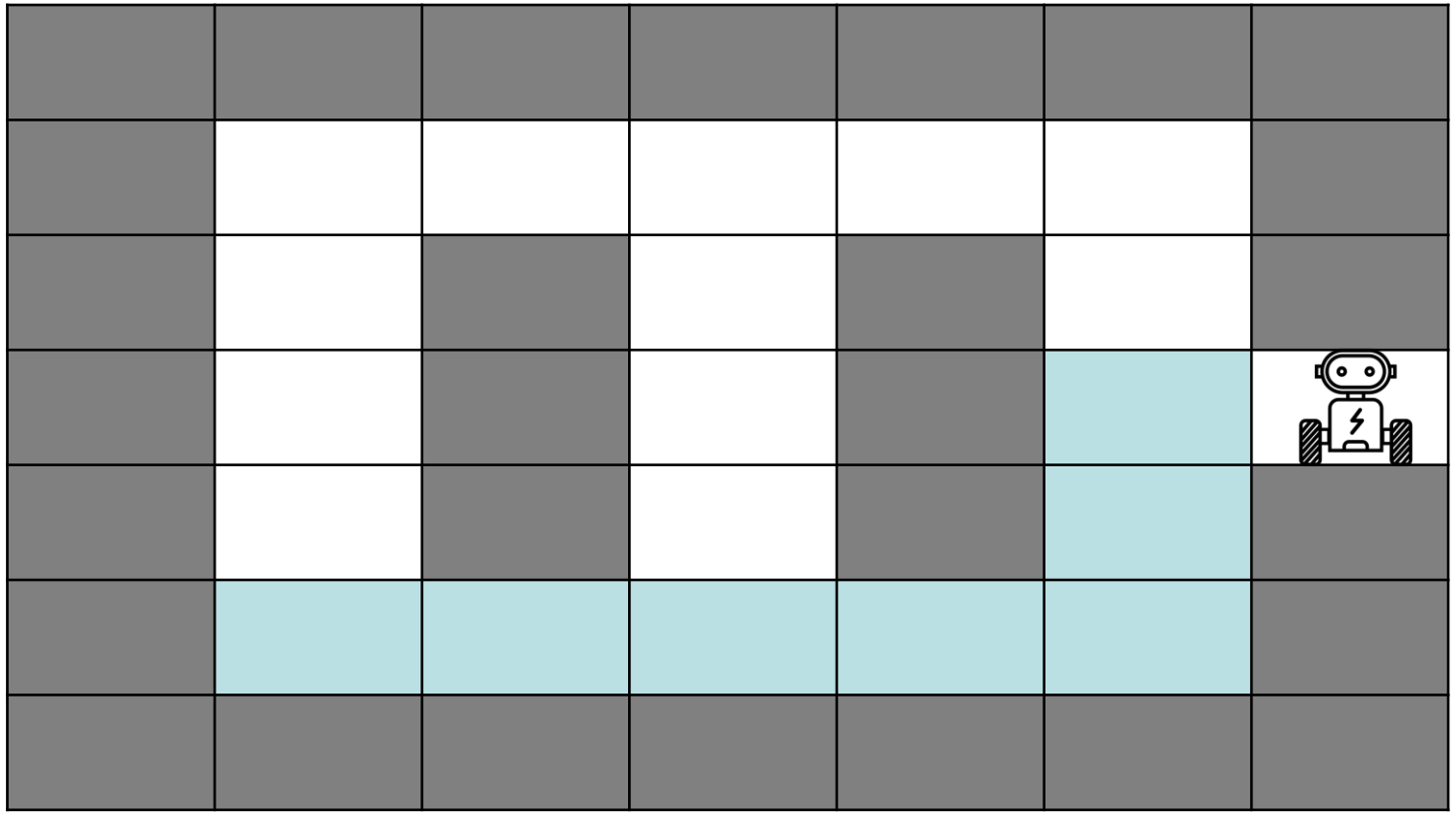
Object-Oriented Programming and Machine Learning

Daniel Tozadore (d.tozadore@ucl.ac.uk)

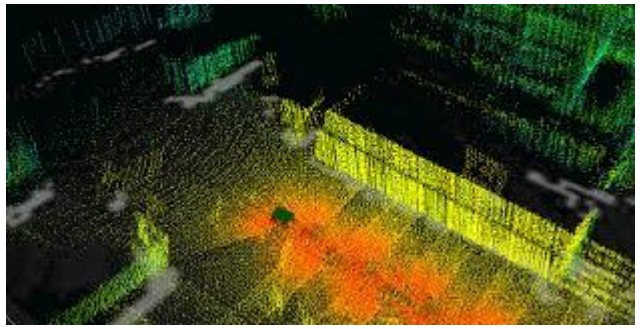
From previous sessions…

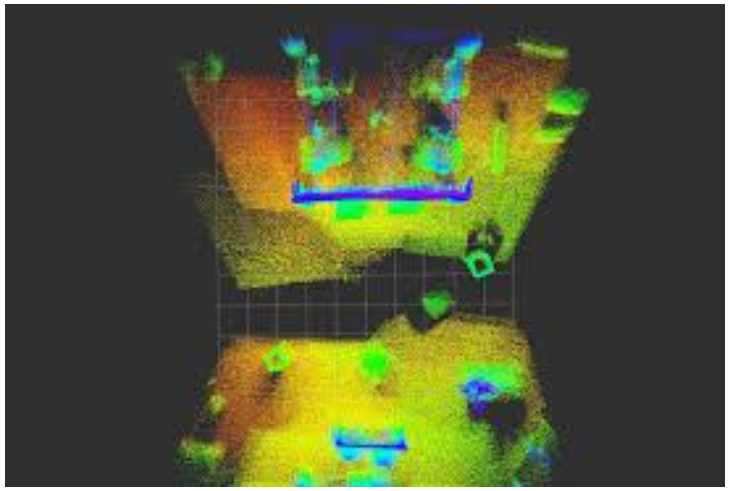
Modelling the Robot/Maze

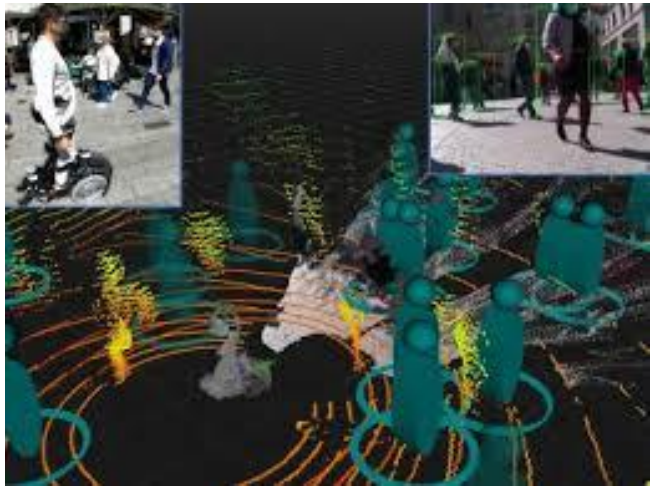
Why does that matter?

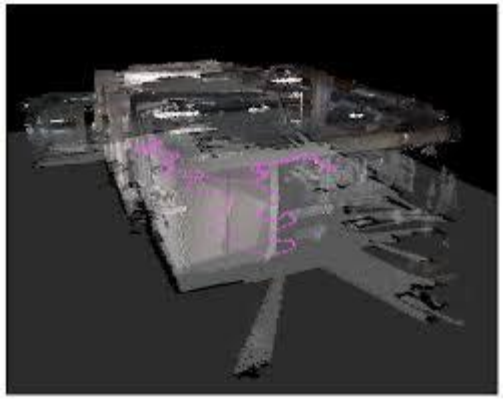


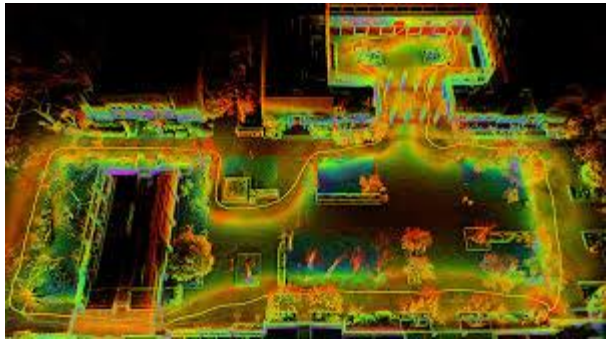
Why does that matter?







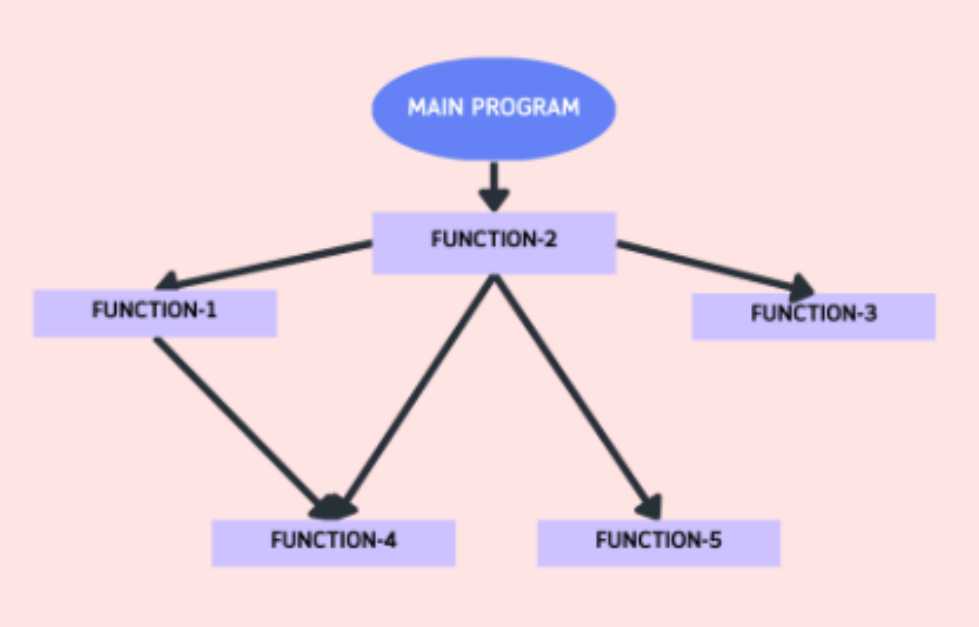




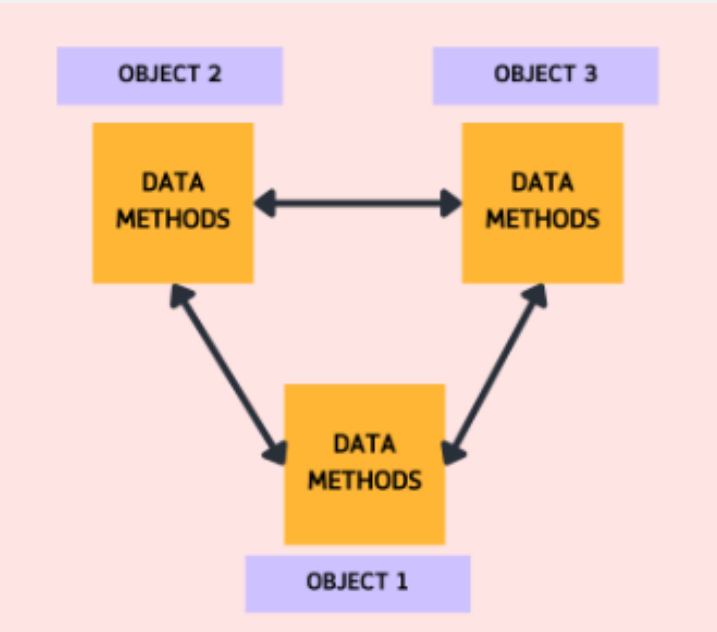
UCL

Object-Oriented Programming for robotics and AI

Procedural programming (POP) vs OOP



POP



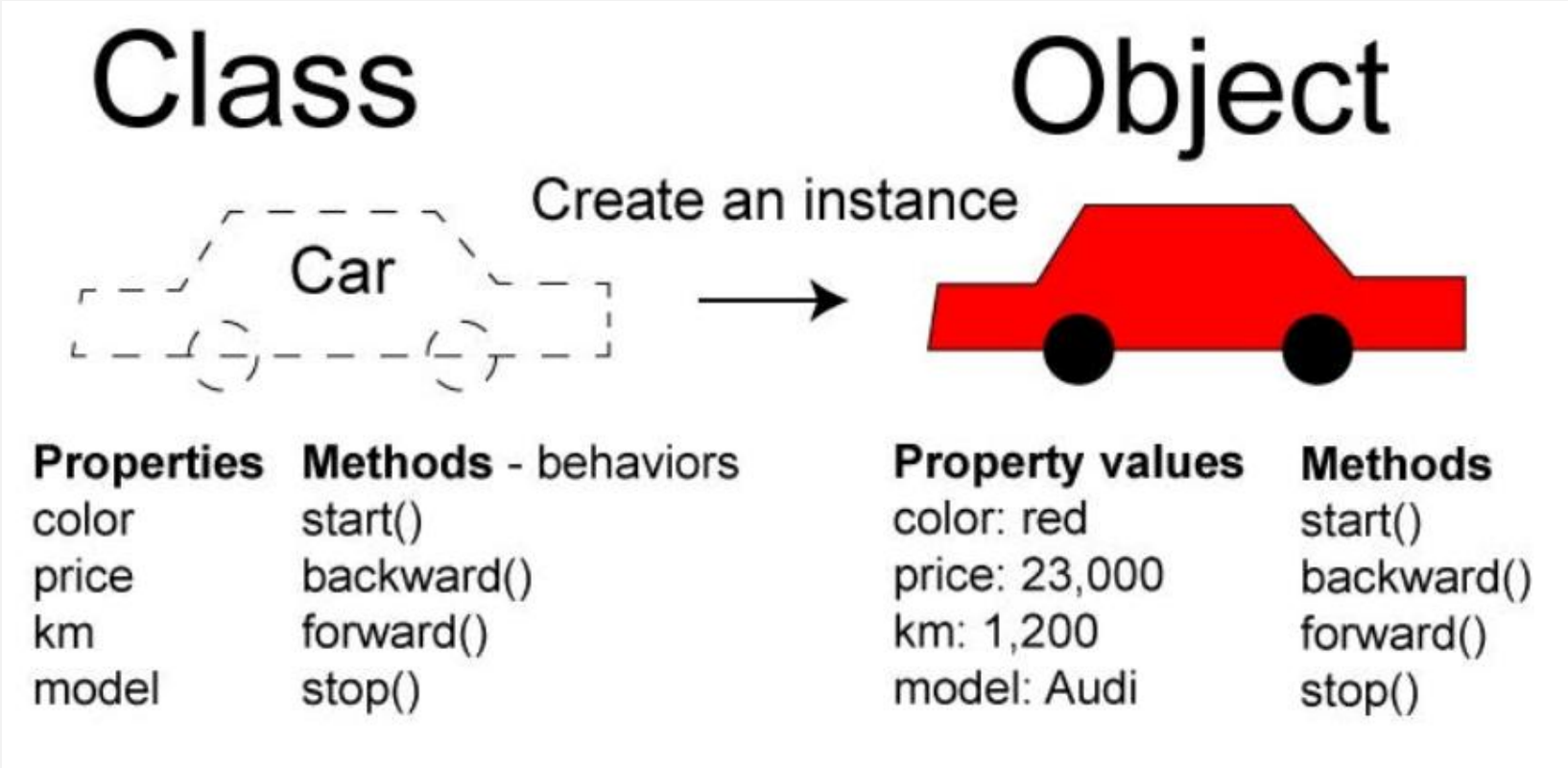
OOP

Class and Object

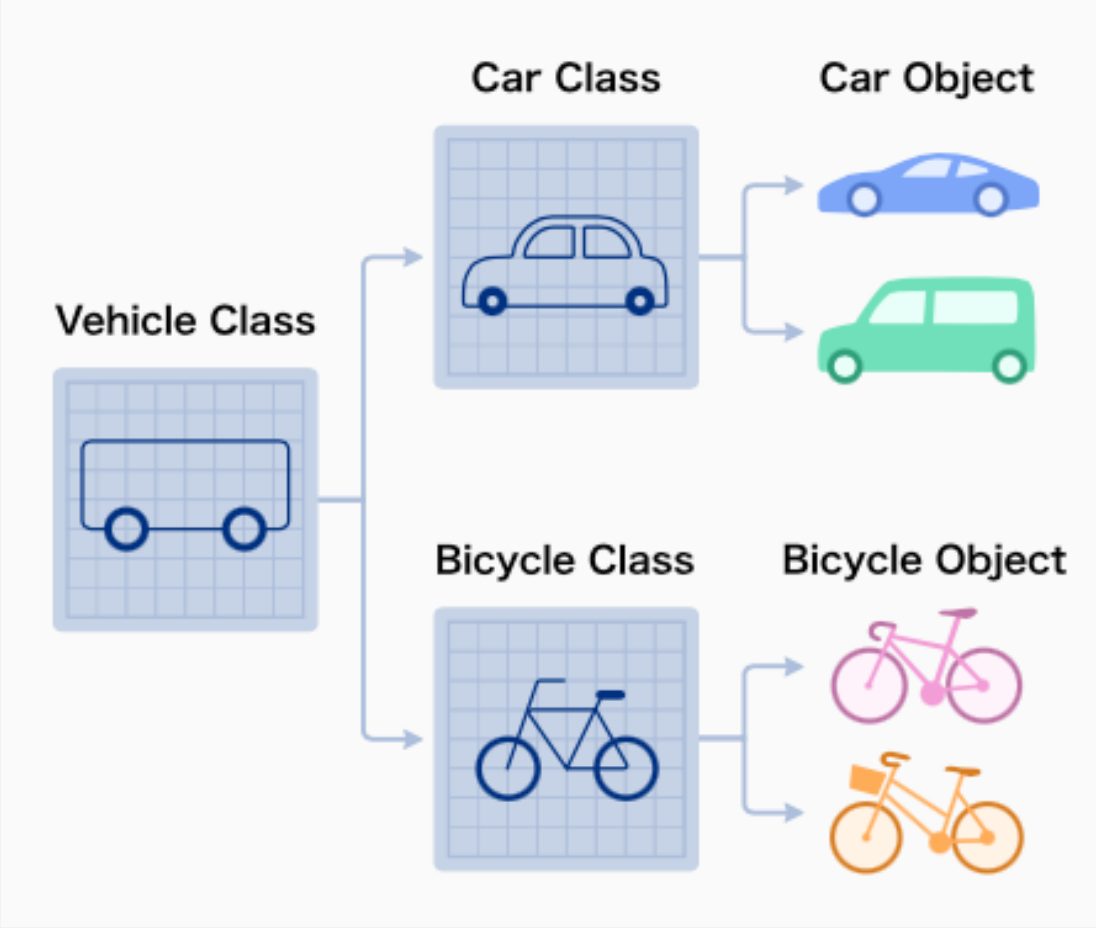
Class – define properties/attributes (variables), behaviors/functionalities (methods)

Object – instances with specific values of properties, their behaviours

An Object is a data structure with both data (variables called attributes) And code (functions called methods)



Using classes and objects



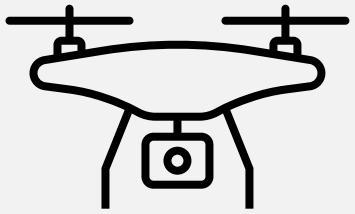
How about our code with robots?

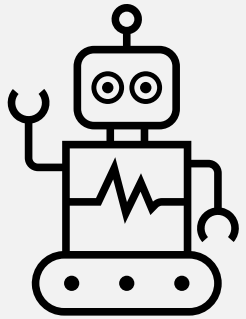
Can we have different types of robots?

Let's try to model them in our code!

Having multiple robots in our map

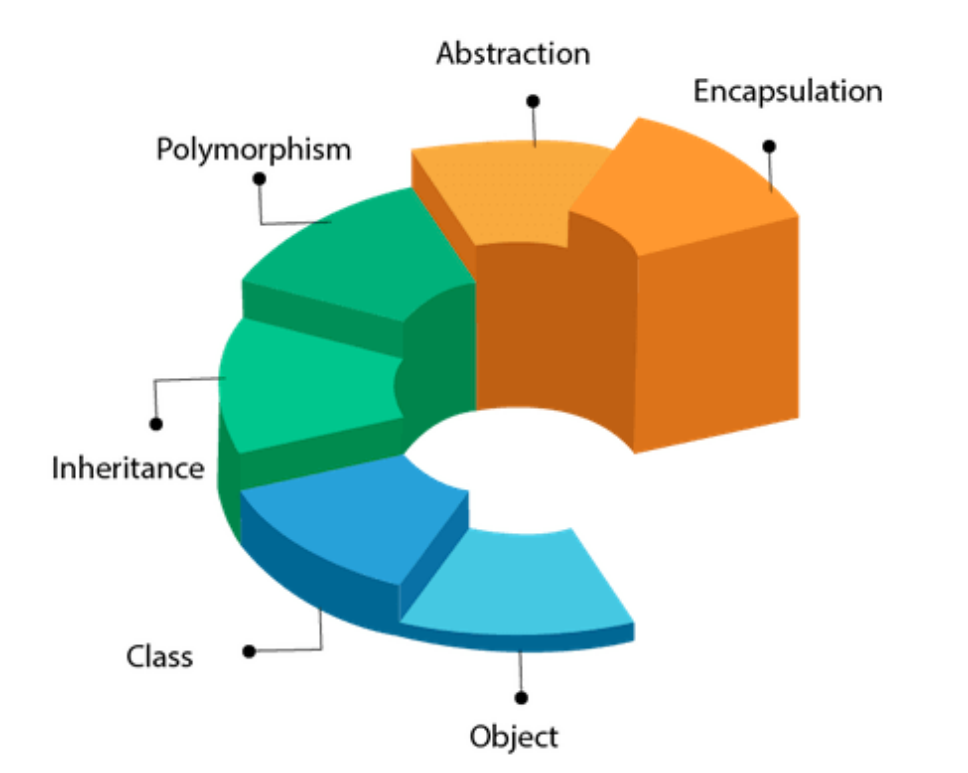
Consider two robots: a wheeled and a drone.





