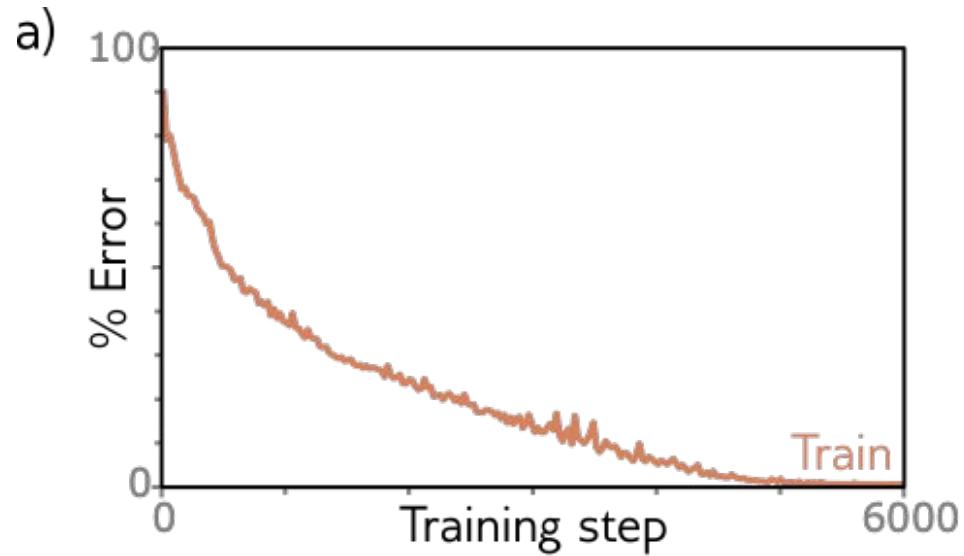
- I know from analysis that the line has a closed form! We can find a global minima! You even showed this in the lab!
  - Yes, but the line is a very simple model. More complicated models give non-convex loss surfaces.
- We could try all of the slope and intercept combinations!

# Why all the hassle?

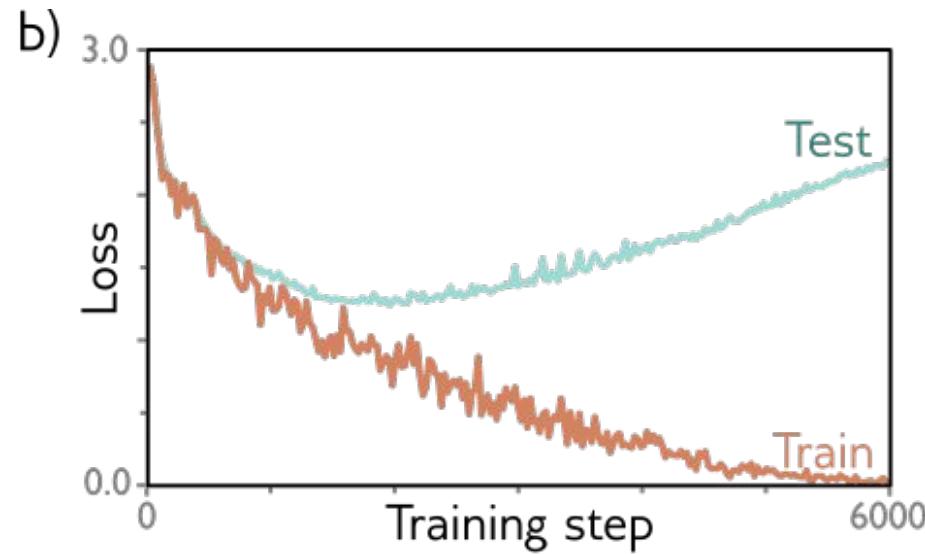
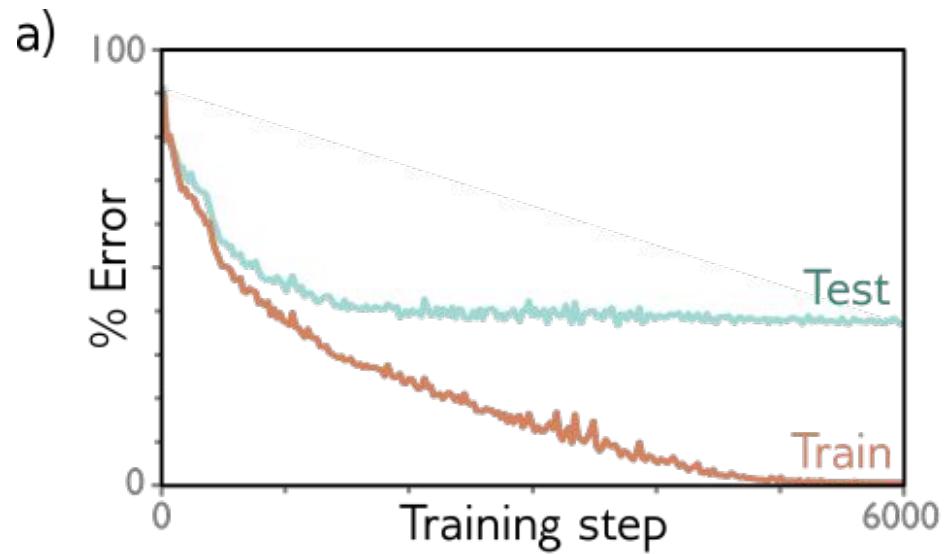
- I know from analysis that the line has a closed form! We can find a global minima! You even showed this in the lab!
  - Yes, but the line is a very simple model. More complicated models give non-convex loss surfaces.
- We could try all of the slope and intercept combinations!
  - Plausible way of optimizing for a line, not tractable for millions of parameters.