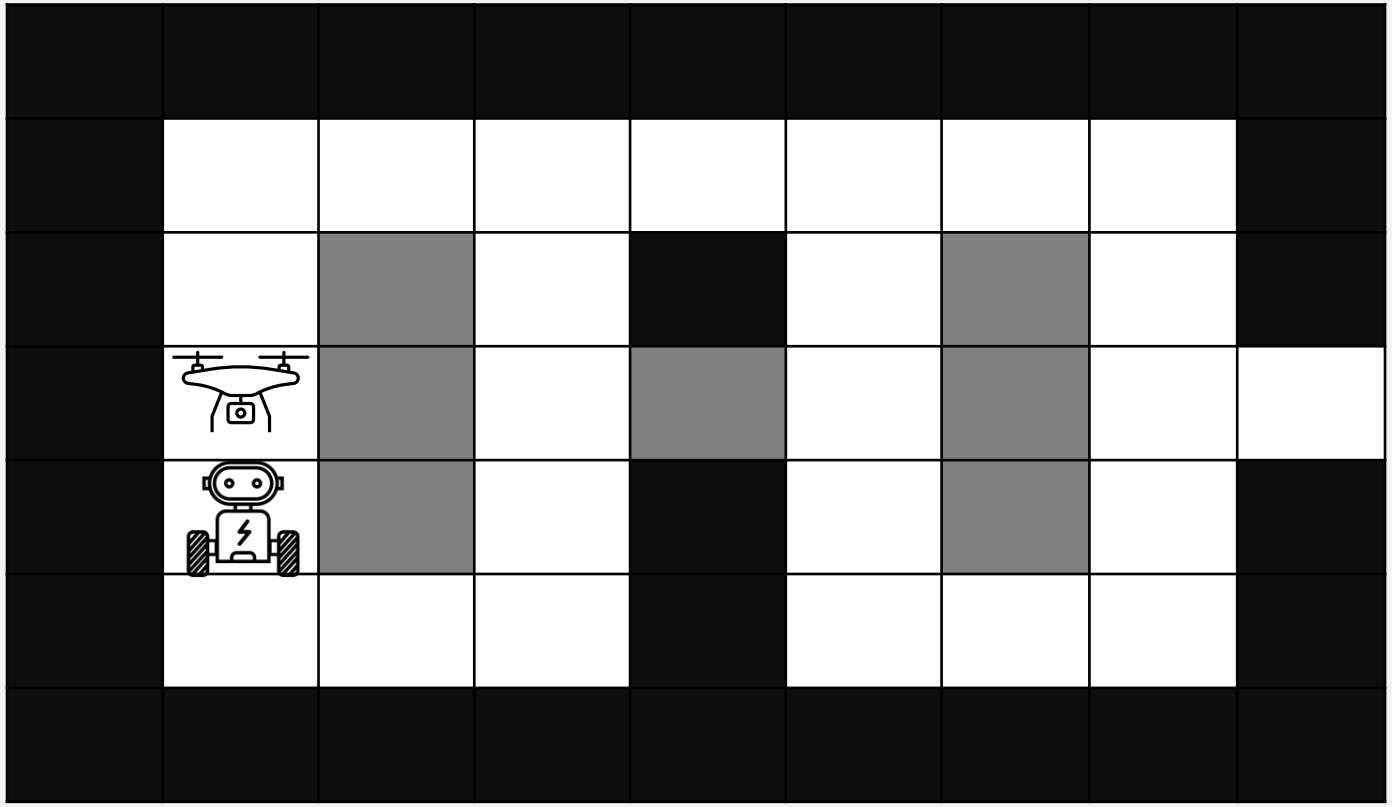
Which variables they will have in common?
Which ones will be exclusive?

Important concepts in OOP

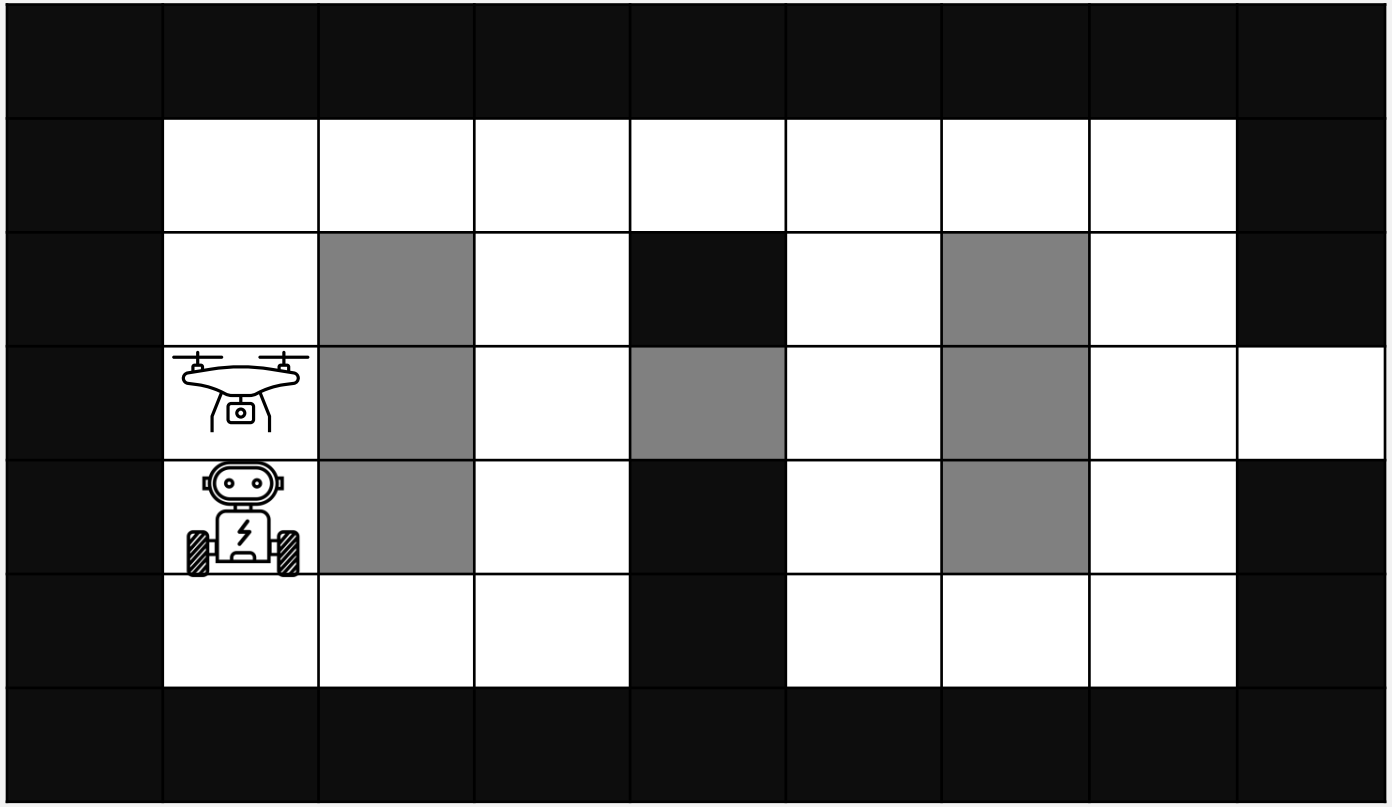


- **Classes** (types)
- **Objects and Instances**
- **Encapsulation** : Data manipulation
- **Inheritance** : Code re-use
- **Polymorphism** : Code efficiency

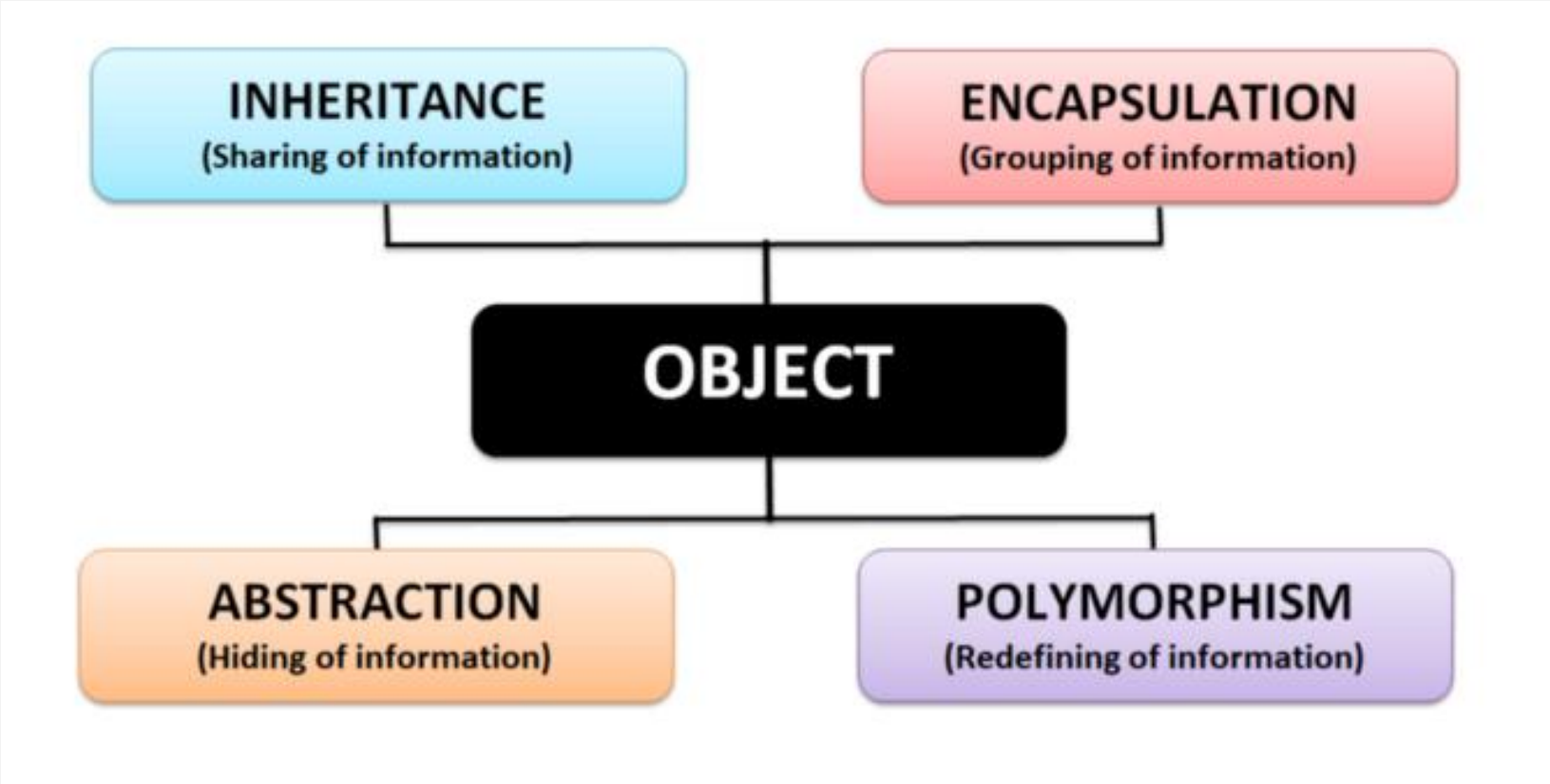
How about the obstacles?



What if obstacles have different highs?



OOP concepts - closer look



Python

- Python is an interpreted, object-oriented programming language.
- Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for rapid code development
  - No need to declare a type of variable. Easy syntax, readable, modular.
  - Indentation to identify separate blocks of code.

```
alist = ['a', 'b', 'c']
def my_function(al):
    print(al)
my_function(alist)
```

Output:

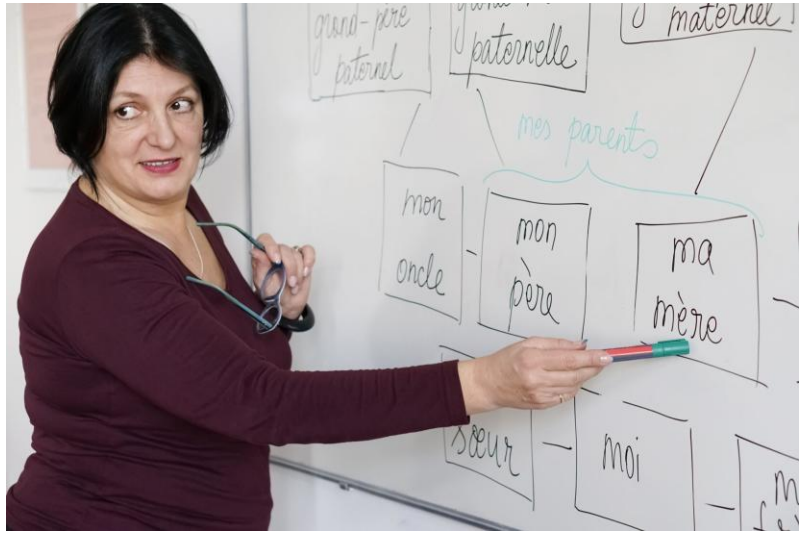
```
['a', 'b', 'c']
```

Tasks in python

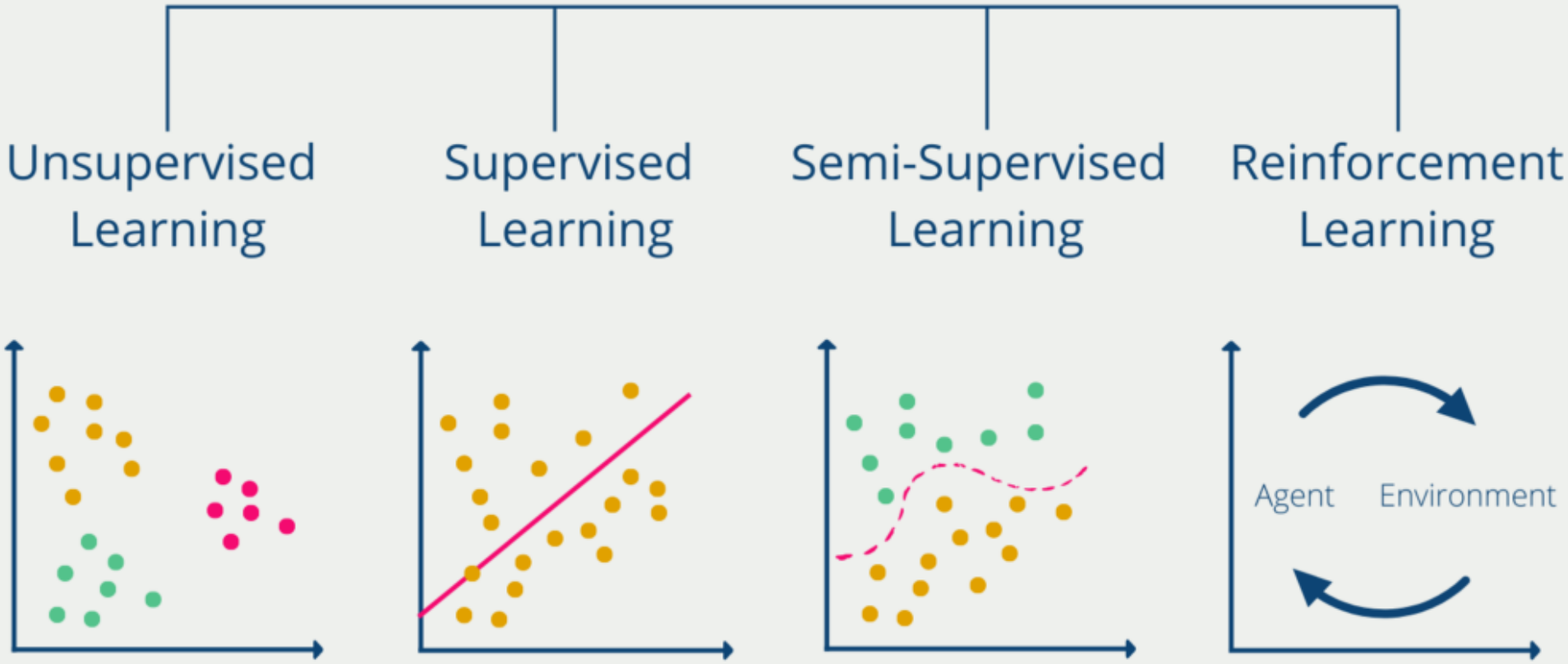
- Translate your code to python!
- Implement a linked list using classes
- Use the linked list to keep track of the visited nodes
- Use a tree to look for the visited nodes
- Implement your map as a graph
- Implement the searches for Breadth and Depth first methods

Machine Learning

How do students learn?



Machine Learning



How do machines learn?

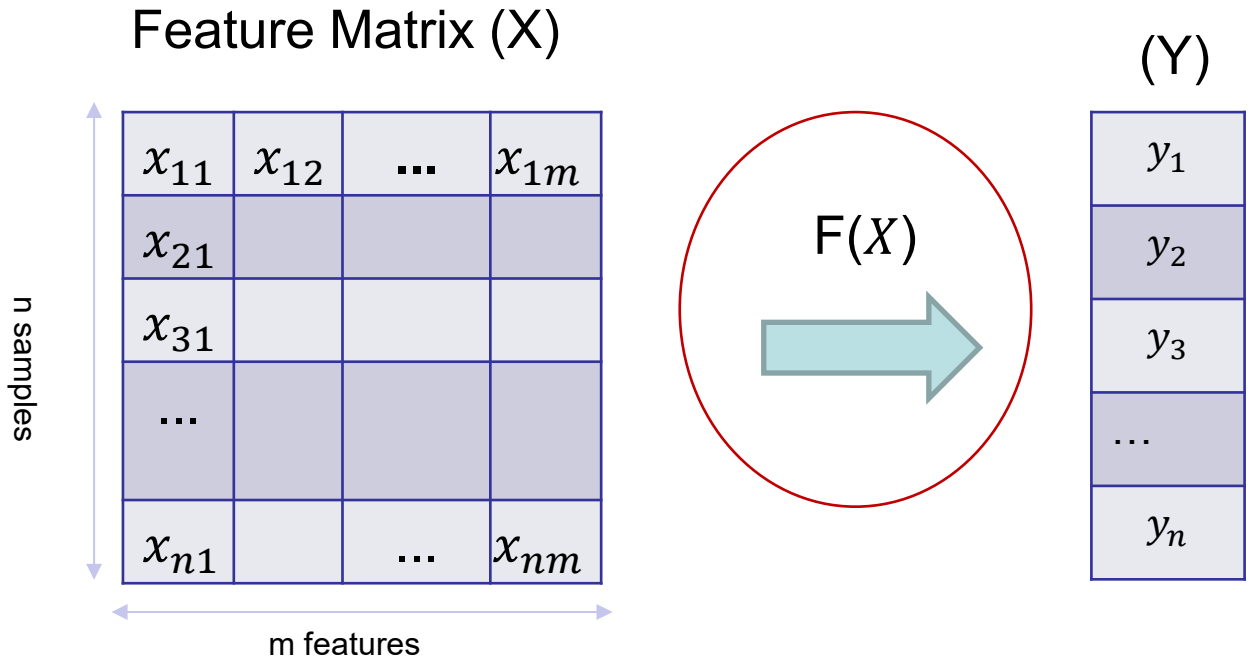
Feature	Supervised Learning	Unsupervised Learning	Reinforcement Learning
Data Type	Labeled	Unlabeled	Agent interacts with environment
Goal	Predict outcomes	Discover patterns or structure	Learn optimal actions via rewards
Examples	Email spam detection, house price prediction	Customer segmentation, topic modeling	Game playing, robotic control
Algorithms	SVM, Decision Trees, Neural Networks	K-Means, PCA, Autoencoders	Q-Learning, Deep Q-Networks, Policy Gradients
Evaluation	Accuracy, Precision, Recall	Silhouette Score, Cluster Purity	Cumulative reward, convergence rate
Human Intervention	Requires labeled data	Minimal labeling required	Requires reward signal design
Learning Process	Learns from examples	Learns from data structure	Learns from trial and error

Key Concepts

- Features: Input variables used for prediction.
- Labels: Output or target variable.
- Training: Process of learning from data.
- Testing: Evaluating model performance on unseen data.

Supervised Learning

Classification algorithms: predict a label based on features!



Supervised Learning

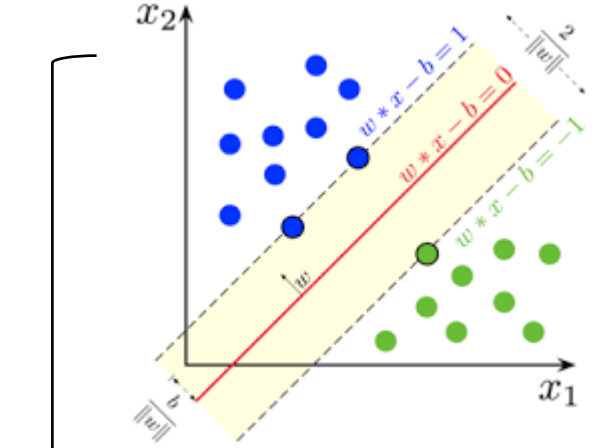
Feature Matrix (X)

x_{11}	x_{12}	...	x_{1m}
x_{21}			
x_{31}			
...			
x_{n1}		...	x_{nm}

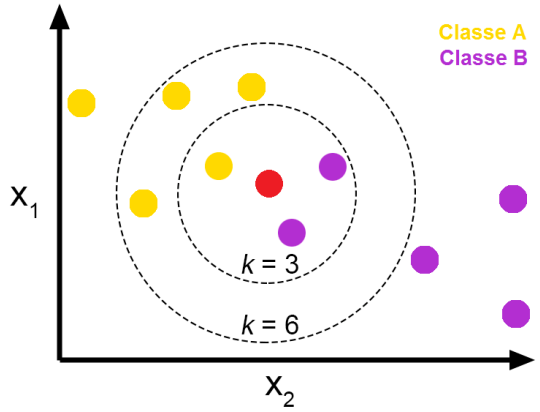
n samples

m features

F(X)



SVM



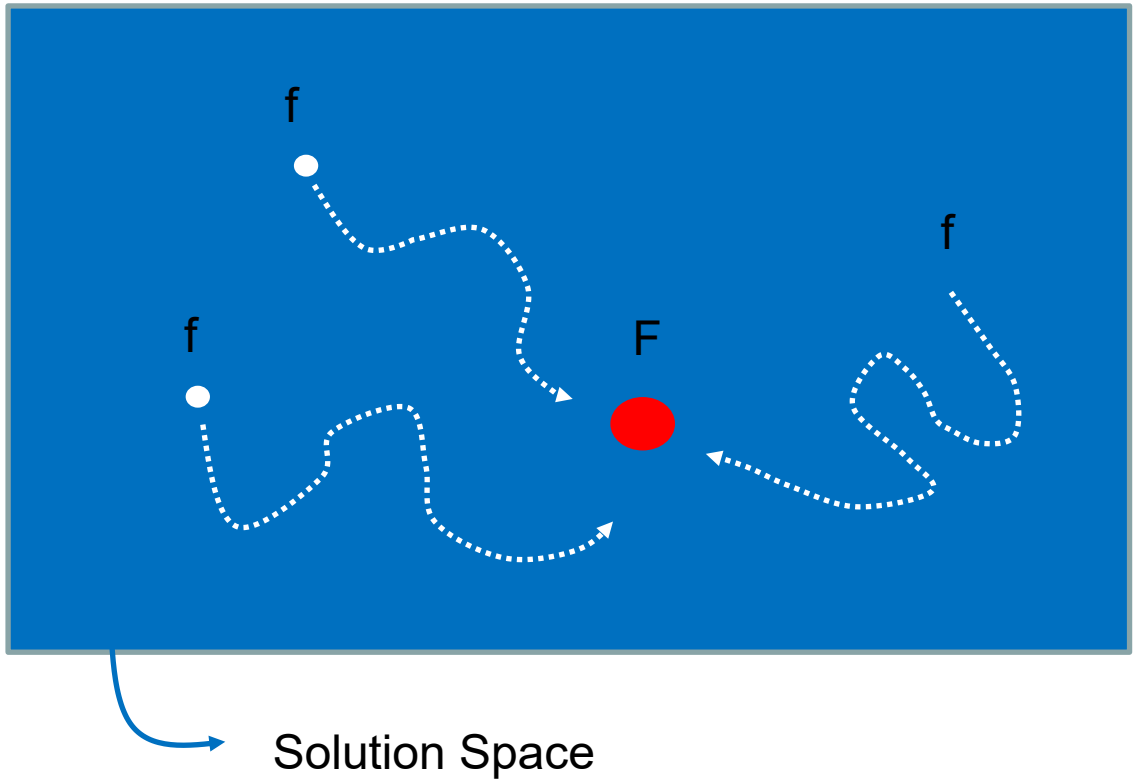
KNN

....