# Measuring performance

# Measuring performance

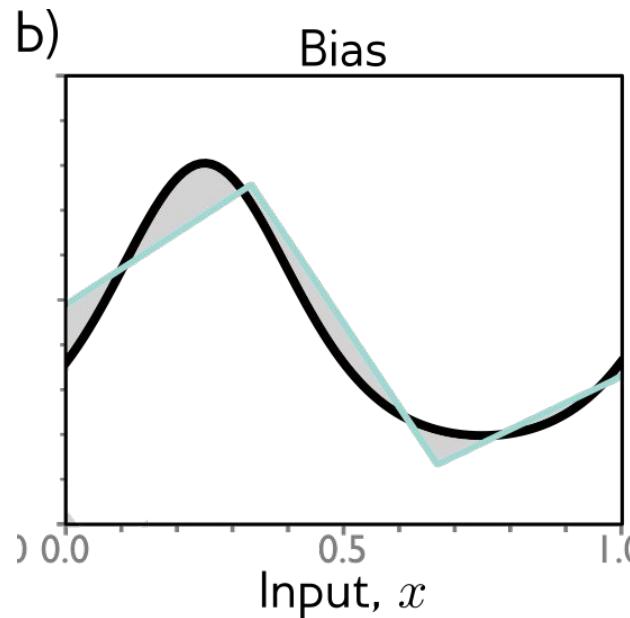


# Measuring performance



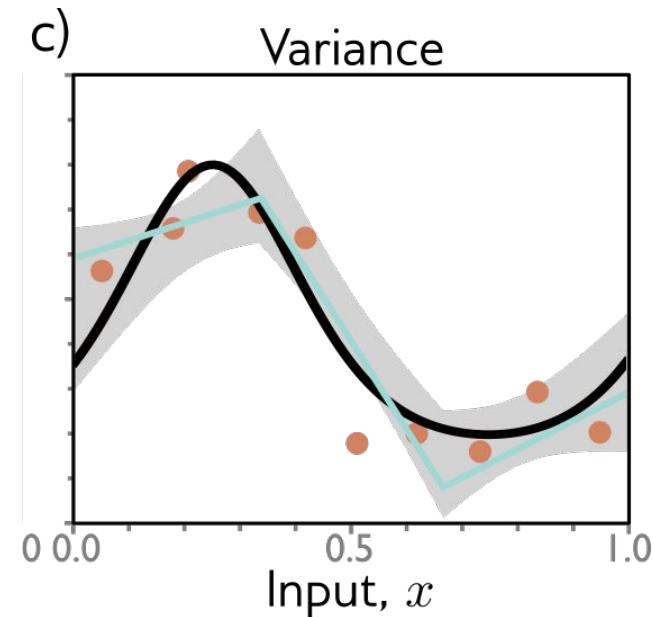
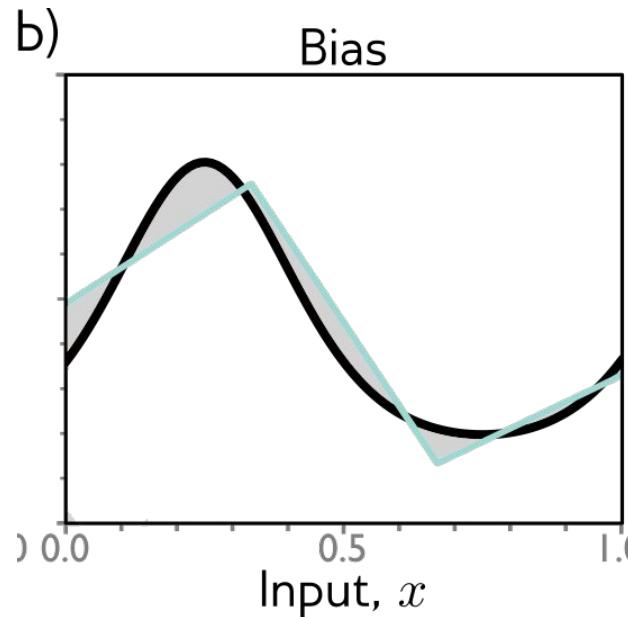
# Bias-Variance Tradeoff

- Bias = deviation from the mean of the function due to model limitations

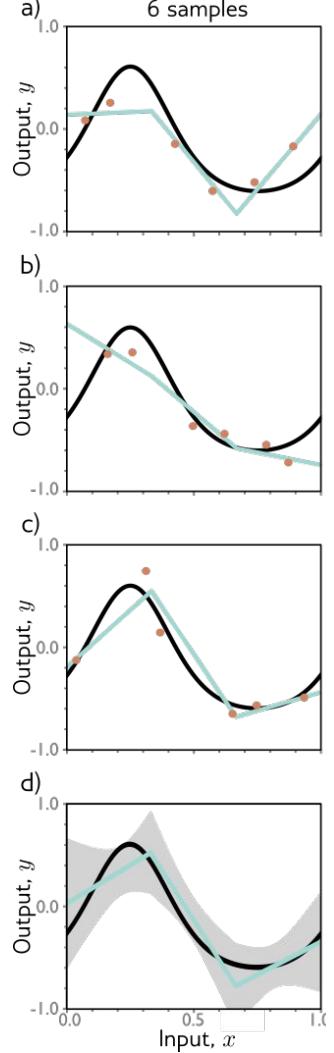


# Bias-Variance Tradeoff

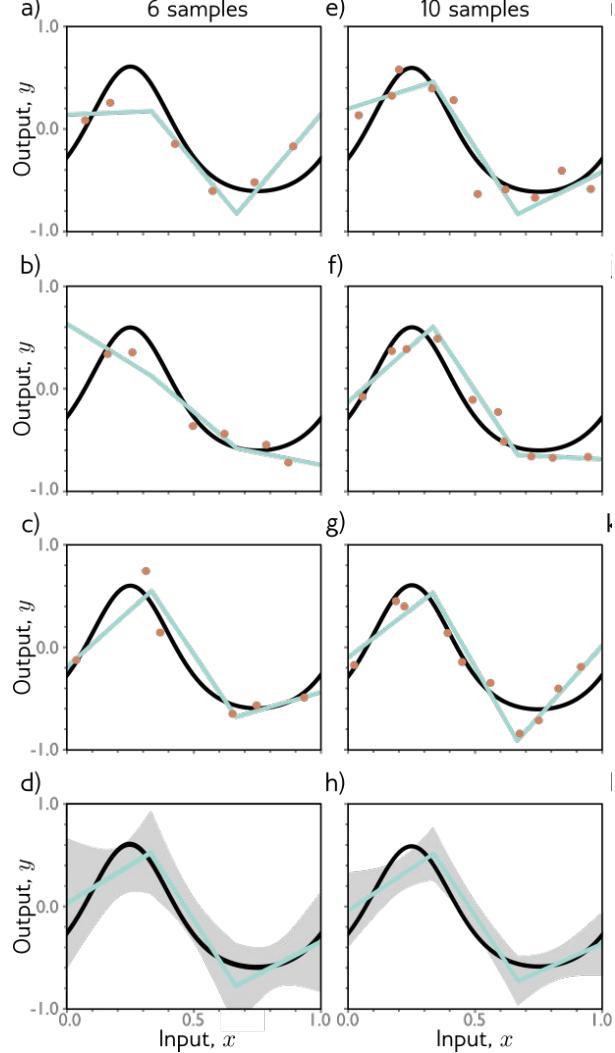
- **Bias** = deviation from the mean of the function due to model limitations
- **Variance** = uncertainty in the model due to the training set