(Y)

y_1
y_2
y_3
...
y_n

Models as functions



f: models

F: Ideal model

Example: 1D Linear regression model

- Model:

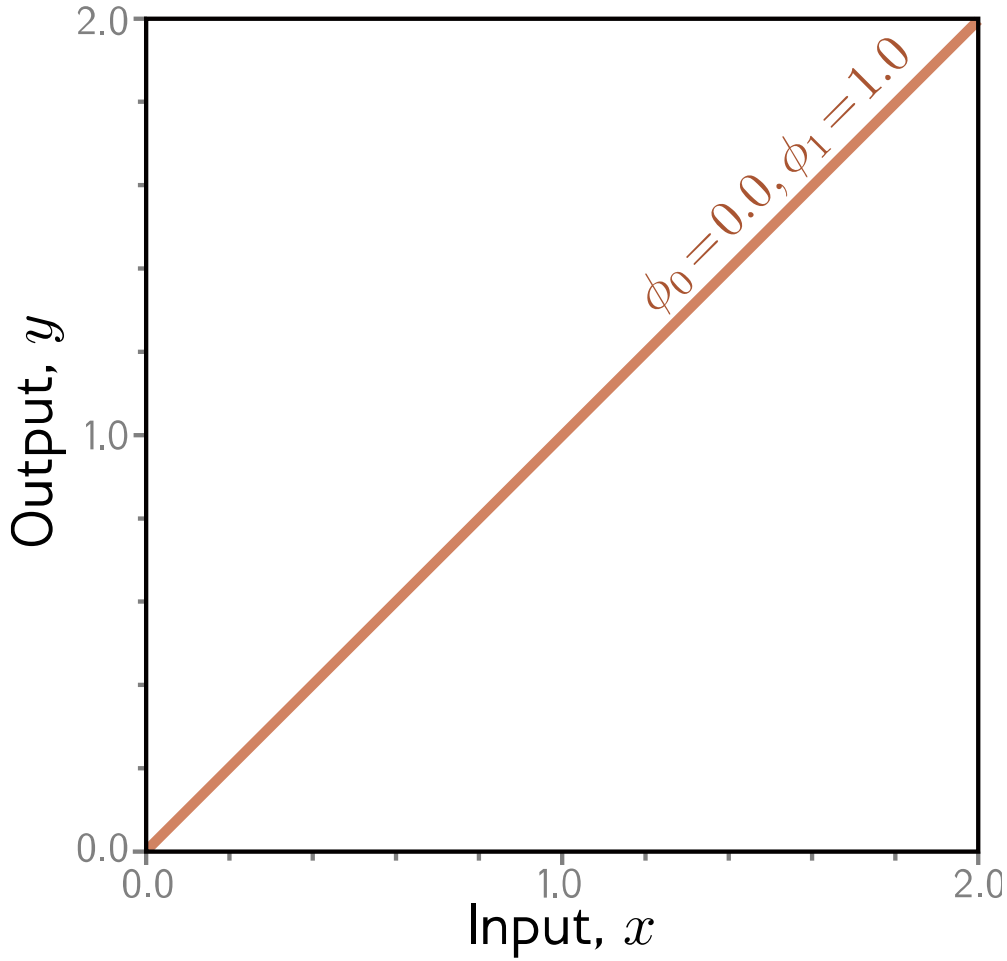
$$y = \mathrm{f}[x, \boldsymbol{\phi}]$$
$$= \phi_0 + \phi_1 x$$

- Parameters

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

← y-offset

← slope



Example: 1D Linear regression model

- Model:

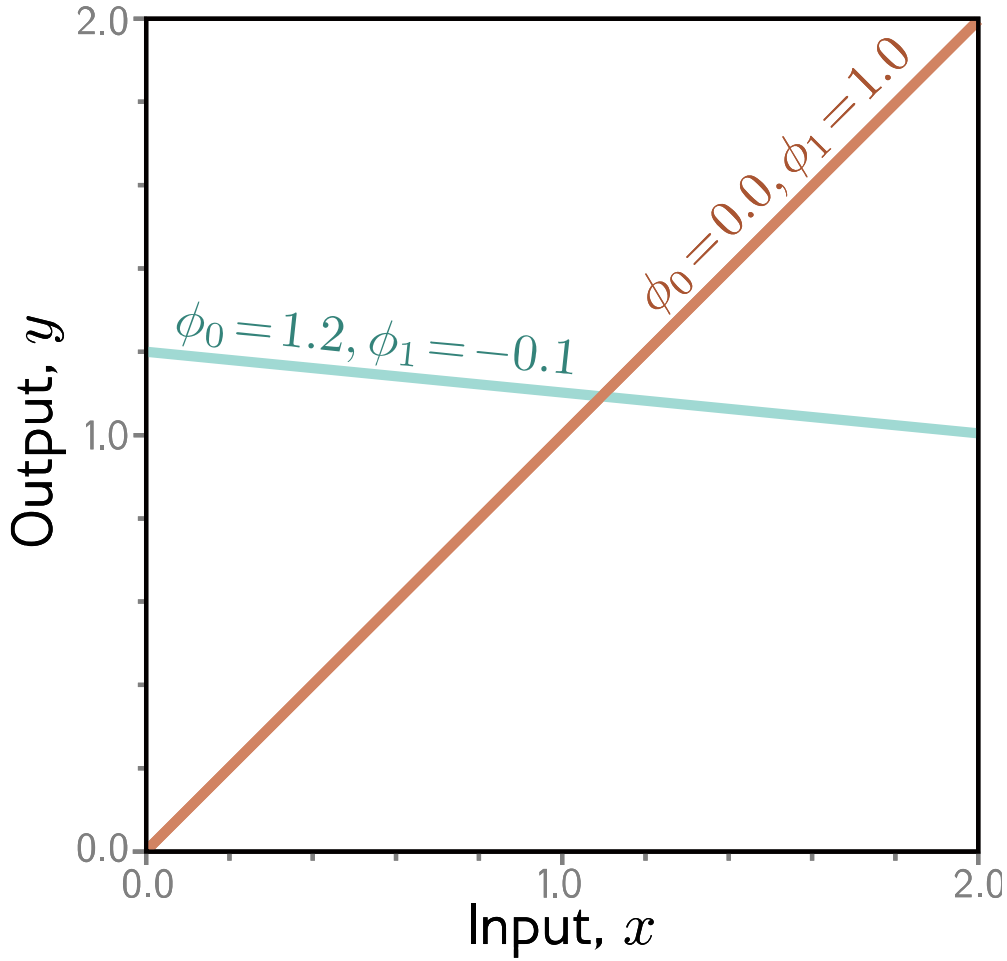
$$y = \mathrm{f}[x, \boldsymbol{\phi}]$$
$$= \phi_0 + \phi_1 x$$

- Parameters

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

← y-offset

← slope



Example: 1D Linear regression model

- Model:

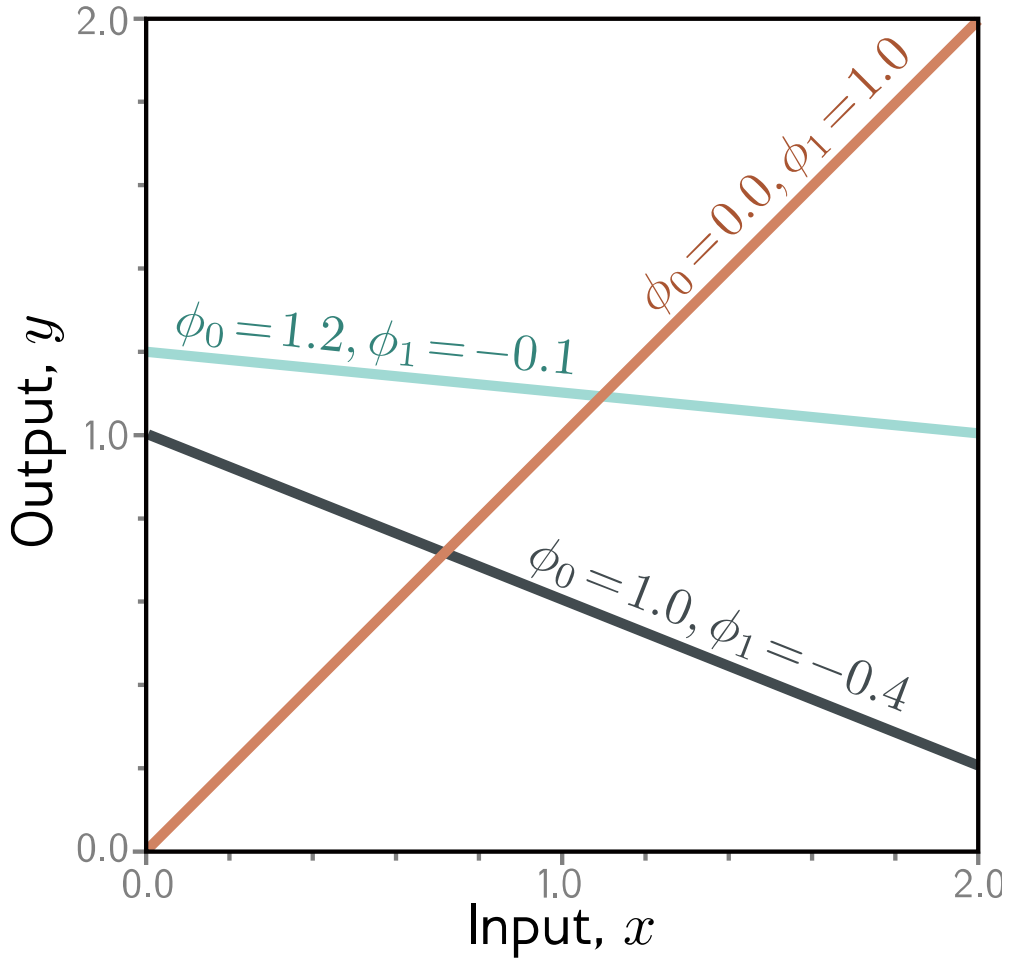
$$y = \mathrm{f}[x, \boldsymbol{\phi}]$$
$$= \phi_0 + \phi_1 x$$

- Parameters

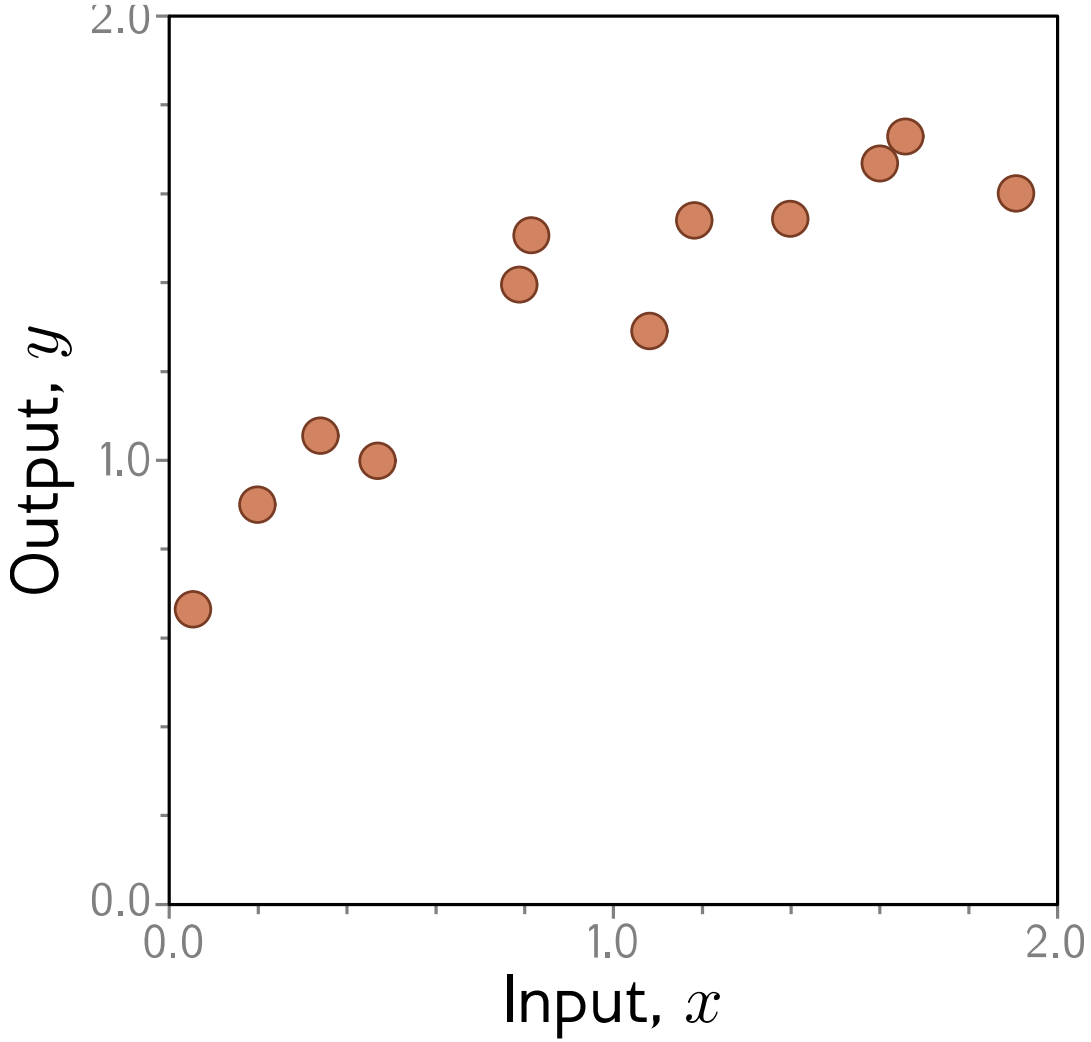
$$\boldsymbol{\phi} = \begin{bmatrix} \phi_0 \\ \phi_1 \end{bmatrix}$$

← y-offset

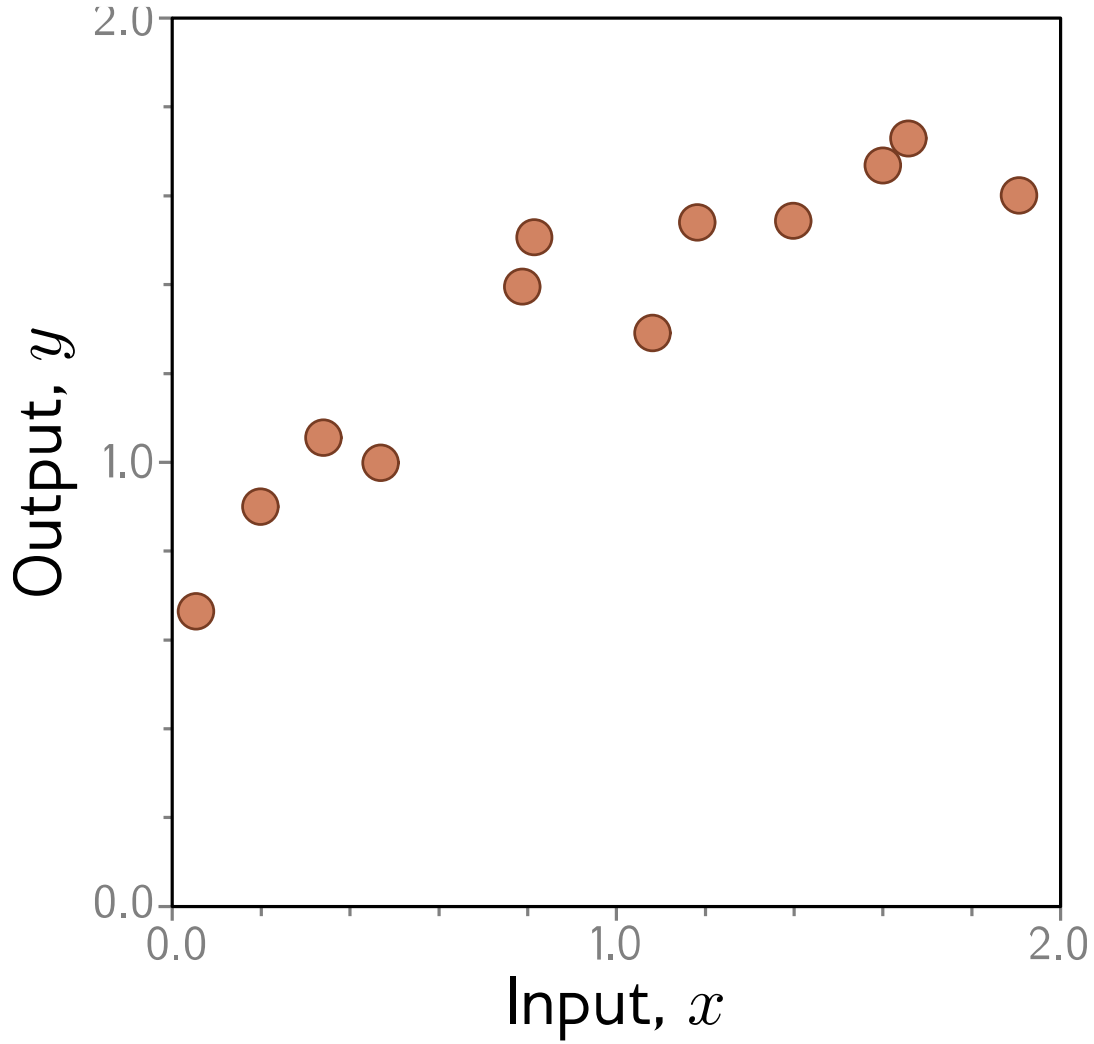
← slope



Example: 1D Linear regression training data



Example: 1D Linear regression training data



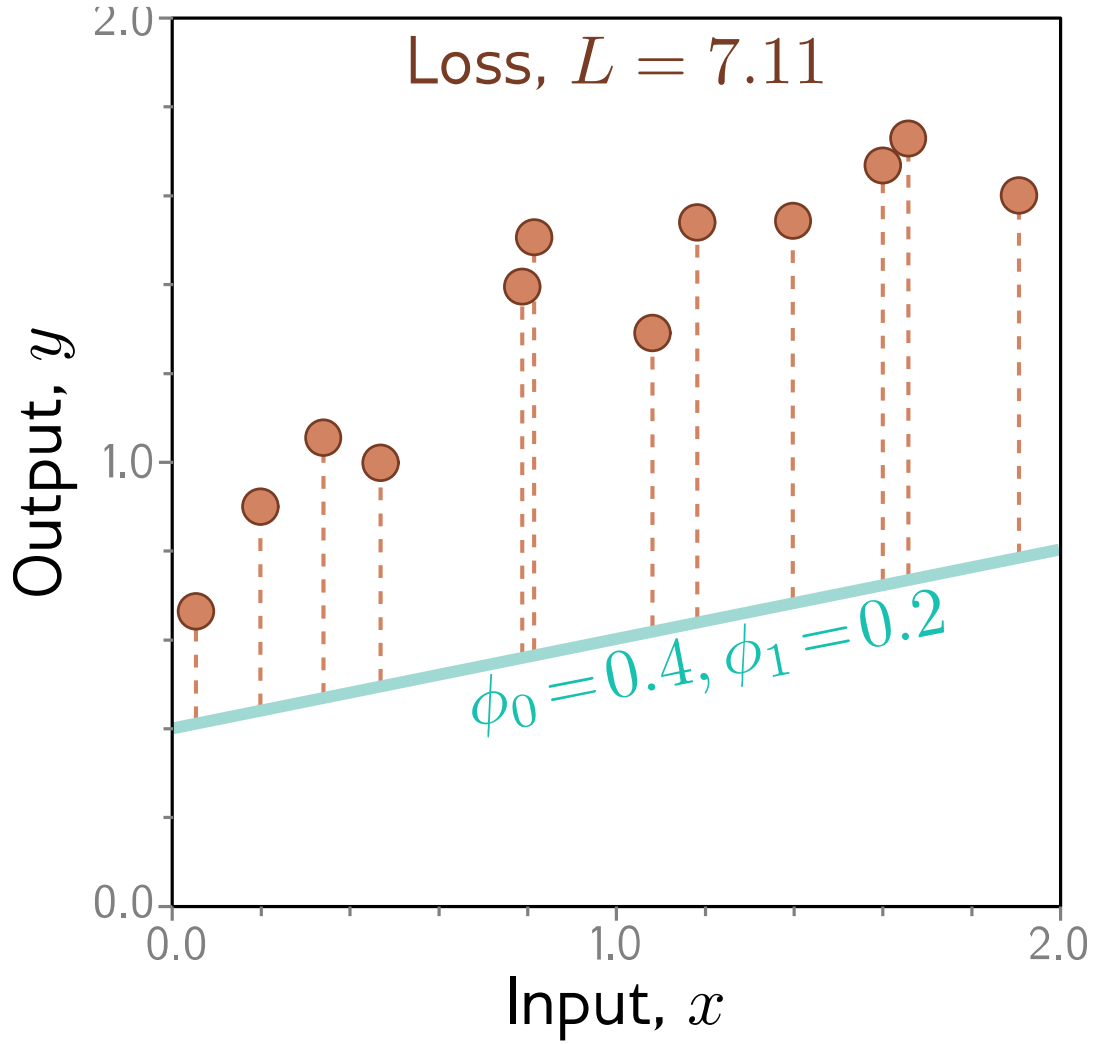
Loss function:

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

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

"Least squares loss function"

Example: 1D Linear regression loss function



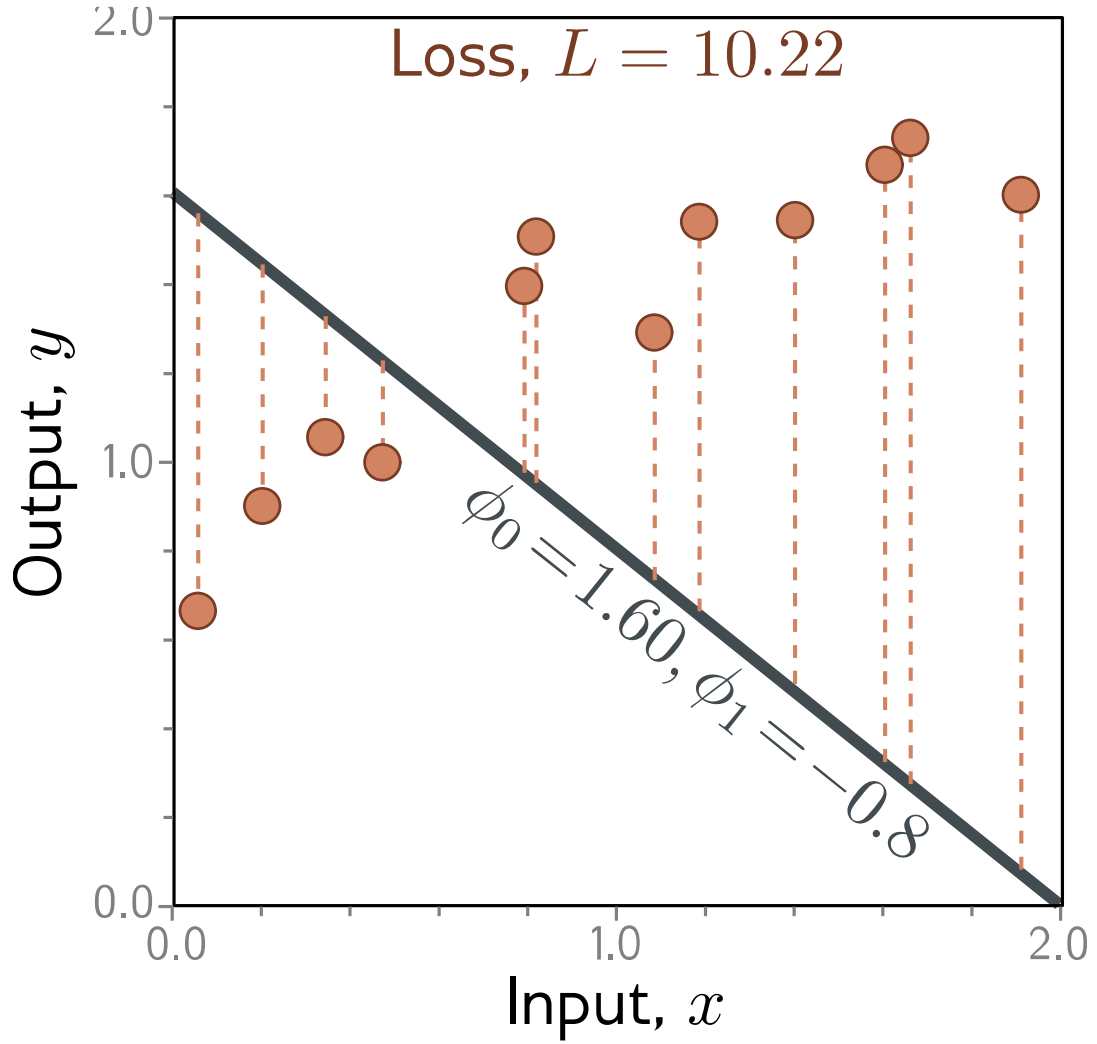
Loss function:

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

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

"Least squares loss function"

Example: 1D Linear regression loss function



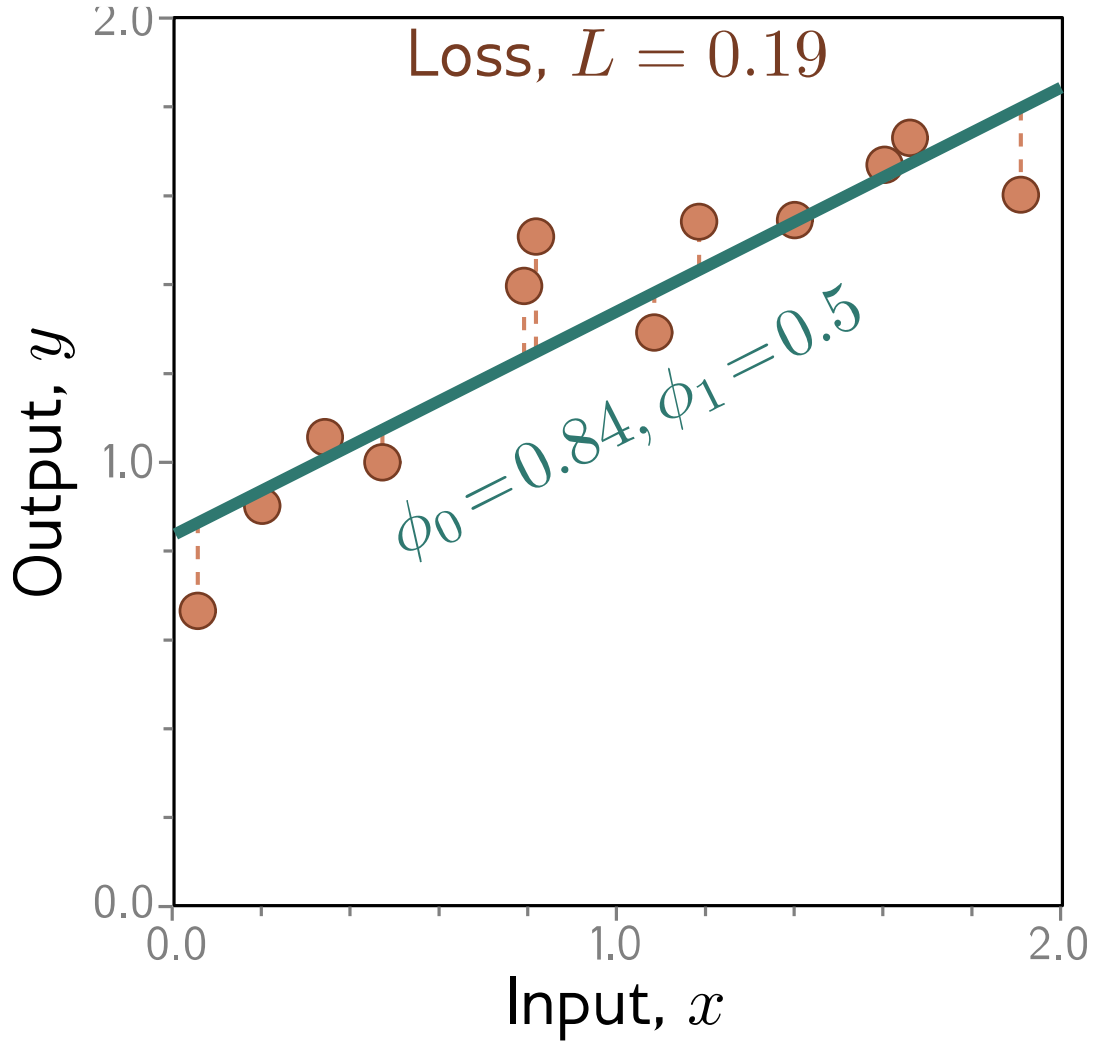
Loss function:

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

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

"Least squares loss function"

Example: 1D Linear regression loss function



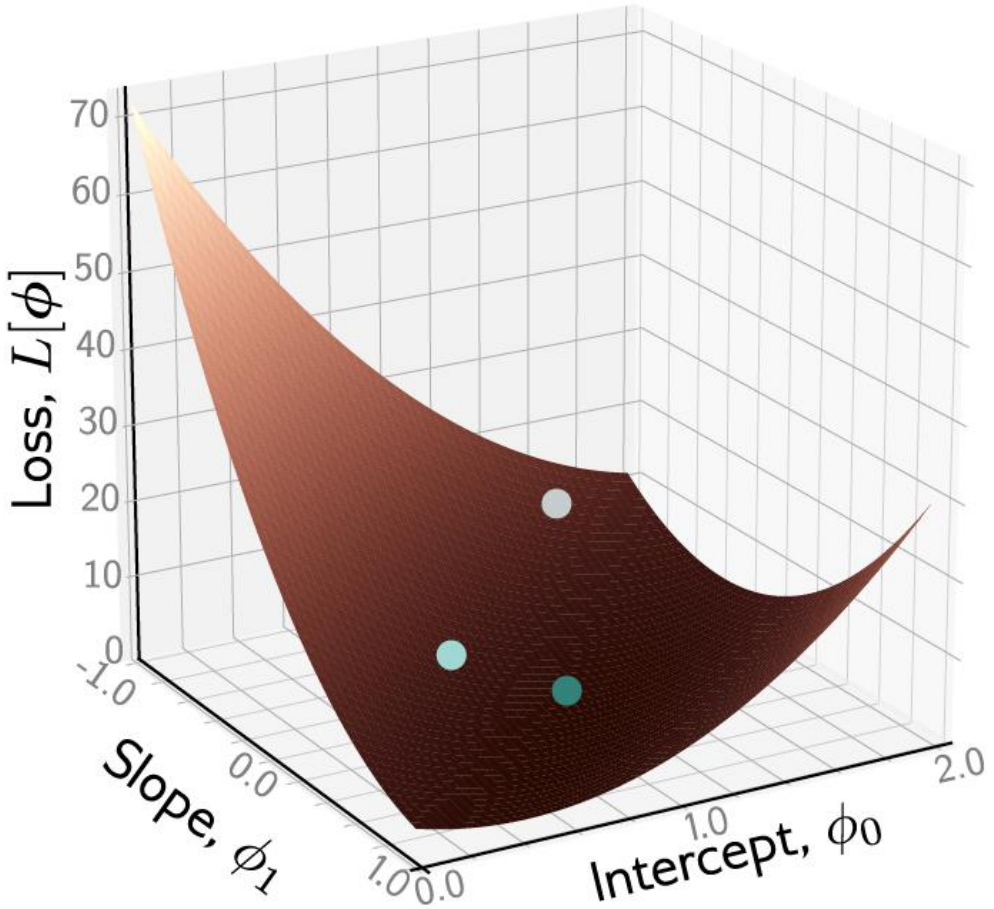
Loss function:

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

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

"Least squares loss function"

Example: 1D Linear regression loss function



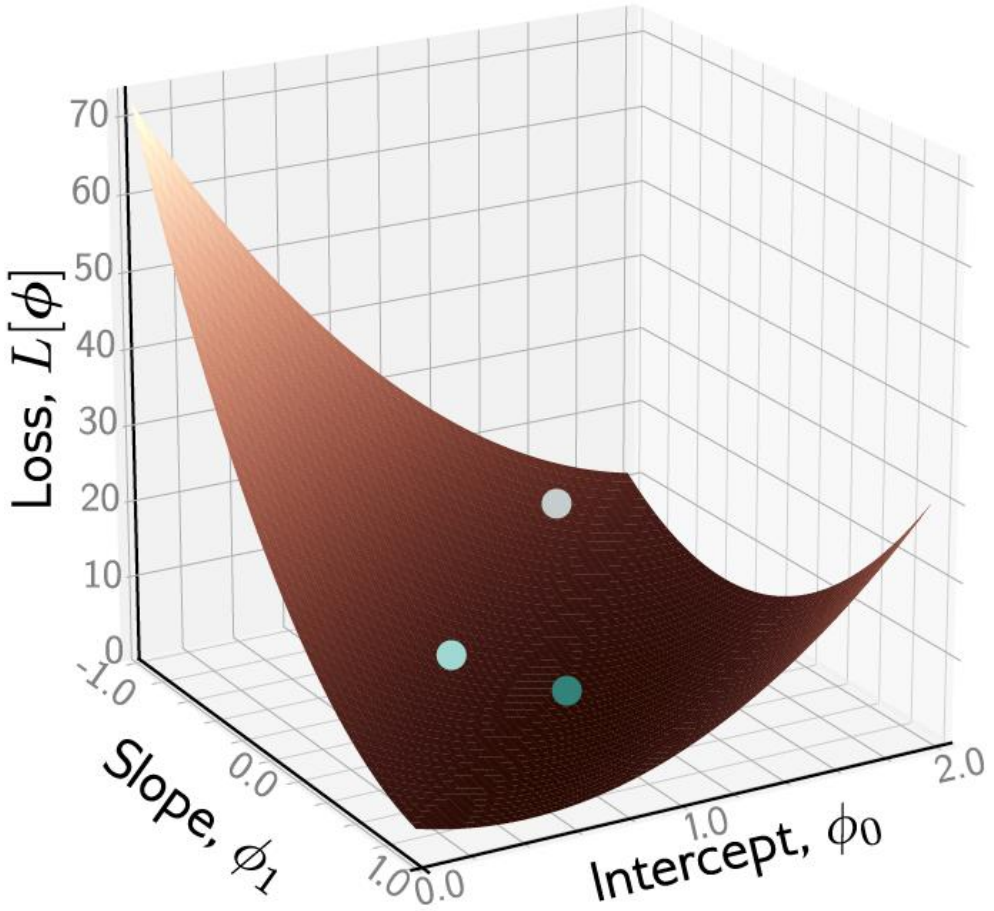
Loss function:

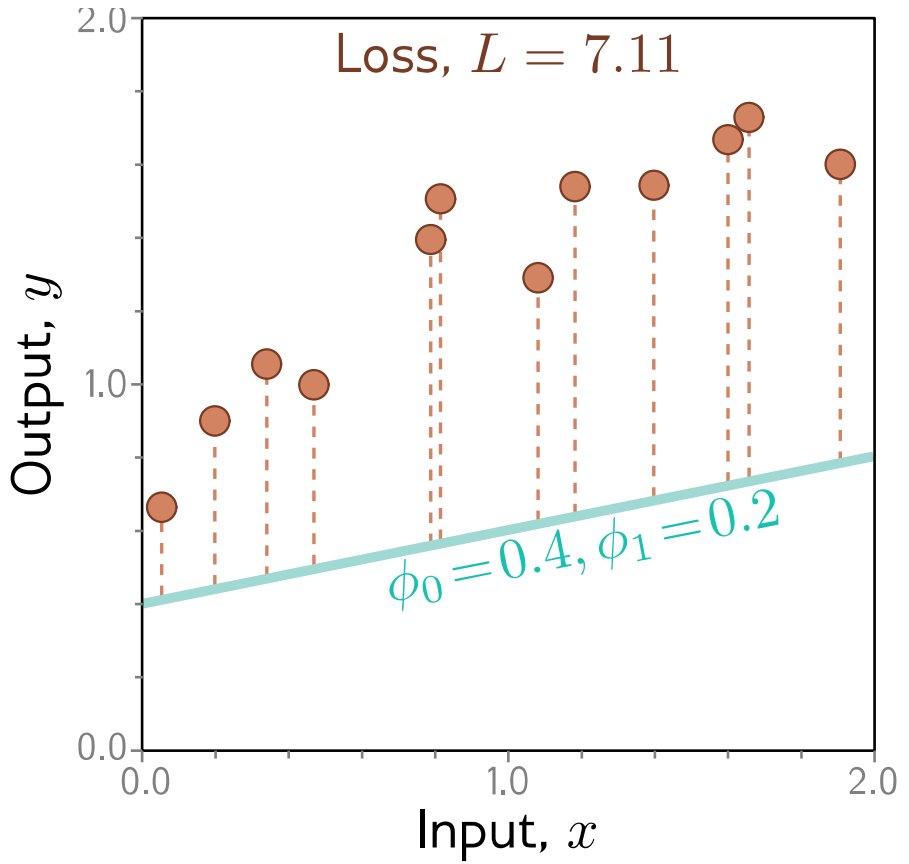
$$L[\boldsymbol{\phi}] = \sum_{i=1}^{I} (\mathrm{f}[x_i, \boldsymbol{\phi}] - y_i)^2$$

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

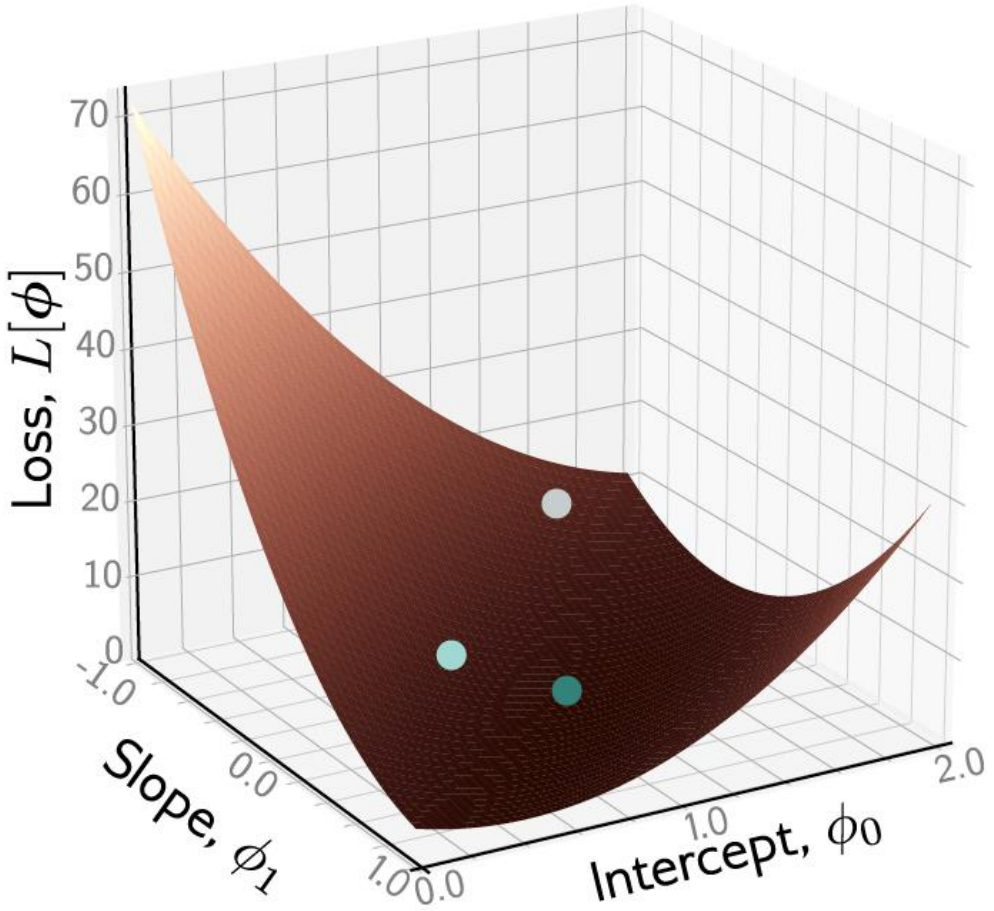
"Least squares loss function"

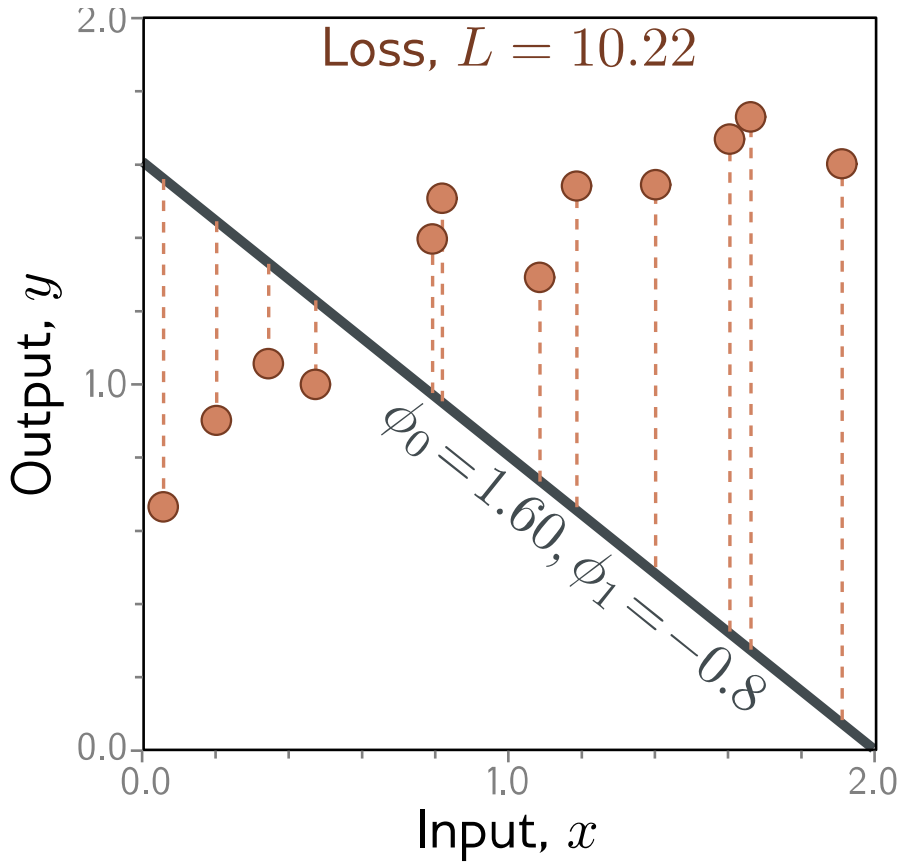
Example: 1D Linear regression loss function



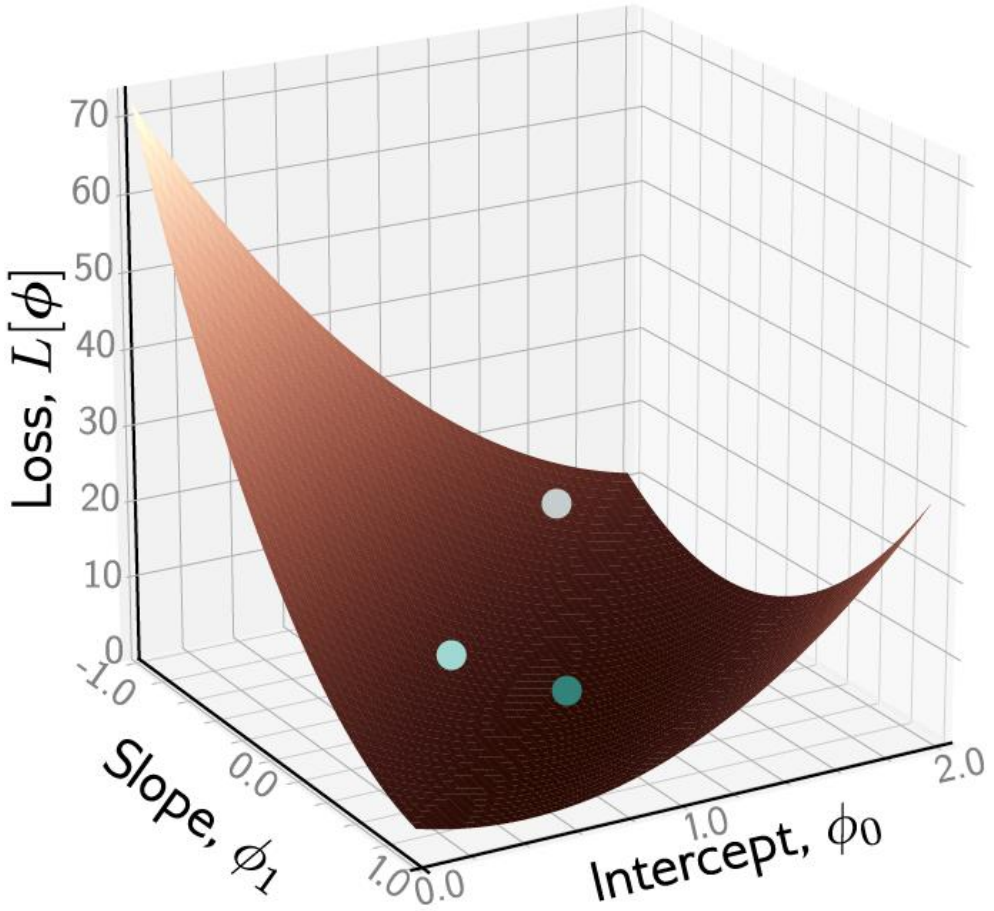


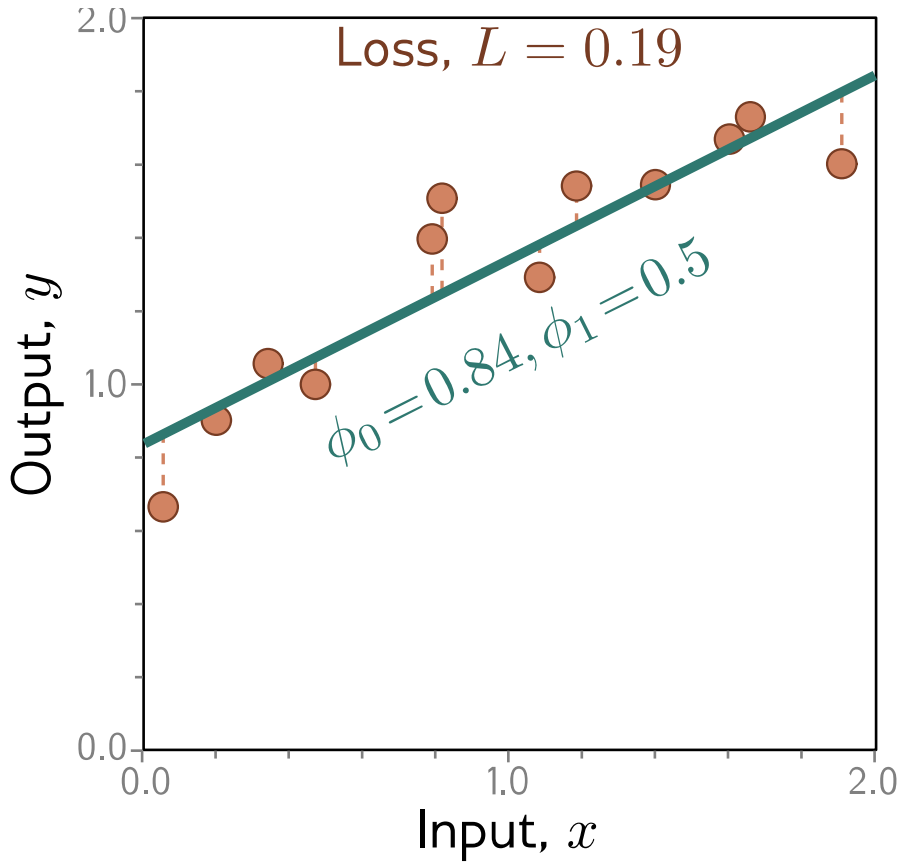
Example: 1D Linear regression loss function



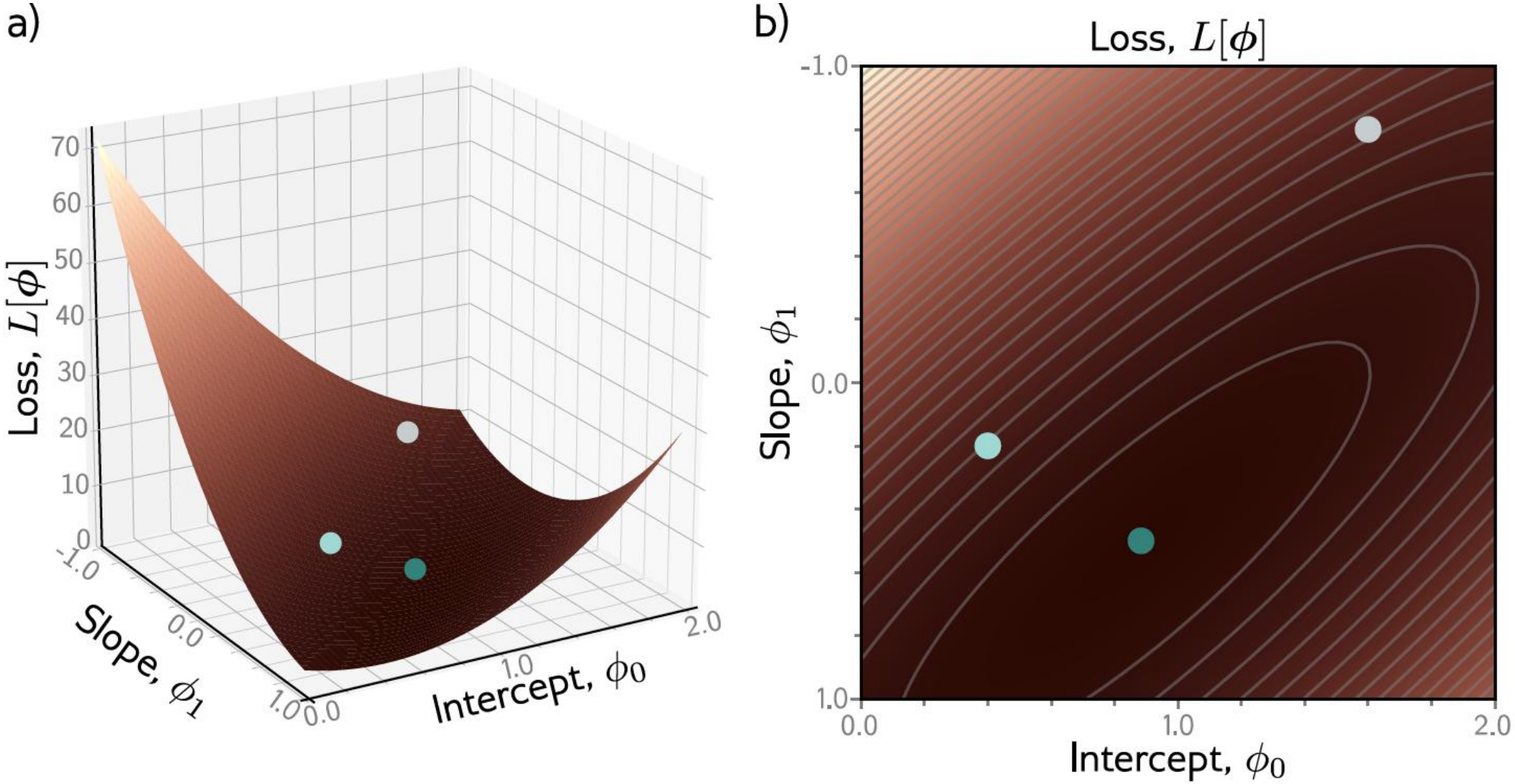


Example: 1D Linear regression loss function

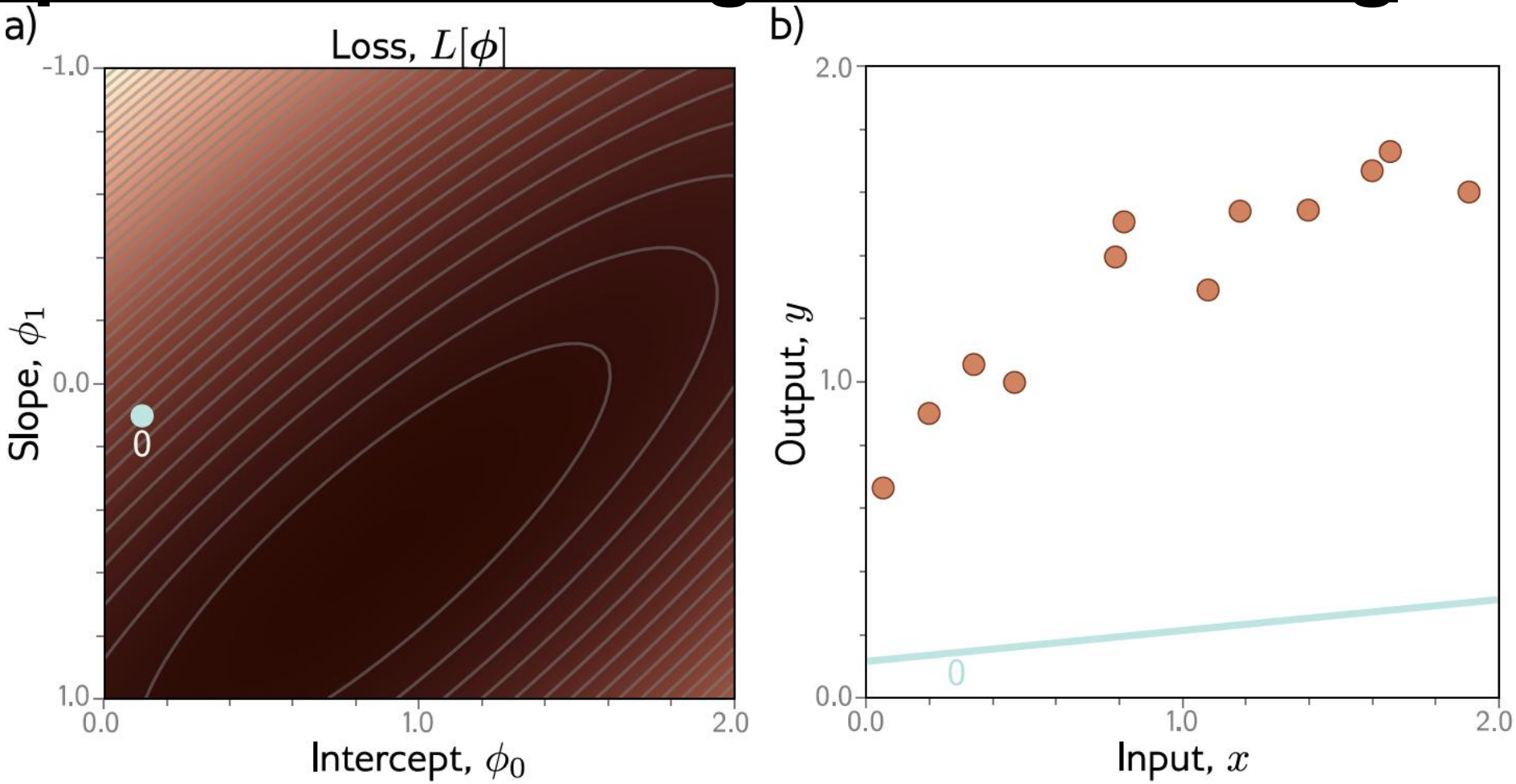




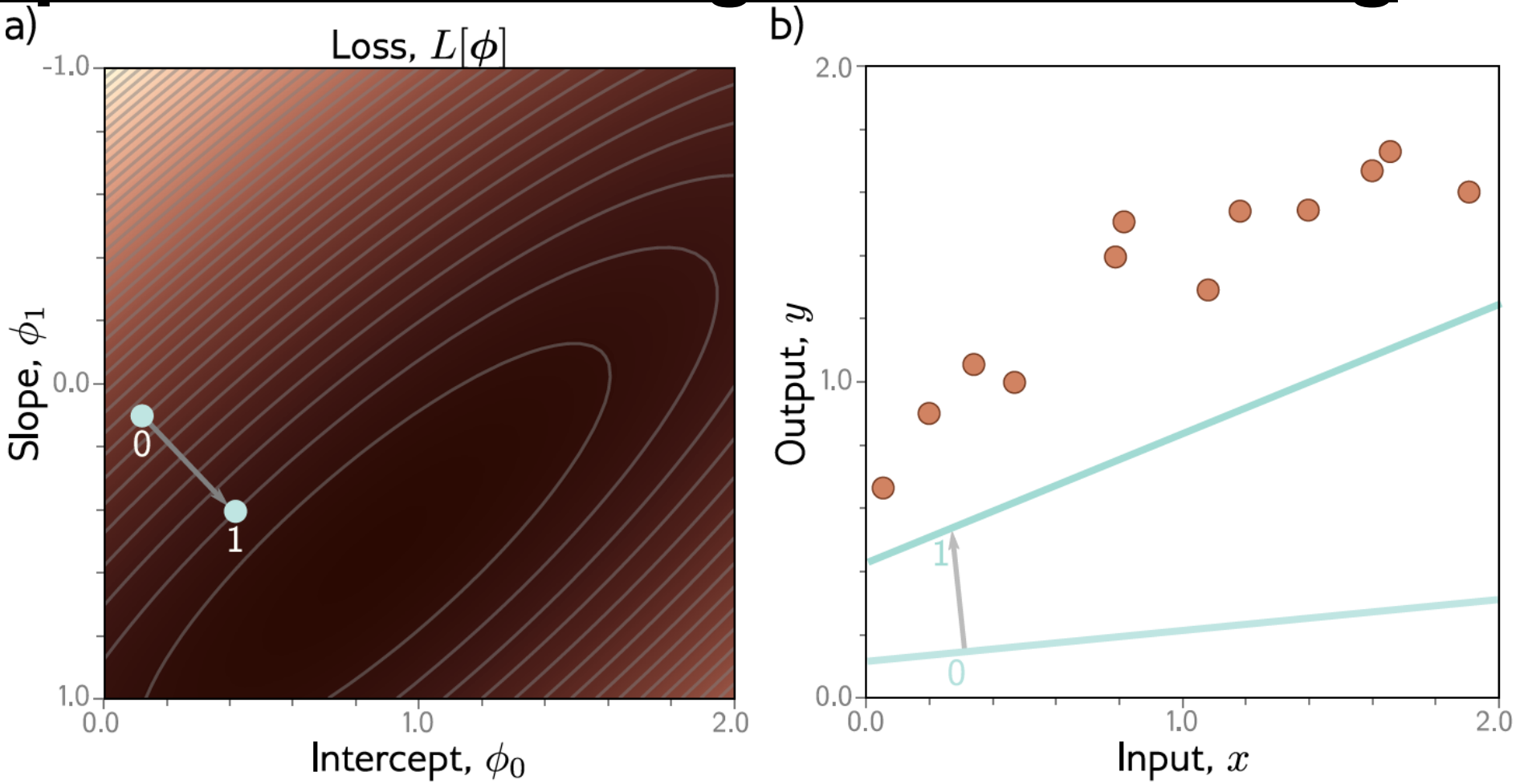
Example: 1D Linear regression loss function