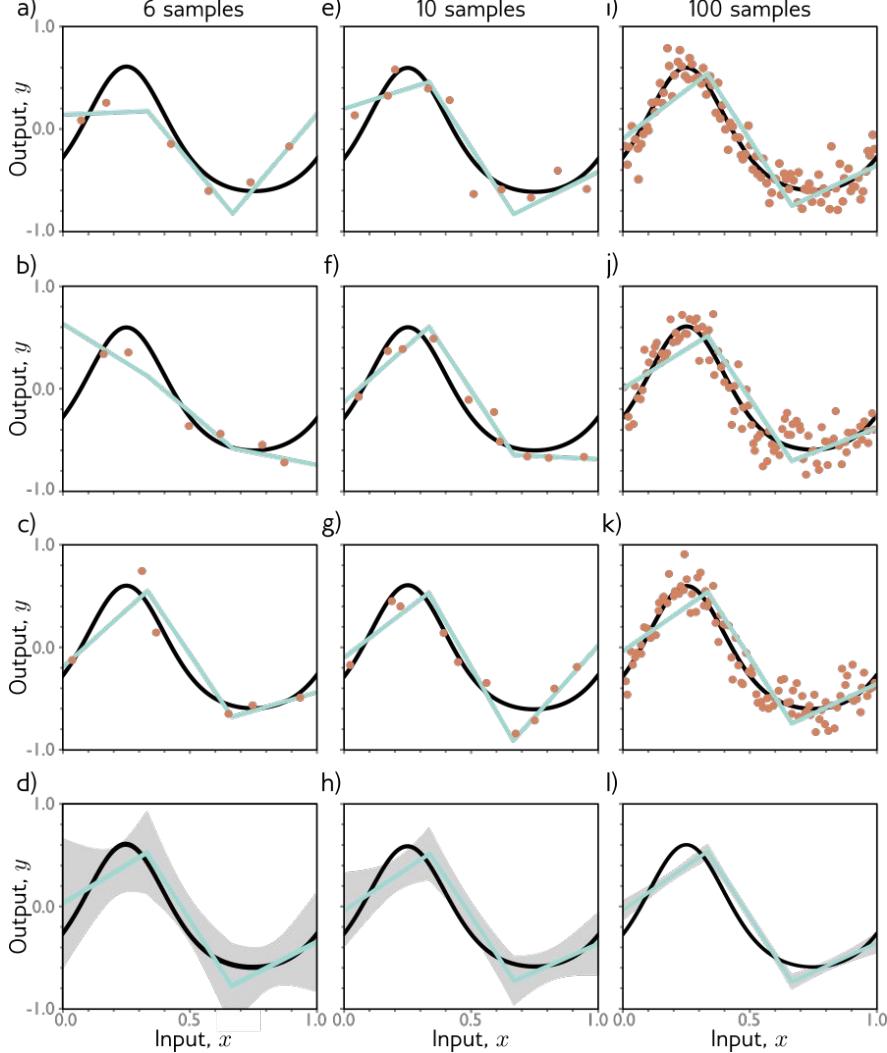
# Reducing Variance



# Reducing Variance

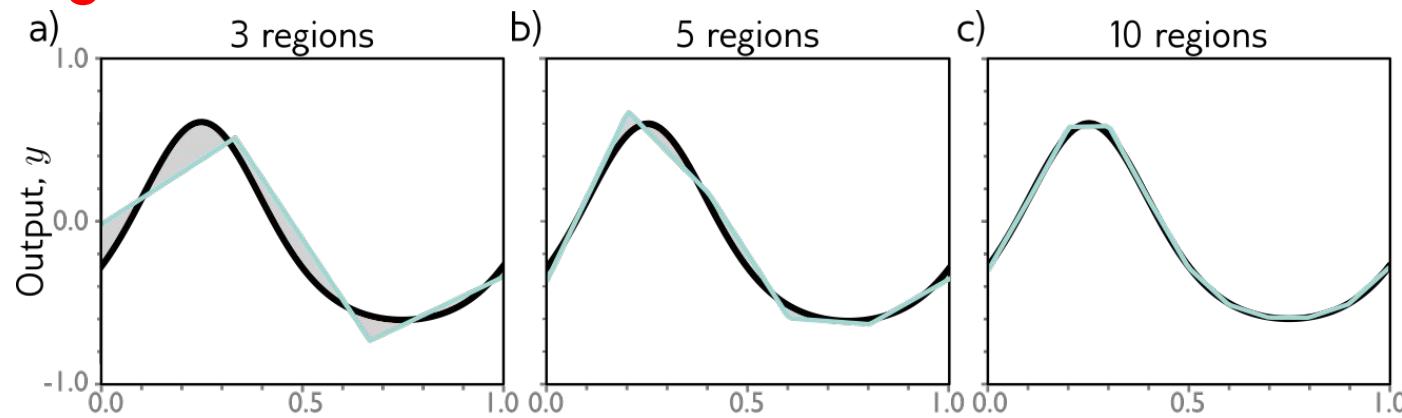


# Reducing Variance

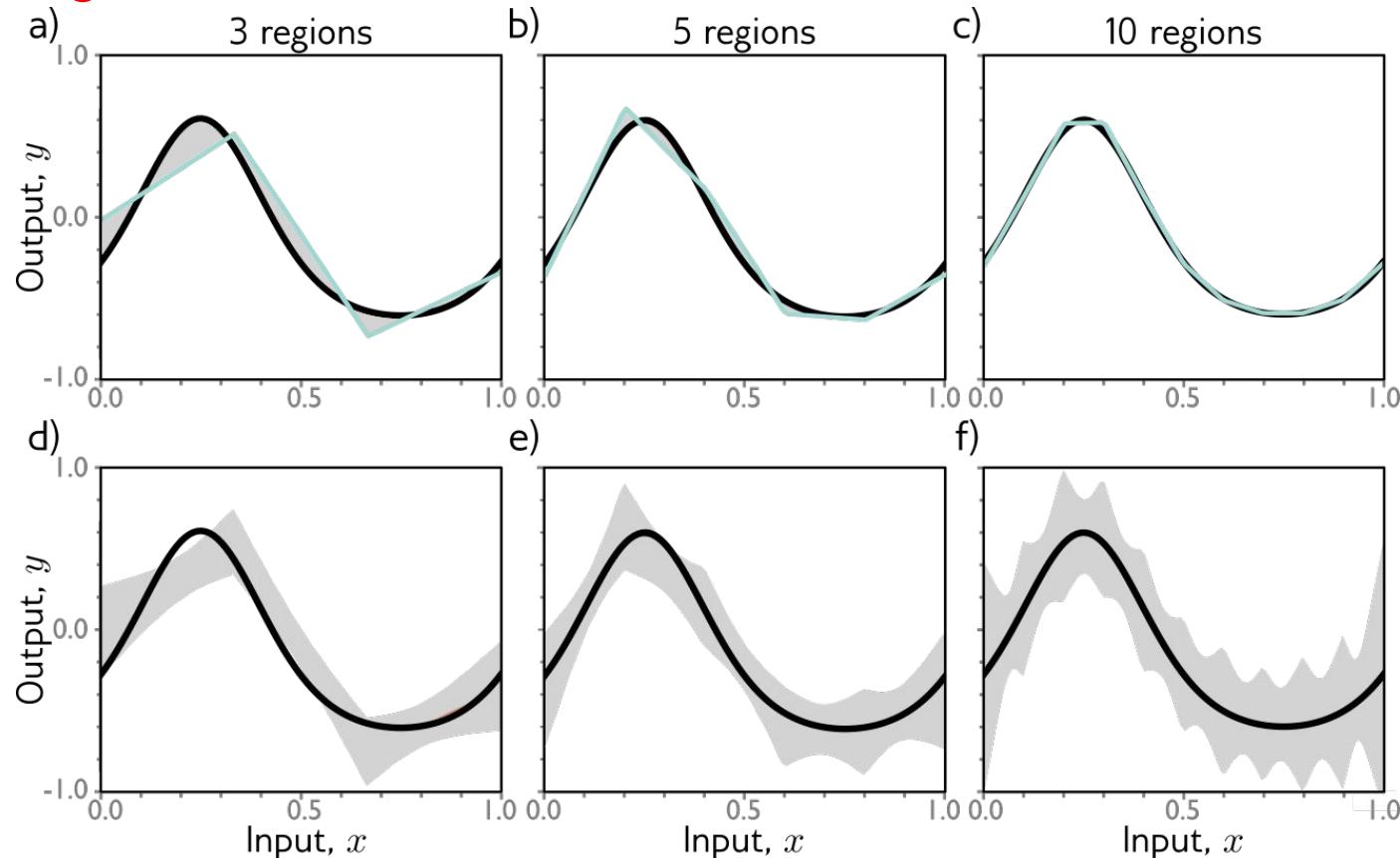


Low variance, but  
high bias  
= **under-fitting**

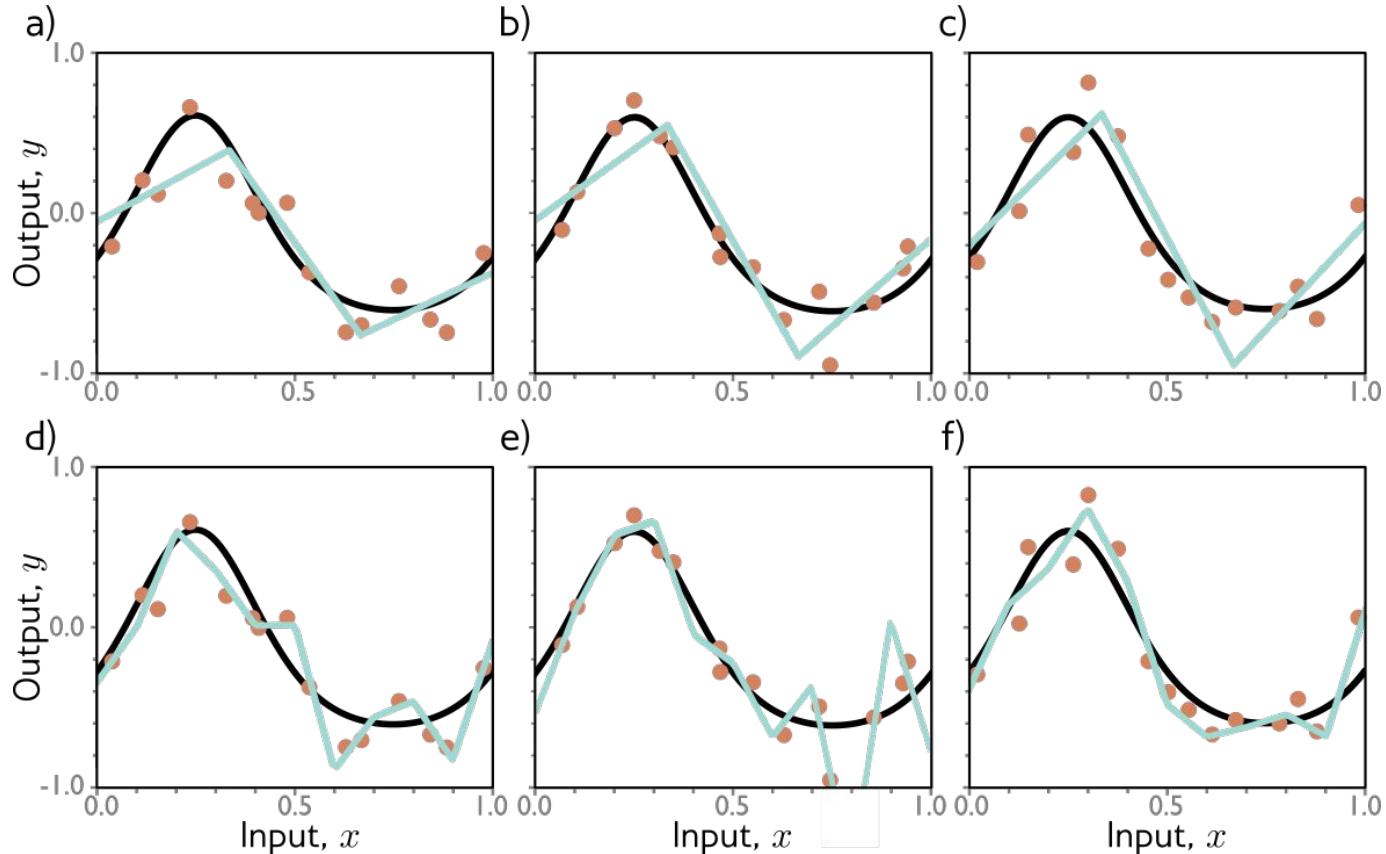
# Reducing Bias



# Reducing Bias



# Low bias, high variance: over-fitting



# Regularization

- Standard loss function:

$$\begin{aligned}\hat{\phi} &= \operatorname{argmin}_{\phi} [L[\phi]] \\ &= \operatorname{argmin}_{\phi} \left[ \sum_{i=1}^I \ell_i[\mathbf{x}_i, \mathbf{y}_i] \right]\end{aligned}$$

- Regularization adds an extra term:

$$\hat{\phi} = \operatorname{argmin}_{\phi} \left[ \sum_{i=1}^I \ell_i[\mathbf{x}_i, \mathbf{y}_i] + \lambda \cdot g[\phi] \right]$$

## L2 Regularization

- **Very general regularization technique:**

$$\hat{\phi} = \operatorname{argmin}_{\phi} \left[ L[\phi, \{\mathbf{x}_i, \mathbf{y}_i\}] + \lambda \sum_j \phi_j^2 \right]$$

## L2 Regularization

- Very general regularization technique:

$$\hat{\phi} = \operatorname{argmin}_{\phi} \left[ L[\phi, \{\mathbf{x}_i, \mathbf{y}_i\}] + \lambda \sum_j \phi_j^2 \right]$$

If the parameters are “big”,  
loss gets bigger

# L2 Regularization

- Very general regularization technique:

$$\hat{\phi} = \operatorname{argmin}_{\phi} \left[ L[\phi, \{\mathbf{x}_i, \mathbf{y}_i\}] + \lambda \sum_j \phi_j^2 \right]$$

alpha controls the strength  
of the regularization term

If the parameters are “big”,  
loss gets bigger

# L2 Regularization

- **Very general regularization technique:**

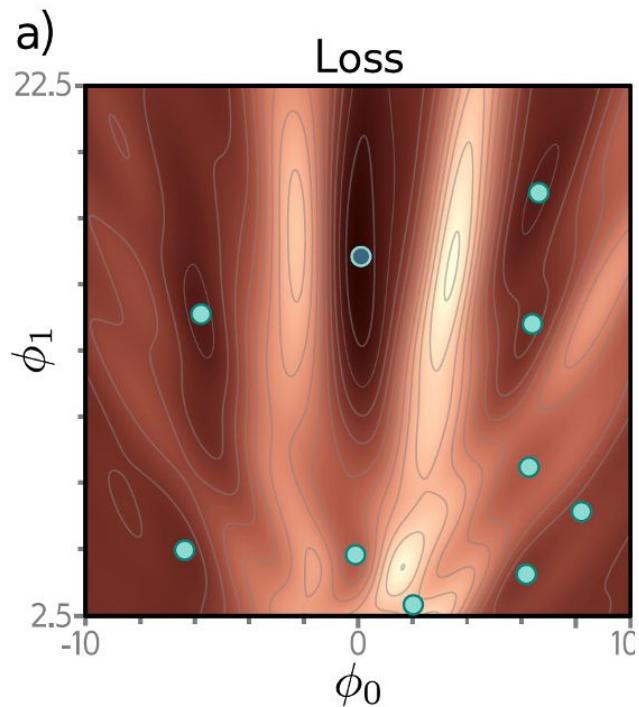
$$\hat{\phi} = \underset{\phi}{\operatorname{argmin}} \left[ L[\phi, \{x_i, y_i\}] + \lambda \sum_j \phi_j^2 \right]$$

alpha controls the strength  
of the regularization term

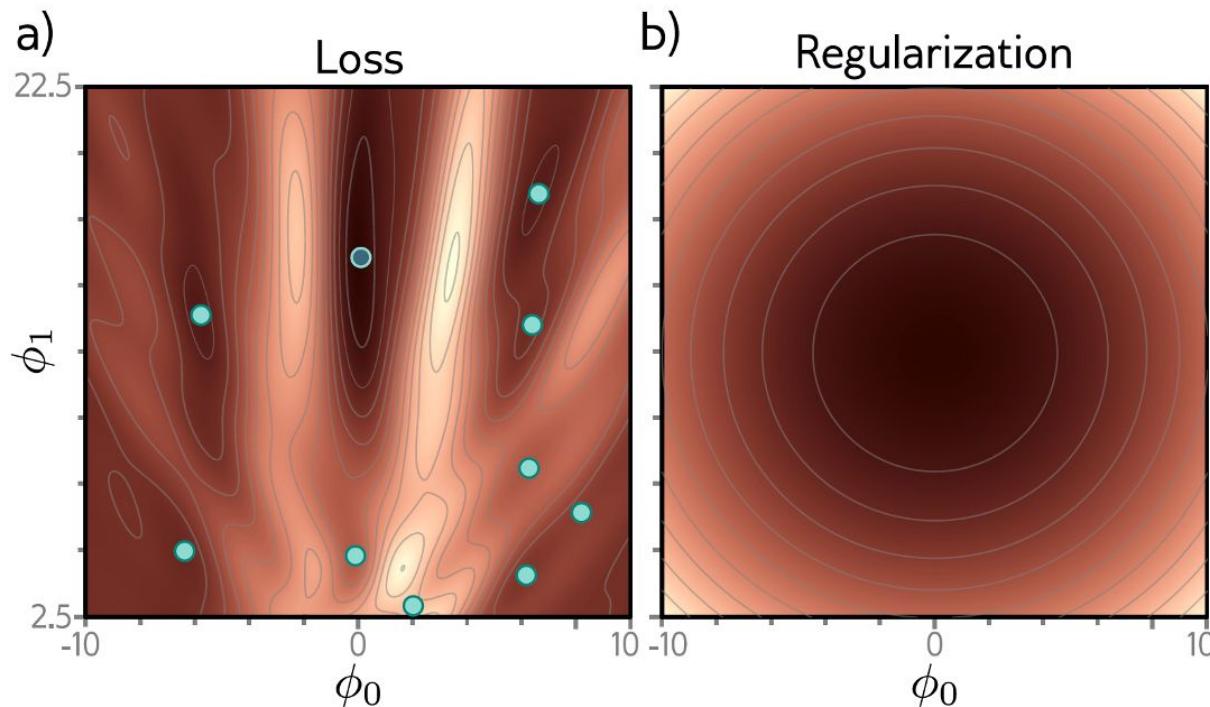
If the parameters are “big”,  
loss gets bigger