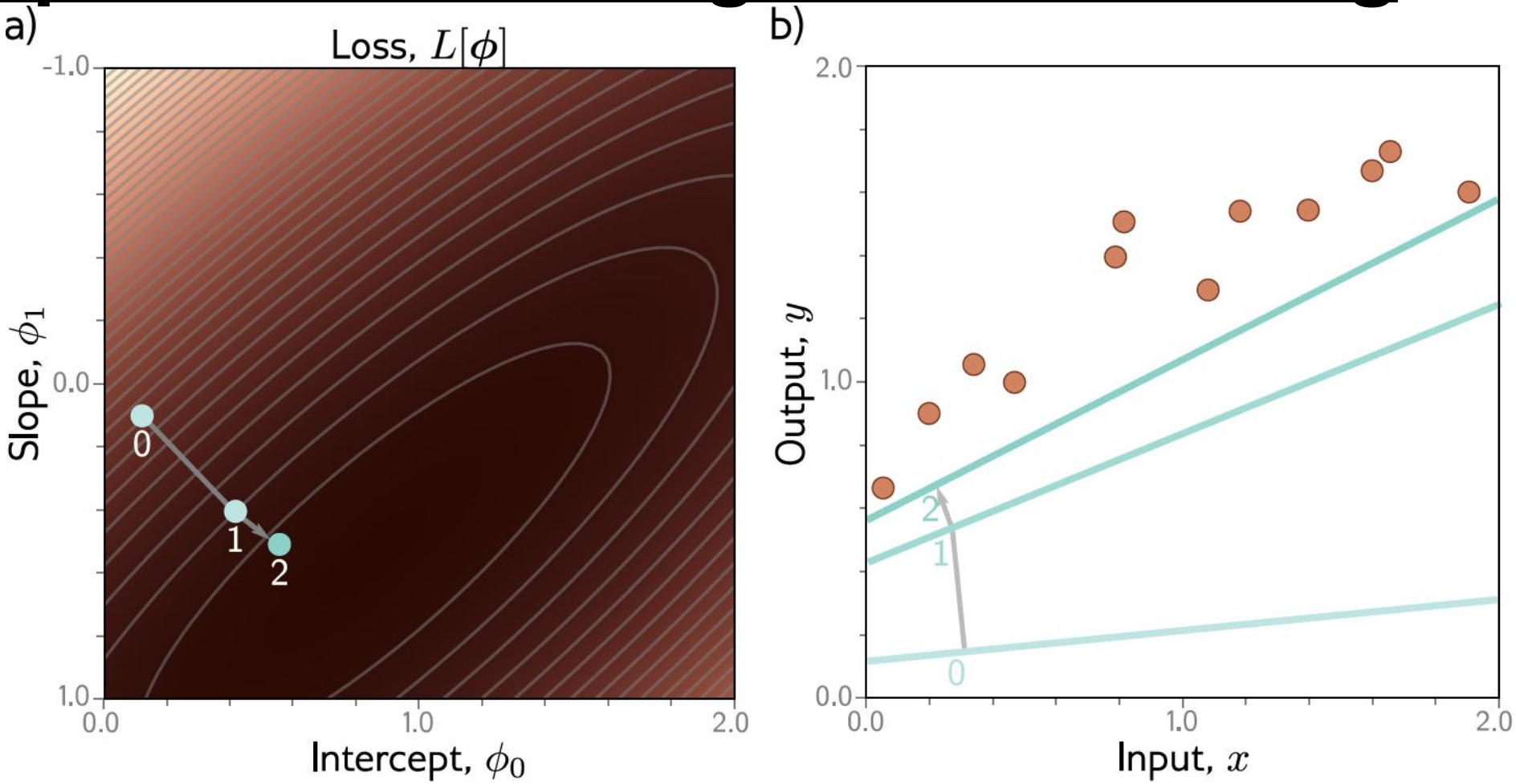
Example: 1D Linear regression training



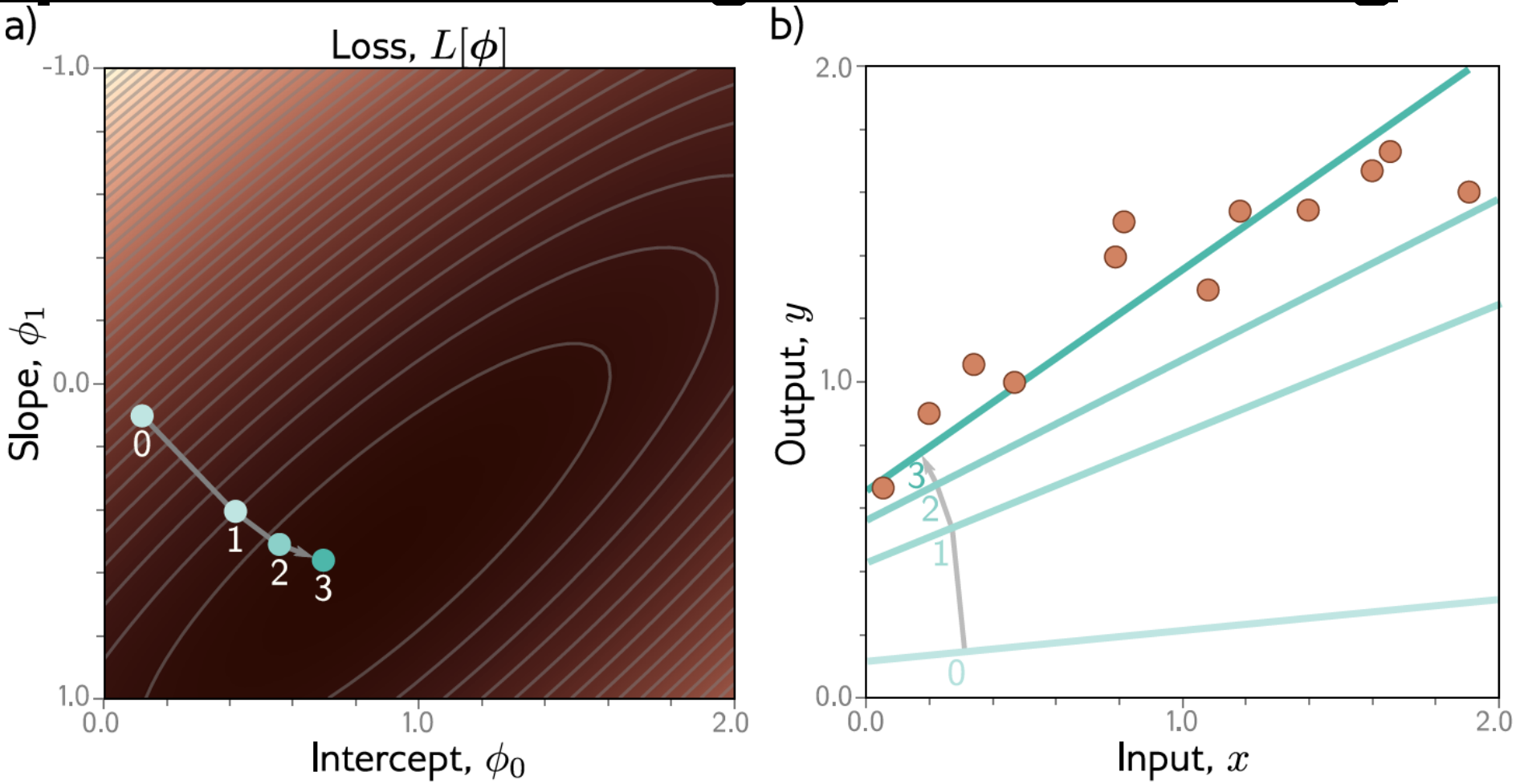
Example: 1D Linear regression training



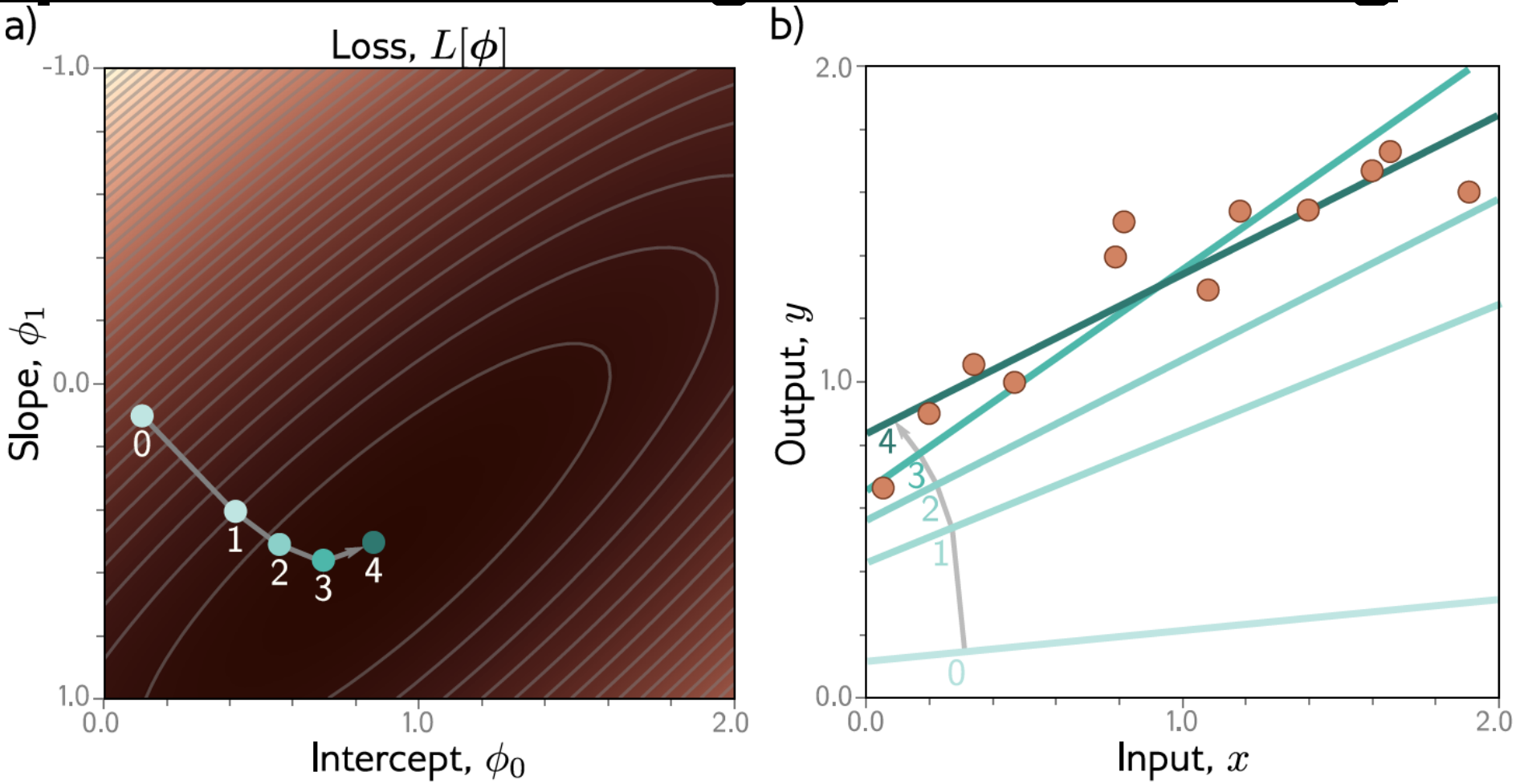
Example: 1D Linear regression training



Example: 1D Linear regression training



Example: 1D Linear regression training



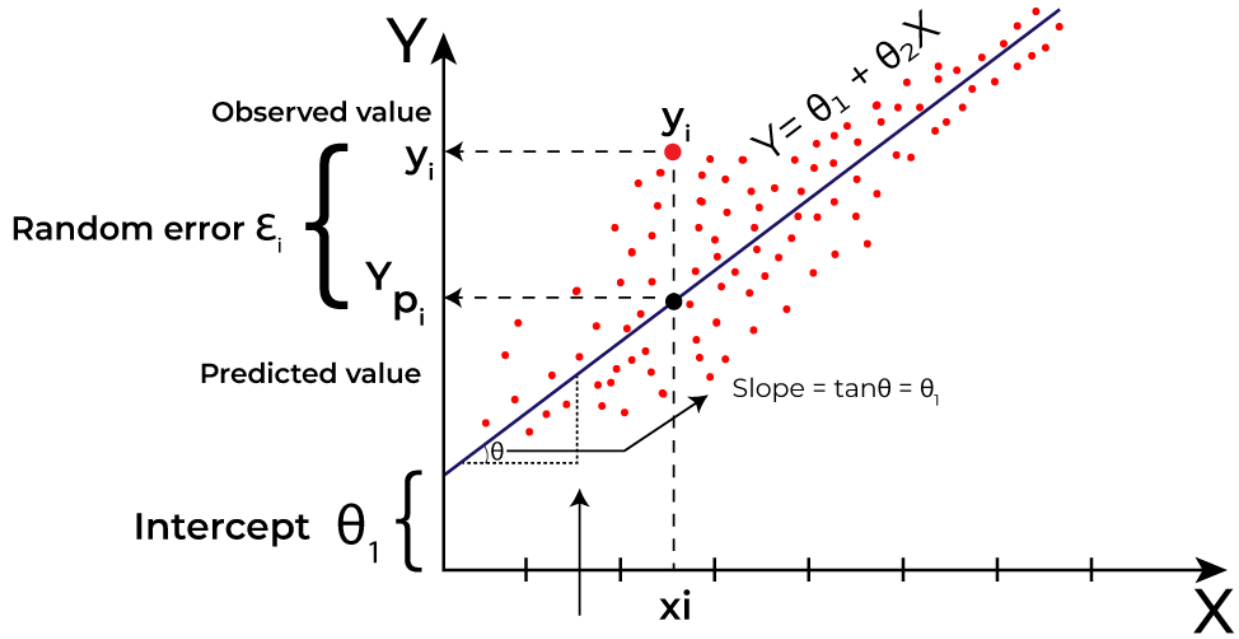
This technique is known as gradient descent

Most common methods

- Linear Regression
- Logistic Regression
- Decision Trees
- Random Forests
- Support Vector Machines
- Neural Networks

Most common methods

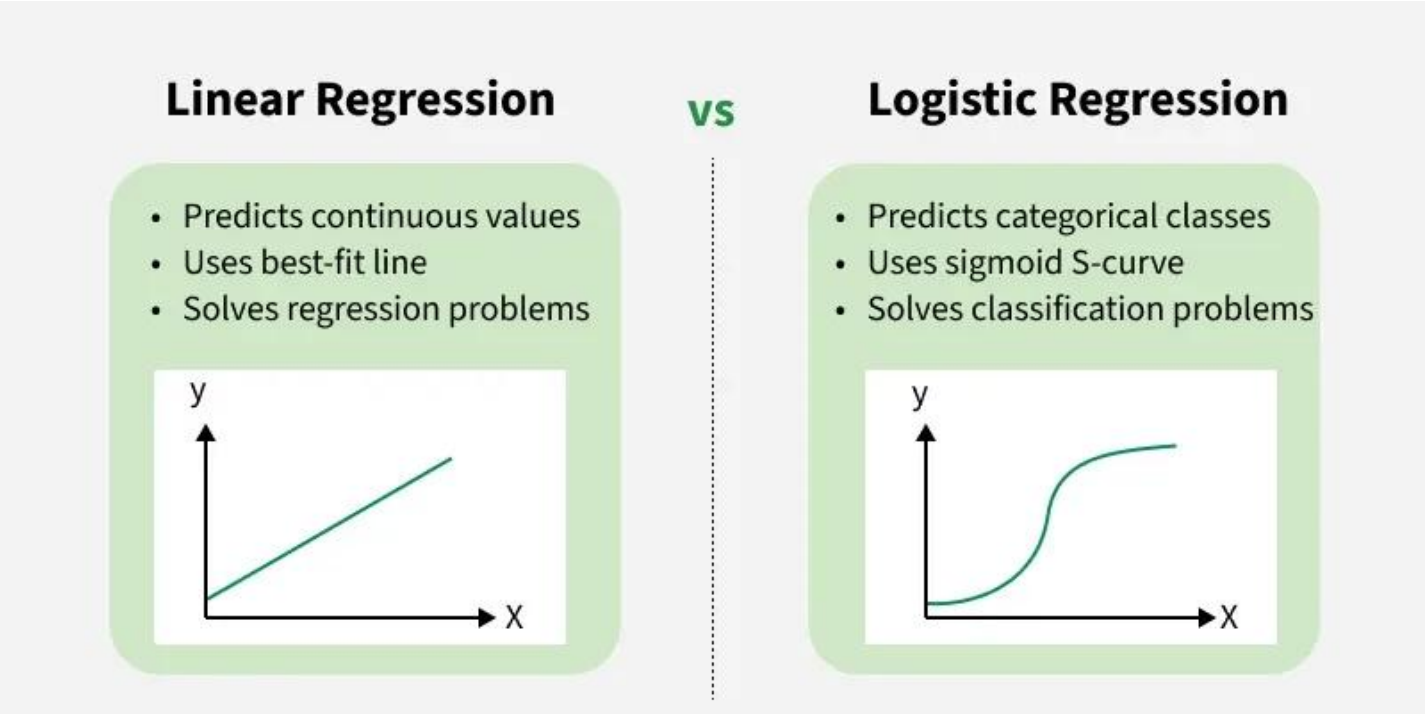
- Linear Regression
- Logistic Regression
- Decision Trees
- Random Forests
- Support Vector Machines
- Neural Networks



https://www.geeksforgeeks.org/machine-learning/ml-linear-regression/

Most common methods

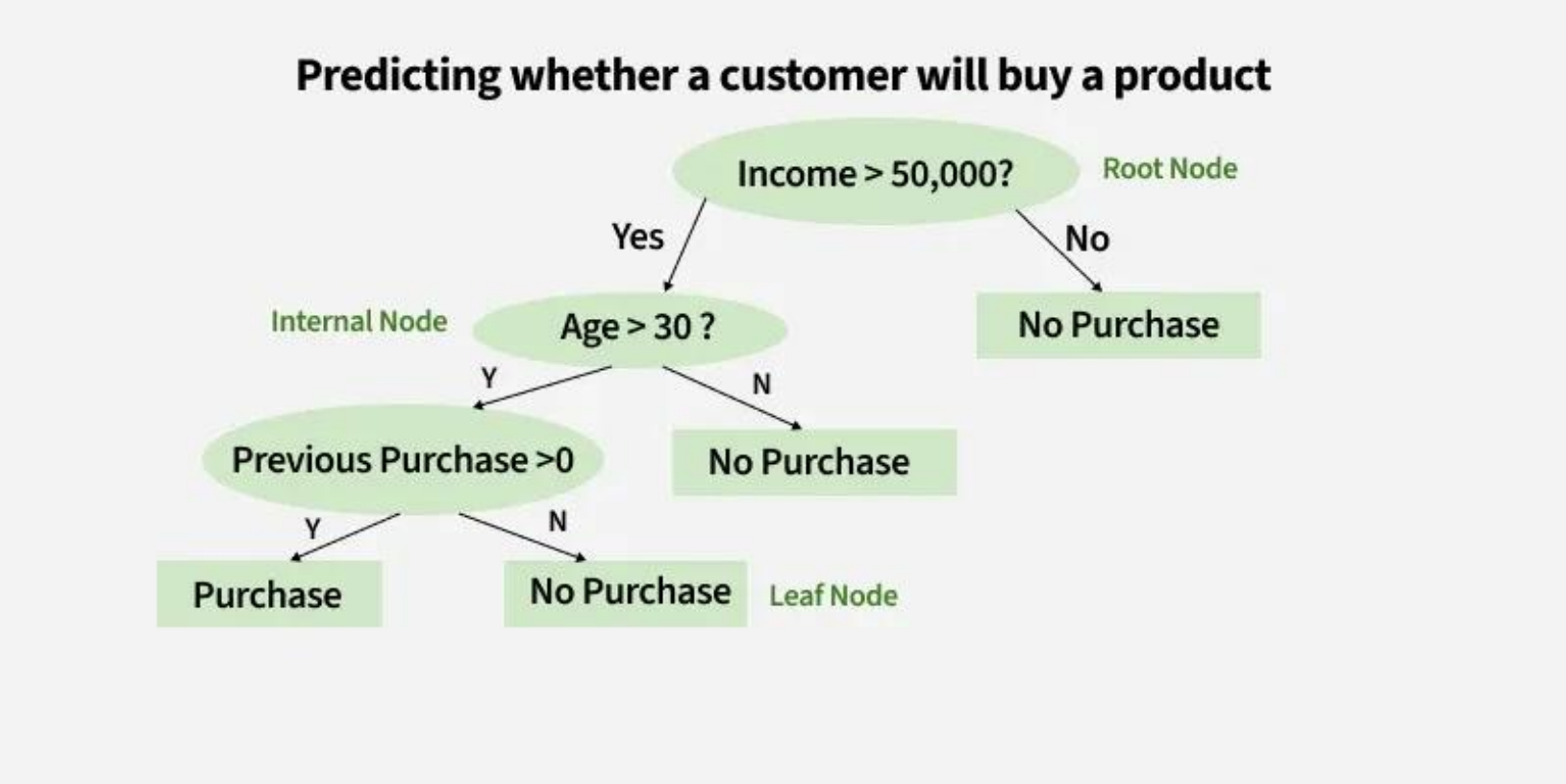
- Linear Regression
- Logistic Regression
- Decision Trees
- Random Forests
- Support Vector Machines
- Neural Networks



https://www.geeksforgeeks.org/machine-learning/understanding-logistic-regression/

Most common methods

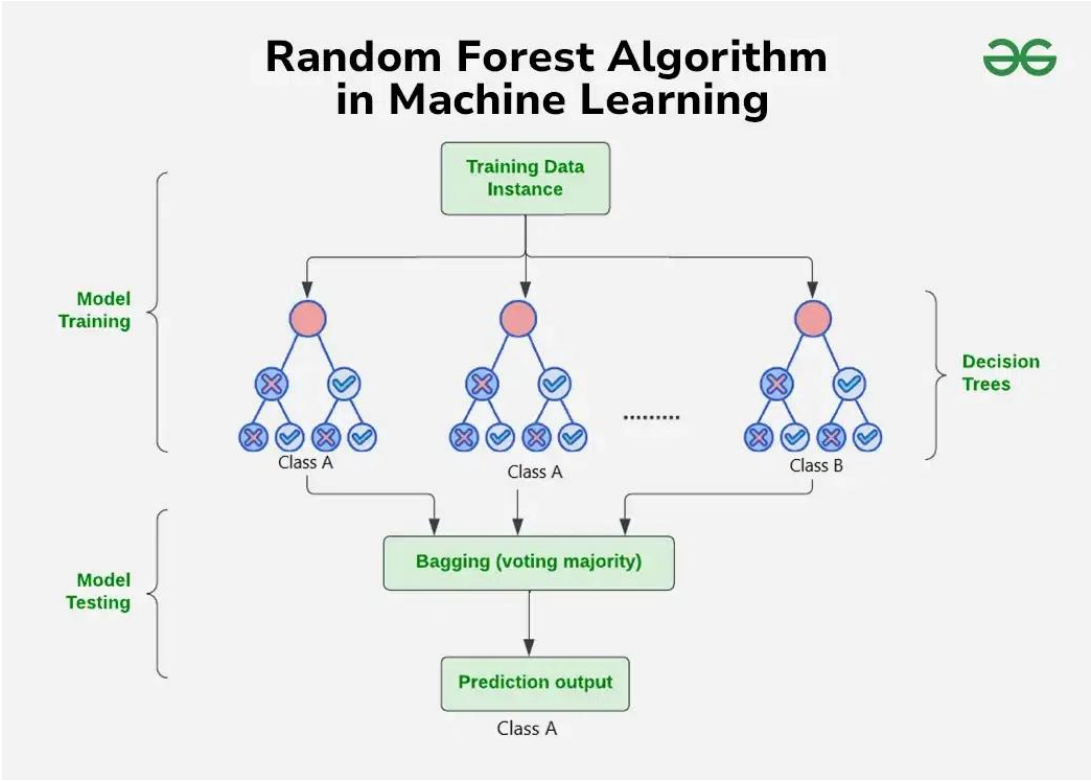
- Linear Regression
- Logistic Regression
- Decision Trees
- Random Forests
- Support Vector Mach
- Neural Networks



https://www.geeksforgeeks.org/machine-learning/decision-tree-introduction-example/

Most common methods

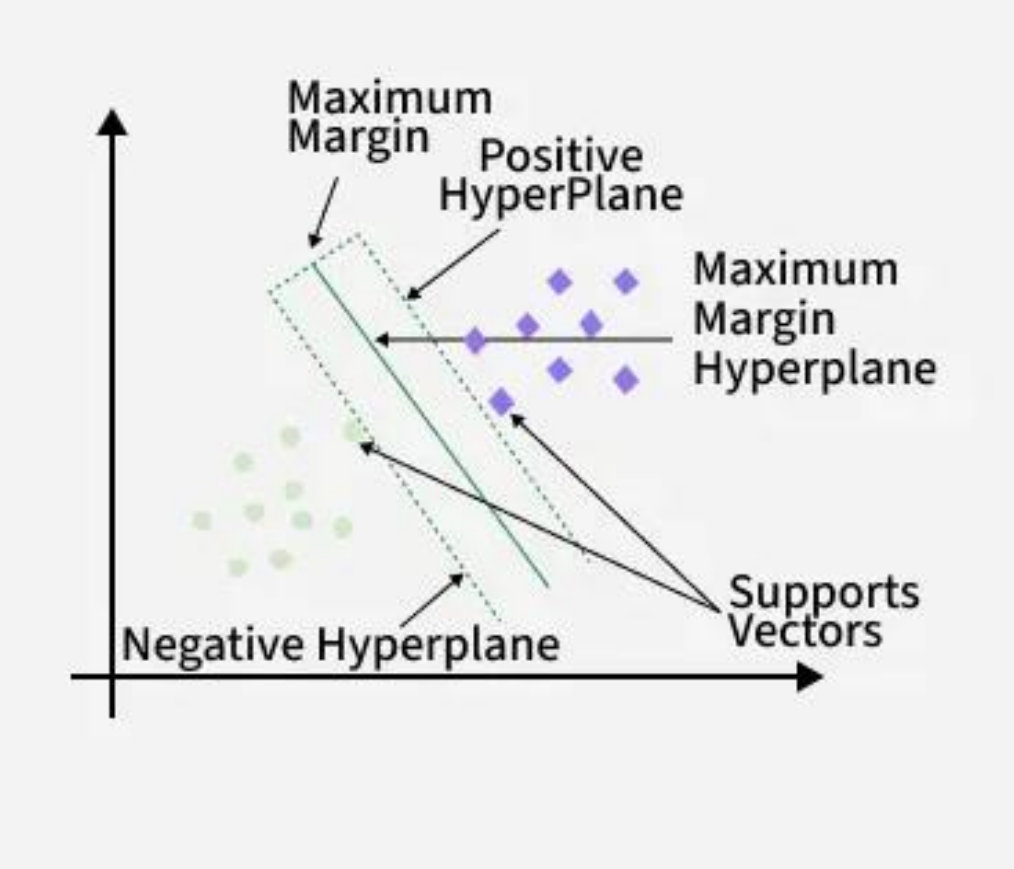
- Linear Regression
- Logistic Regression
- Decision Trees
- Random Forests
- Support Vector Machines
- Neural Networks



https://www.geeksforgeeks.org/machine-learning/random-forest-algorithm-in-machine-learning/

Most common methods

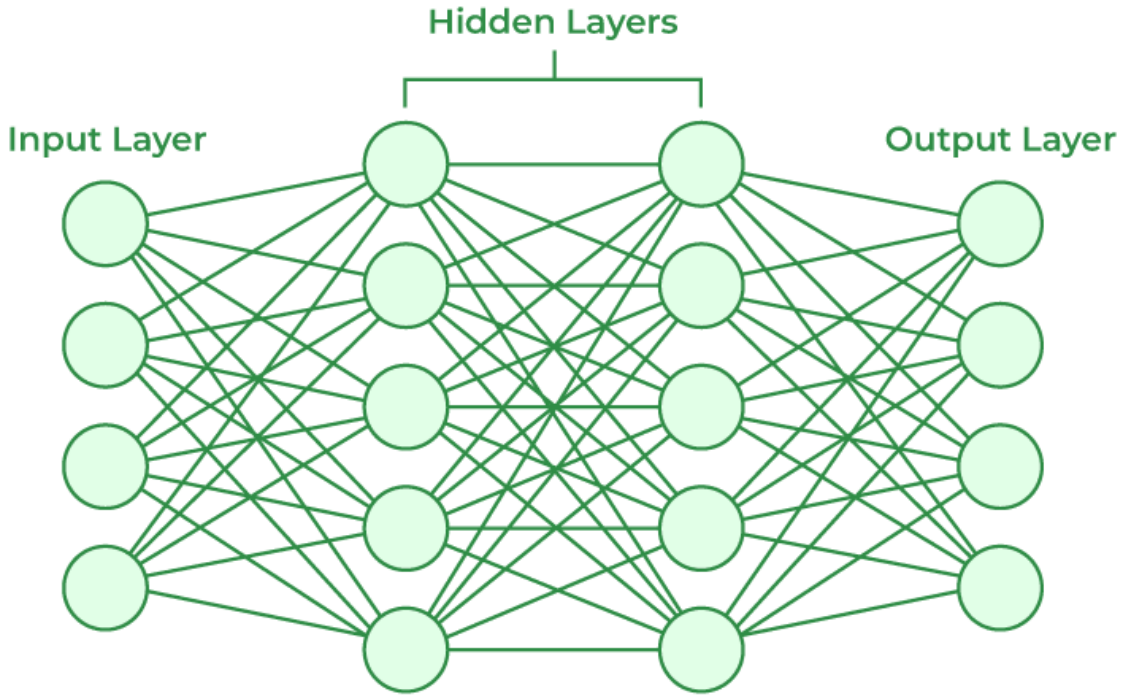
- Linear Regression
- Logistic Regression
- Decision Trees
- Random Forests
- Support Vector Machines
- Neural Networks



https://www.geeksforgeeks.org/machine-learning/support-vector-machine-algorithm/

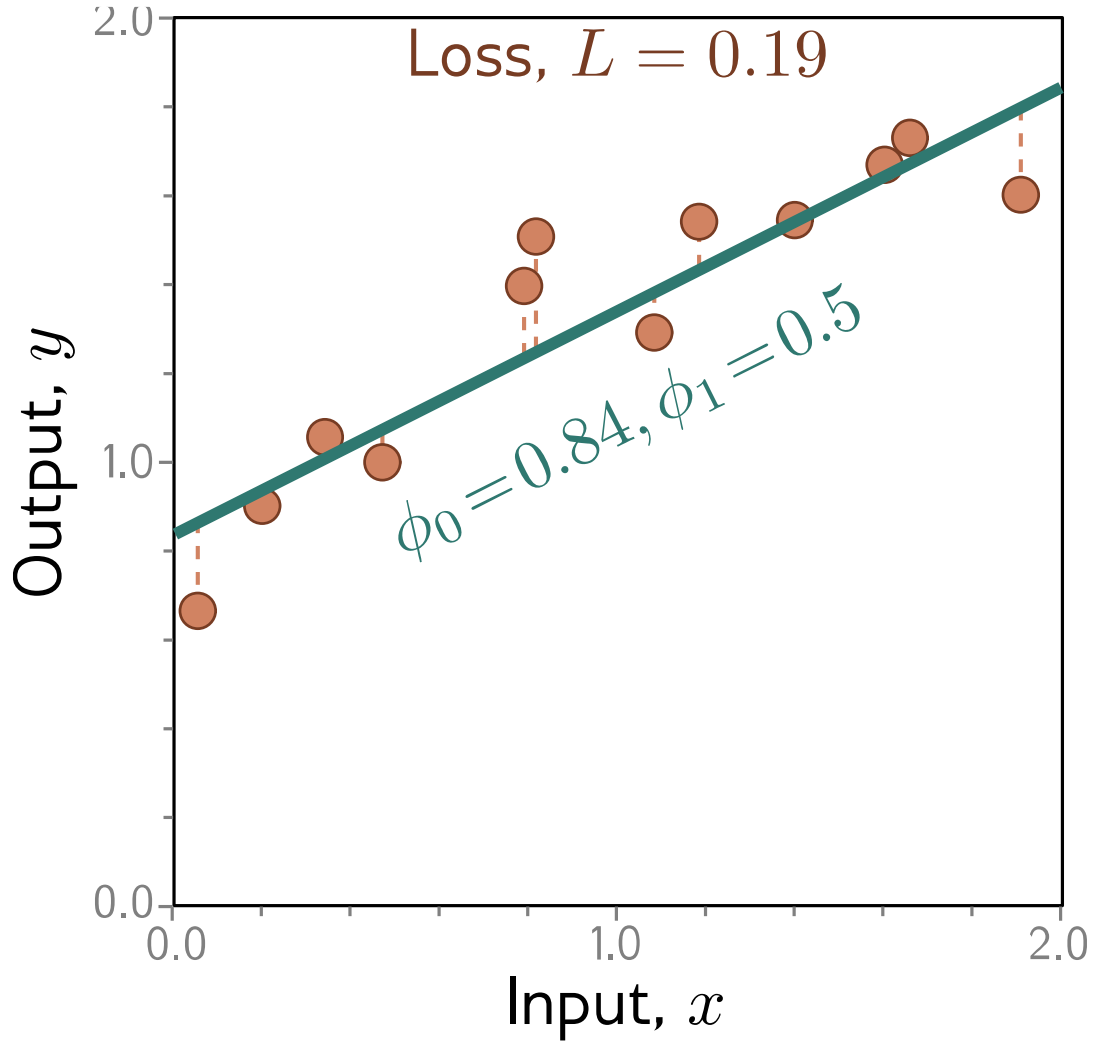
Most common methods

- Linear Regression
- Logistic Regression
- Decision Trees
- Random Forests
- Support Vector Machines
- Neural Networks



https://www.geeksforgeeks.org/artificial-intelligence/artificial-neural-networks-and-its-applications/

Example: 1D Linear regression loss function



Loss function:

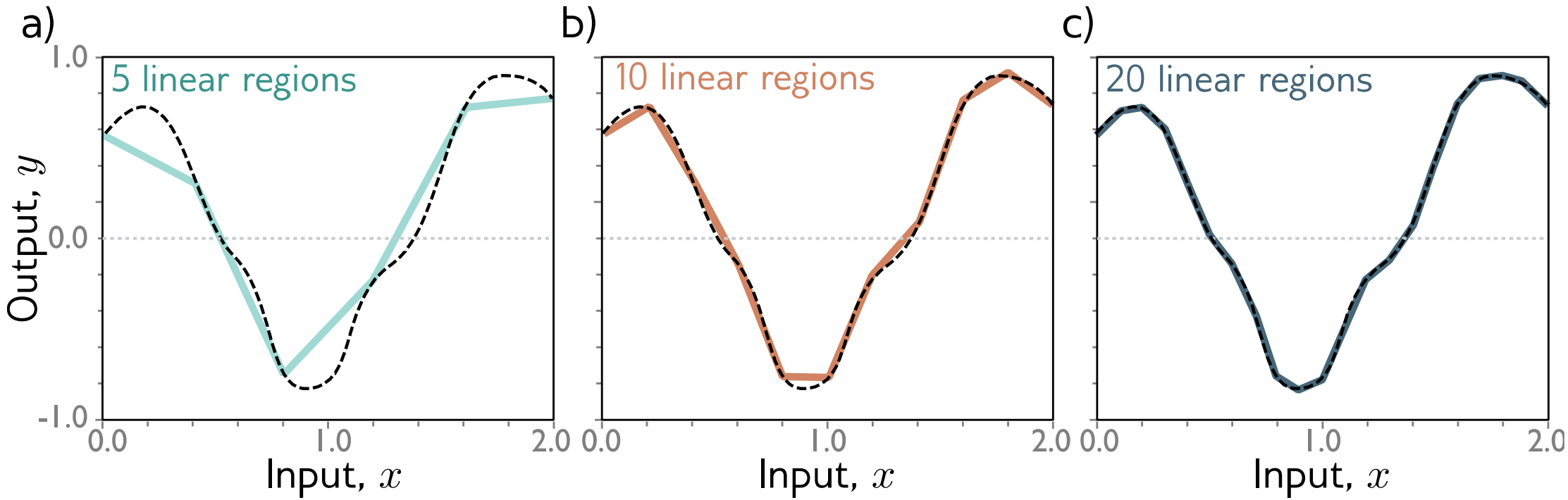
$$L[\boldsymbol{\phi}] = \sum_{i=1}^{I} (\mathrm{f}[x_i, \boldsymbol{\phi}] - y_i)^2$$

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

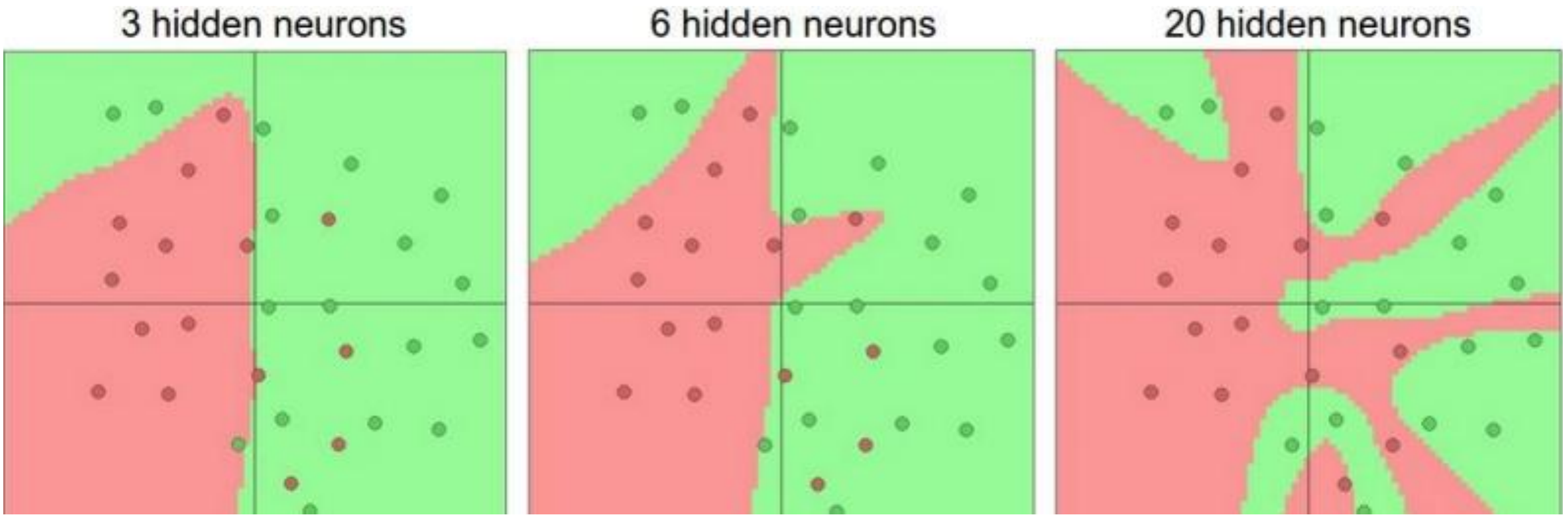
"Least squares loss function"

With enough hidden units…

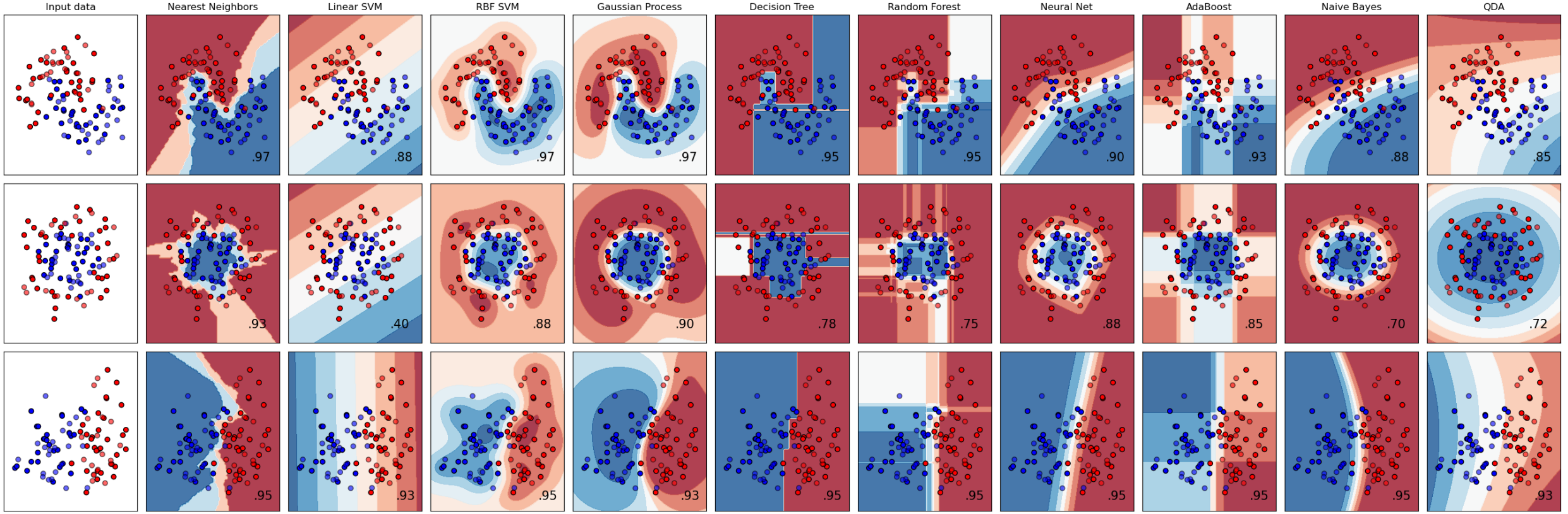
… we can describe any 1D function to arbitrary accuracy



Deep Larning

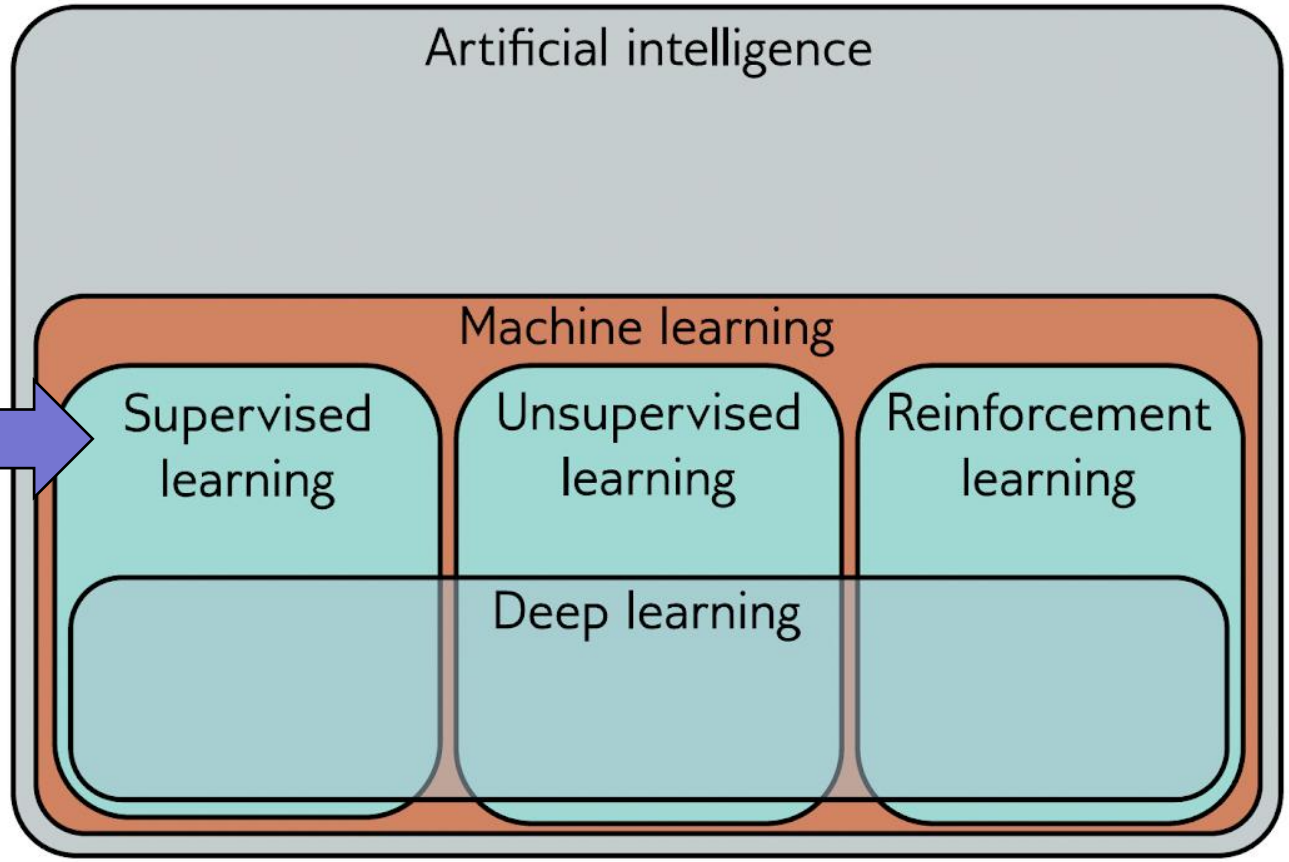


Classifiers comparison

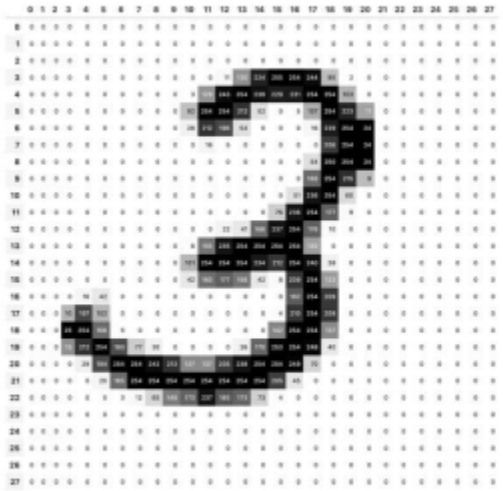


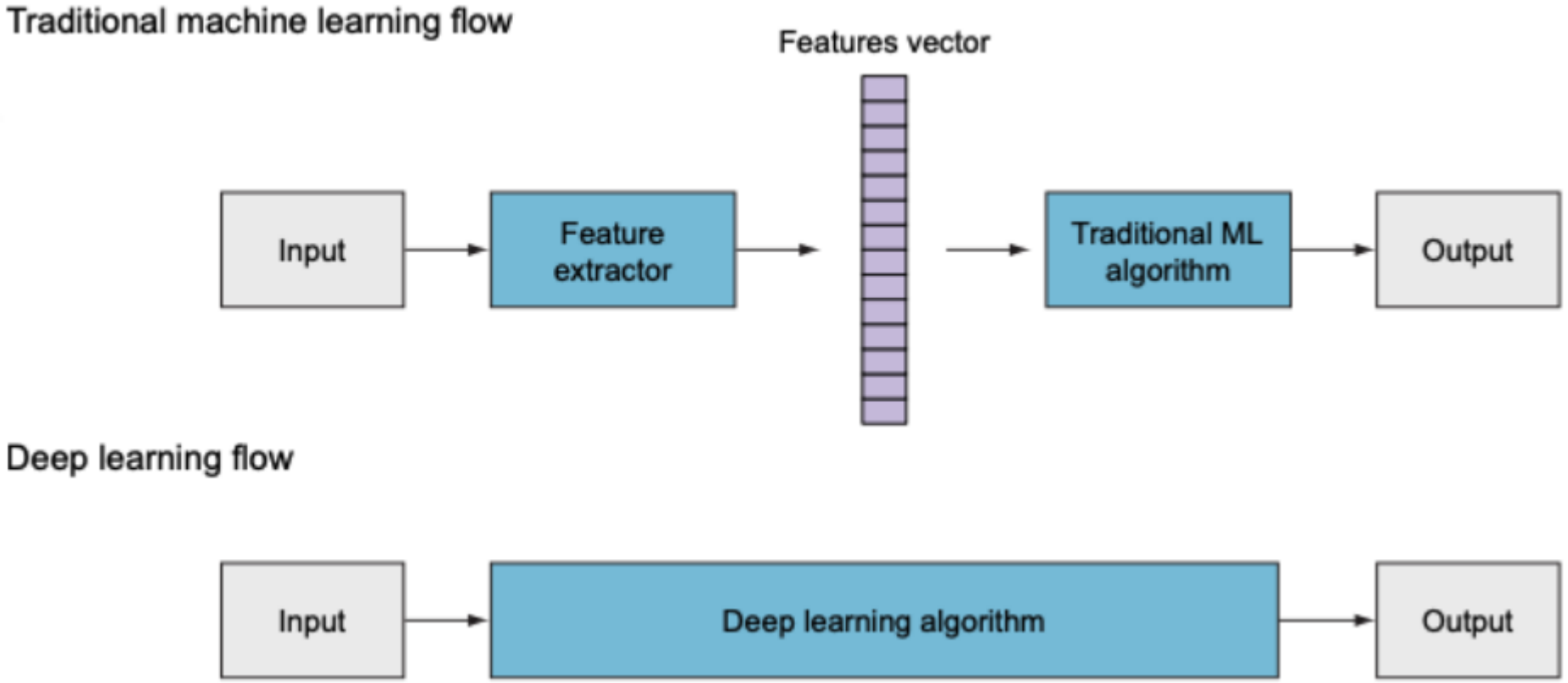
What about Deep Learning?