- Also called “ridge regression” or “Tikhonov regularization”
- In neural networks it is usually called “weight decay”

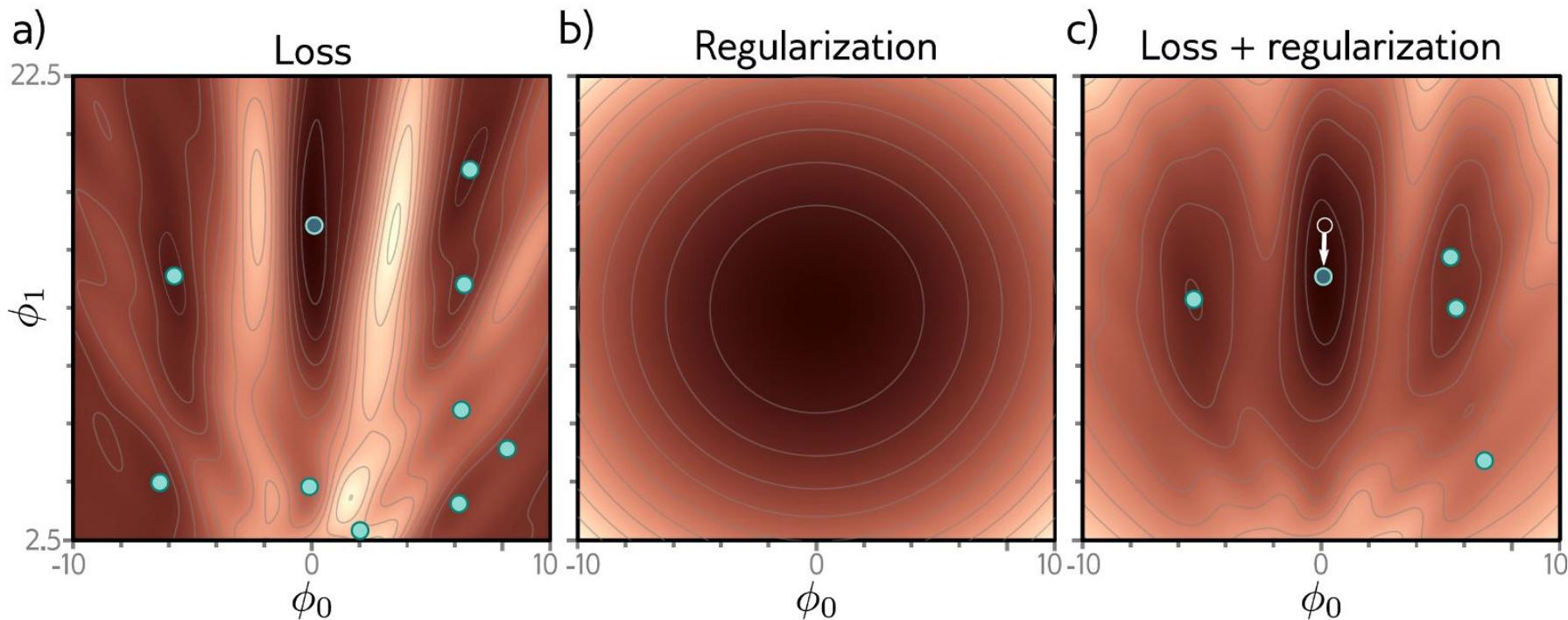
# Loss surface intuition



# Loss surface intuition



# Loss surface intuition



# Regularization

- Regularization “simplifies” the model == reduces over-fitting

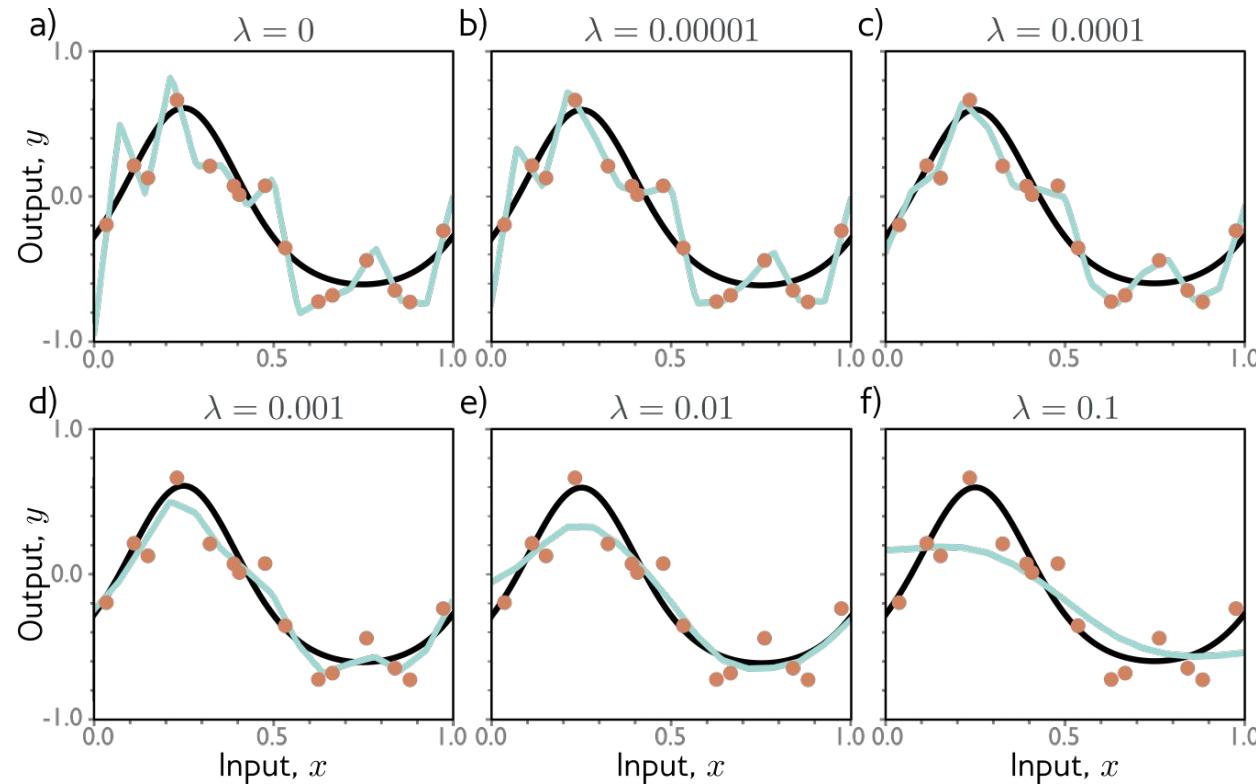
# Regularization

- Regularization “simplifies” the model == reduces over-fitting
- Too much regularization can also lead to under-fitting

# Regularization

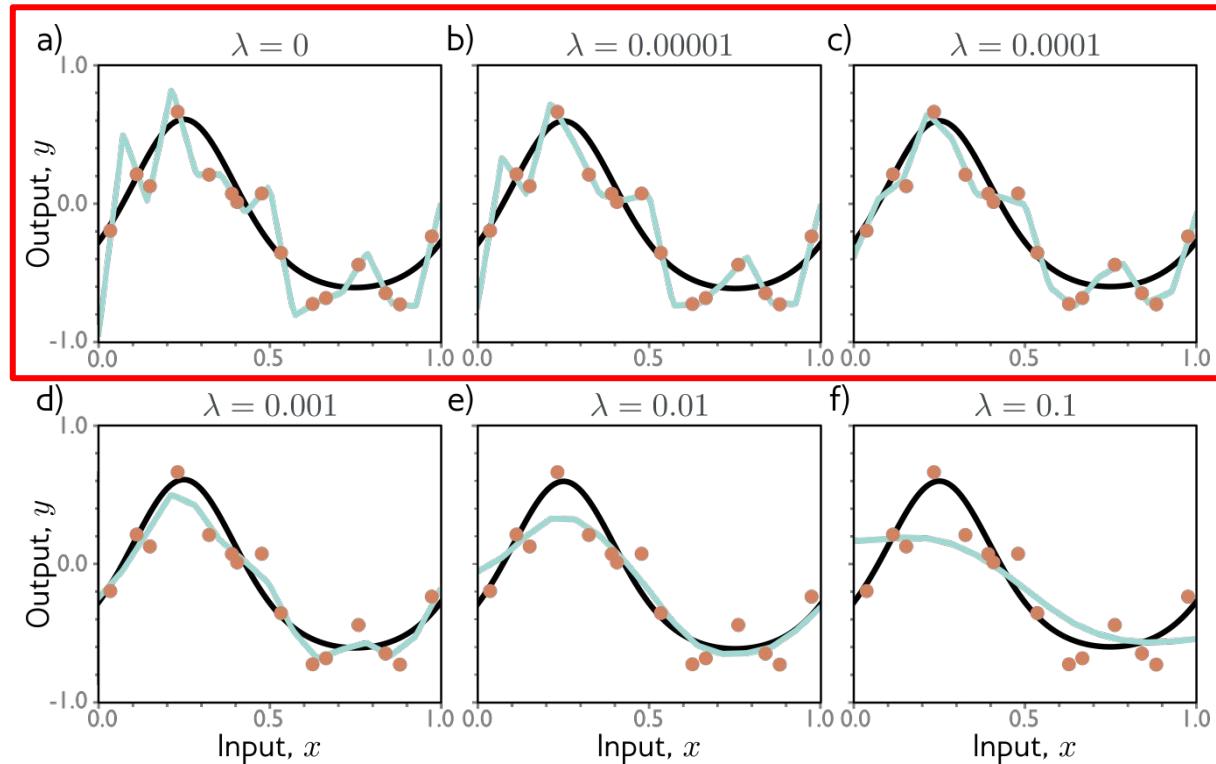
- Regularization “simplifies” the model == reduces over-fitting
- Too much regularization can also lead to under-fitting
- We can control the regularization term using *alpha*, a *hyper-parameter*

# Regularization - alpha



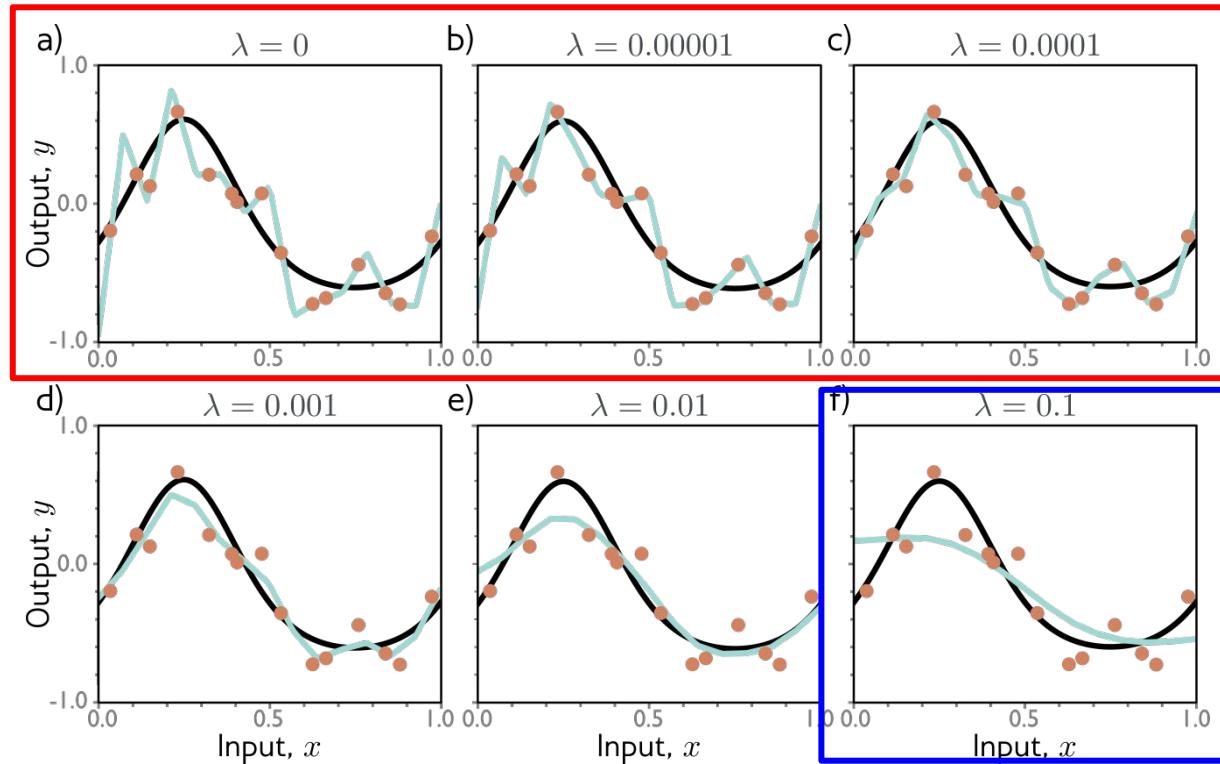
# Regularization - alpha

alpha too small = over-fitting



# Regularization - alpha

alpha too small = over-fitting



alpha too large = under-fitting

# Hyper-parameters

- We can control the regularization term using *alpha*, a *hyper-parameter*

# Hyper-parameters

- We can control the regularization term using *alpha*, a *hyper-parameter*
- Other hyper-parameters:
  - Number of model learnable parameters
  - Number of training steps
  - Learning rate (next courses)
  - Batch size (next courses)

# Hyper-parameters

- We can control the regularization term using *alpha*, a *hyper-parameter*
- Other hyper-parameters:
  - Number of model learnable parameters
  - Number of training steps
  - Learning rate (next courses)
  - Batch size (next courses)
- We need to be careful when we pick the hyper-parameters

# Train/validation/test

- Picking hyper-parameters on the test set can lead to *hyper-parameter over-fitting*
- We find hyper-parameters by evaluating the model on a **validation split**
- The final performance of the model is the performance on the test set

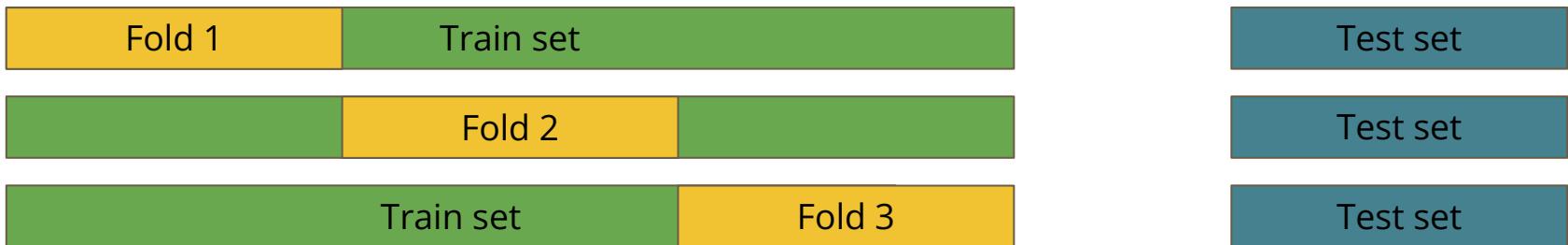


# Train/validation/test

- Picking hyper-parameters on the test set can lead to *hyper-parameter over-fitting*
- We find hyper-parameters by evaluating the model on a **validation split**
- The final performance of the model is the performance on the test set



- When we don't have much data, we can **cross-validate**



## Lecture recap - 1

- The *model is a class of functions* that takes some input and gives us an approximate output that we desire

## Lecture recap - 1

- The *model is a class of functions* that takes some input and gives us an approximate output that we desire
- *Learning* means *finding “good” parameters* of a model on a training dataset (fitting the model)

## Lecture recap - 1

- The *model is a class of functions* that takes some input and gives us an approximate output that we desire
- *Learning* means *finding “good” parameters* of a model on a training dataset (fitting the model)
- Deep neural networks are very flexible models

## Lecture recap - 2

- Having very complicated models can lead to *over-fitting* (high bias)

## Lecture recap - 2

- Having very complicated models can lead to *over-fitting* (high bias)
- Having models that are too simple can lead to *under-fitting* (high variance)

## Lecture recap - 2

- We can fix under-fitting by using more data and having models that are more expressive

## Lecture recap - 2

- We can fix under-fitting by using more data and having models that are more expressive
- We can fix over-fitting by *regularizing* our model or simplifying it

## Lecture recap - 2

- We can fix under-fitting by using more data and having models that are more expressive
- We can fix over-fitting by *regularizing* our model or simplifying it
- There are special parameters that are not learned, called *hyper-parameters*

## Lecture recap - 2

- We can fix under-fitting by using more data and having models that are more expressive
- We can fix over-fitting by *regularizing* our model or simplifying it
- There are special parameters that are not learned, called *hyper-parameters*
- We should pick our hyper-parameters on a *validation subset*

# Next: Neural Networks



# References

- Understanding Deep Learning - Simon J.D. Prince
  - <http://udlbook.com/>
- Pattern Recognition and Machine Learning - Christopher Bishop