But heavily

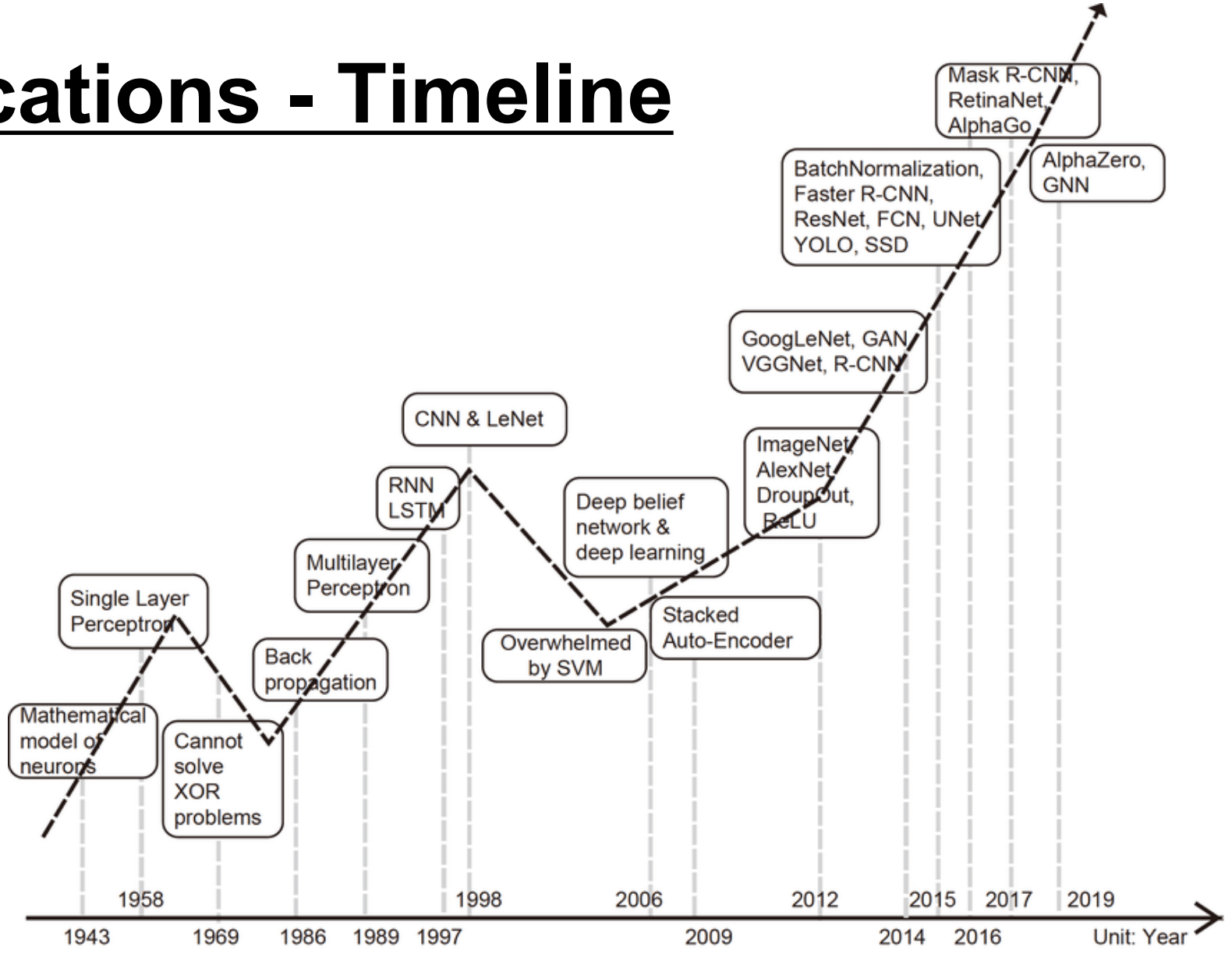


Traditional ML vs Deep Learning



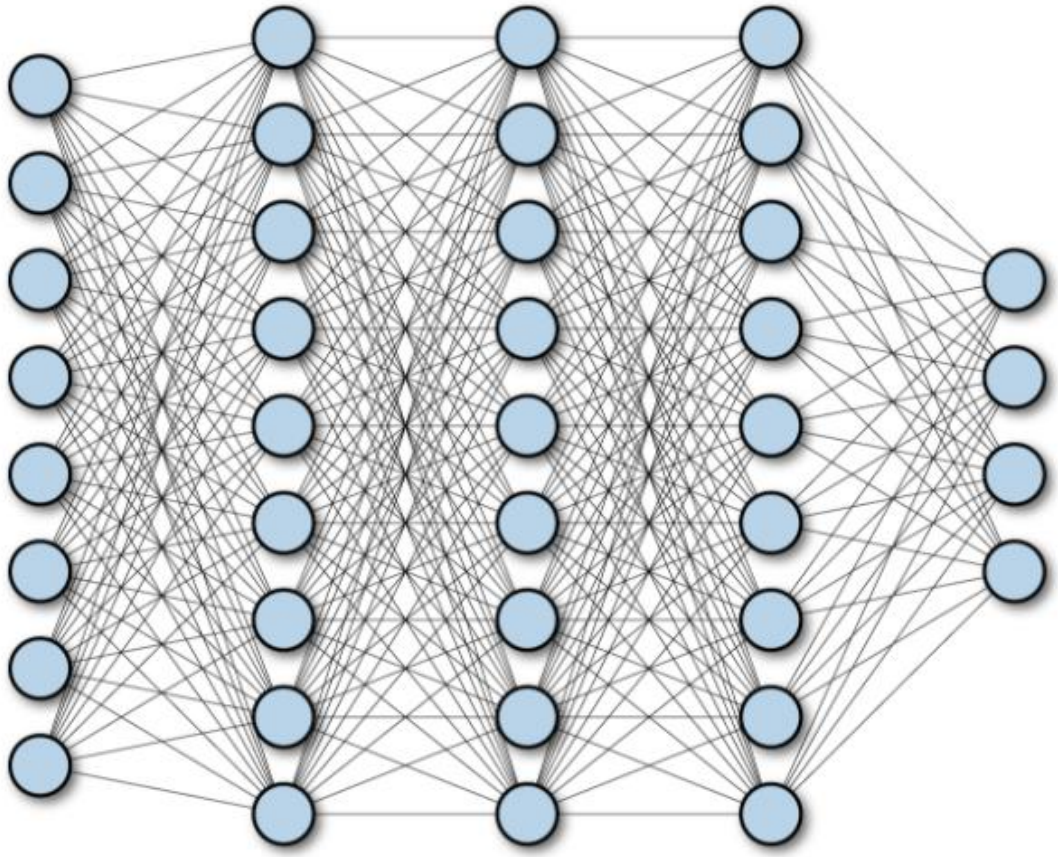


Applications - Timeline



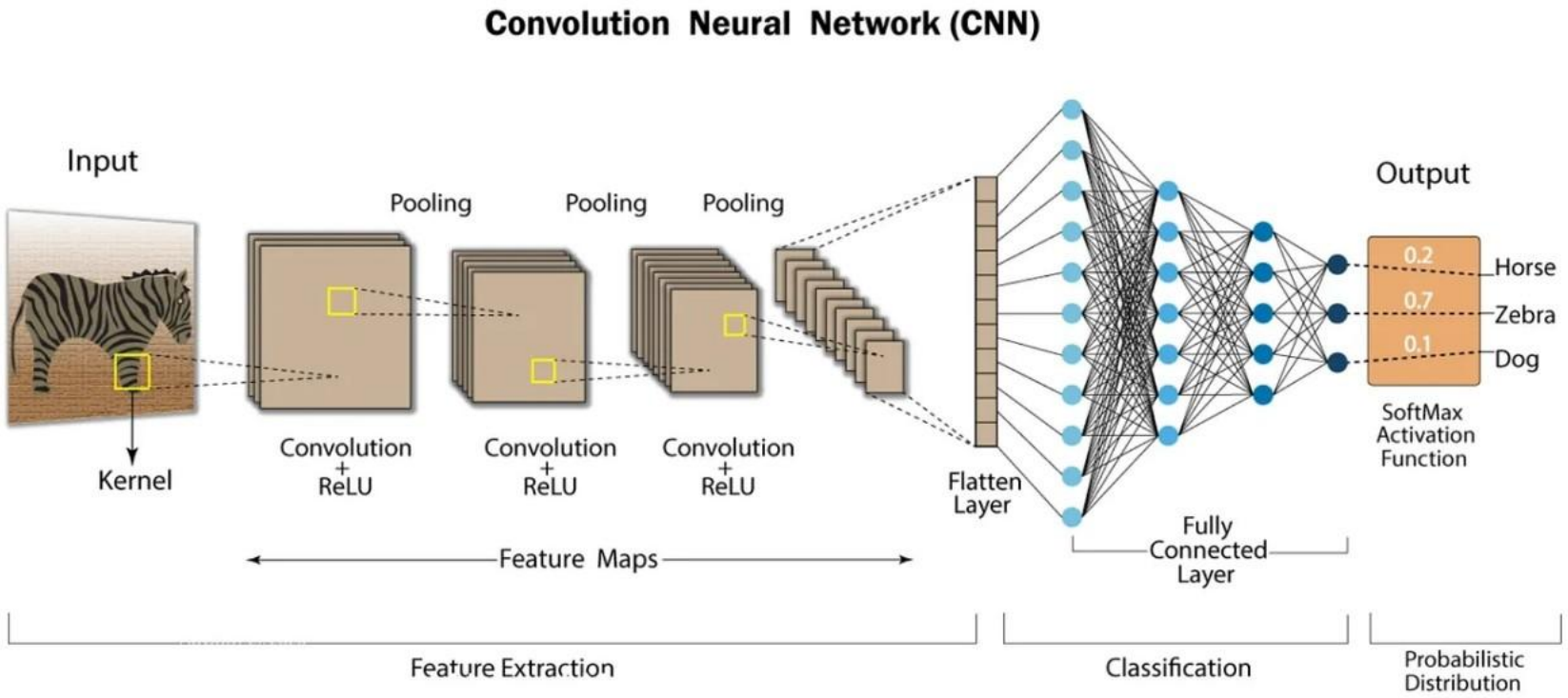
Feedforward Network

- Image and speech recognition
- Classification tasks like fraud detection, forecasting and prediction (e.g., weather, finance)
- Natural language processing, and regression analysis for various problems.



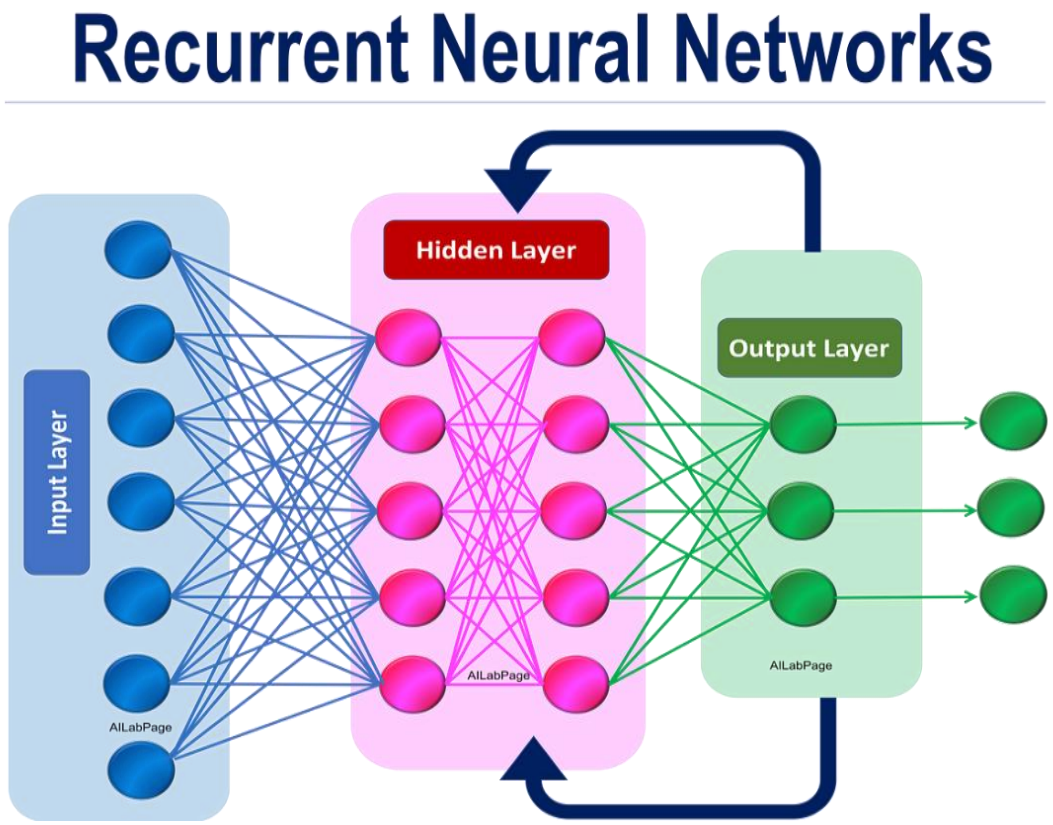
Convolutional Neural Networks (CNN)

- Image classification
- Image segmentation
- Image reconstruction
- Object detection
- Facial recognition
- Image generation



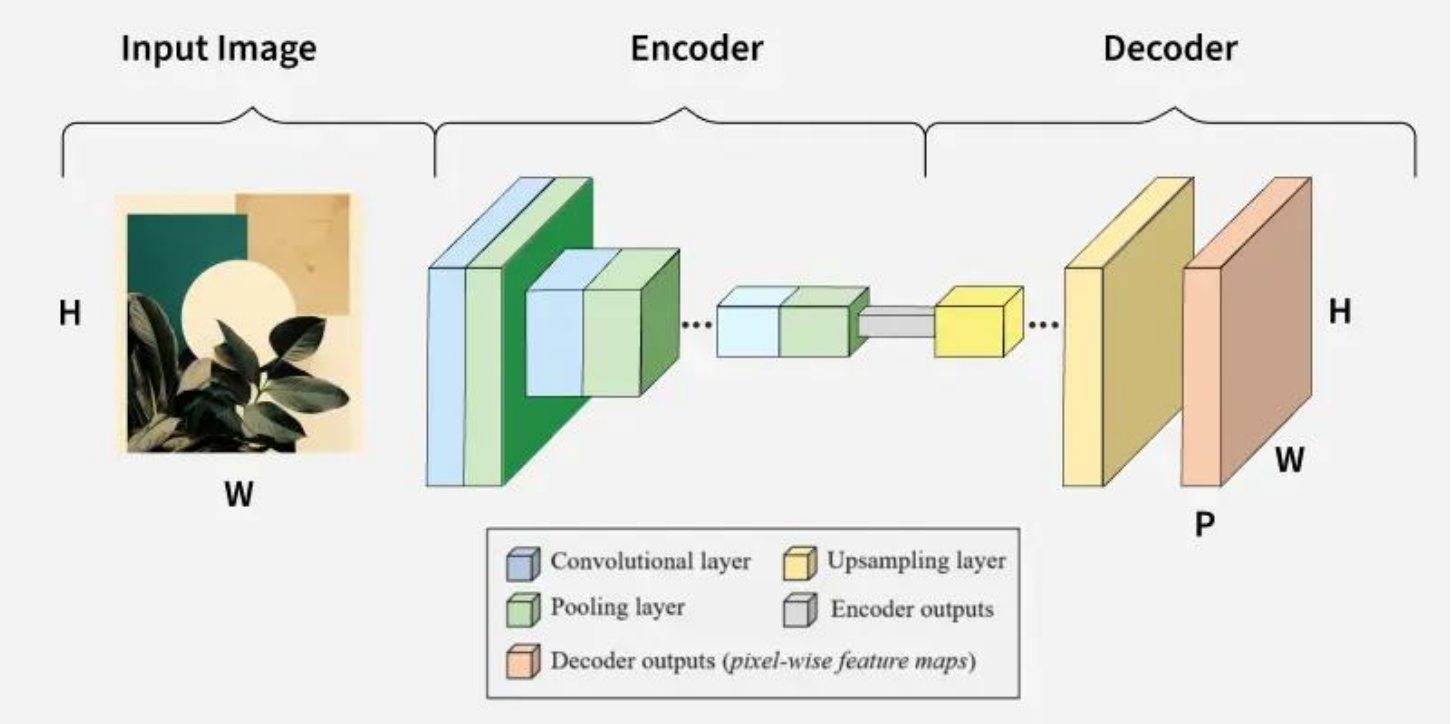
Recurrent Neural Networks (RNN)

- Temporal series
  - Stock market prediction
  - Forecasting
- Natural language processing
- Sentiment analysis
- Text generation



Encoders, Decoders, Autoencoders

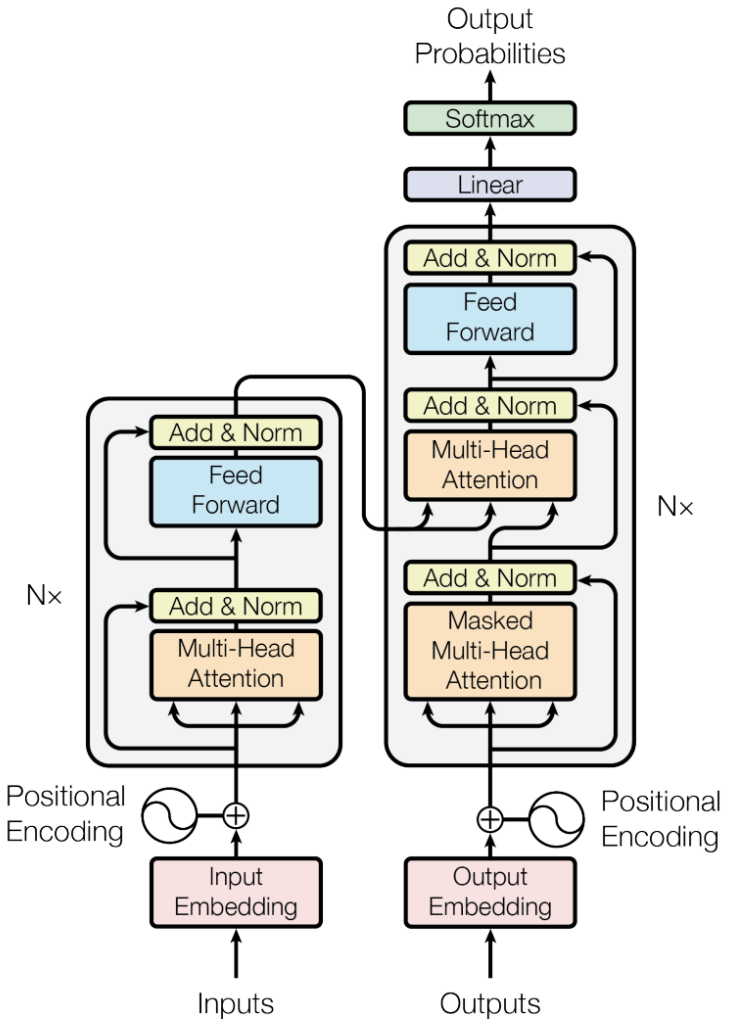
- Machine translation
- Text summarization
- Image and video captioning
- Speech recognition



Transformers

BERT

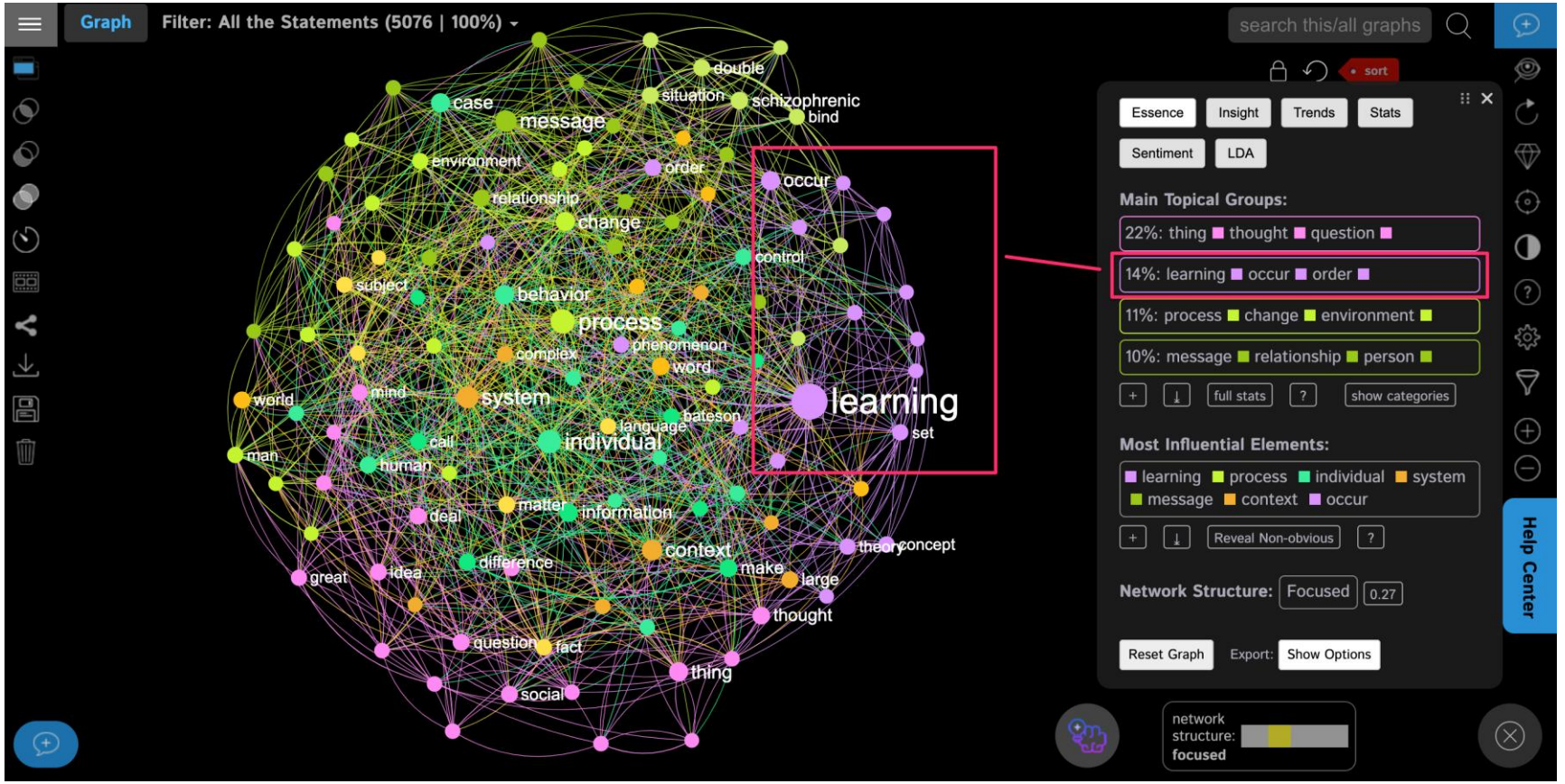
Encoder



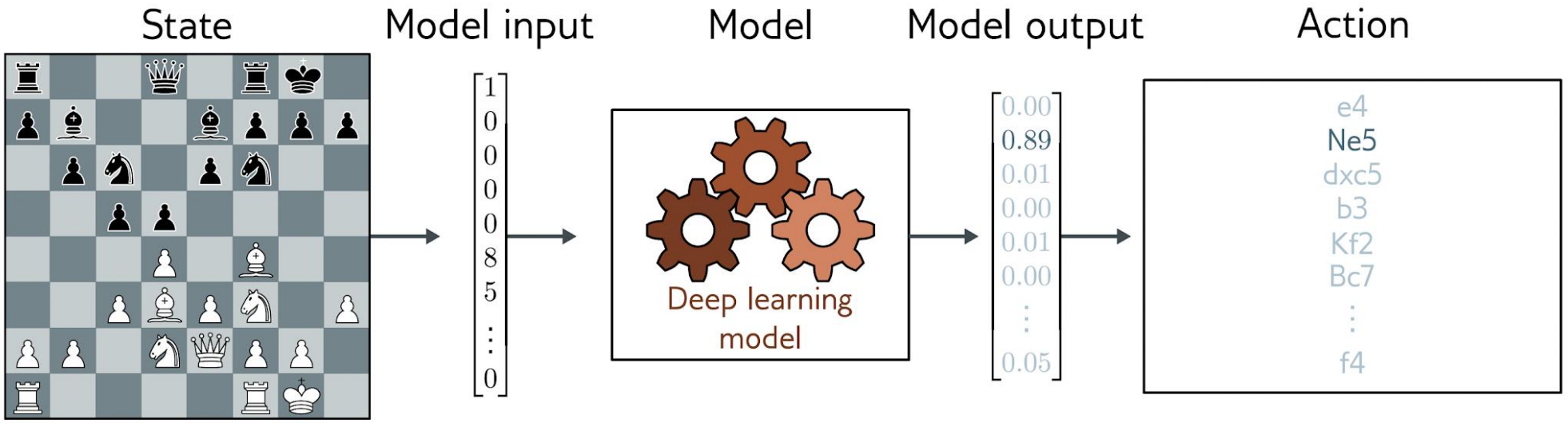
GPT

Decoder

Transformers



Deep Reinforcement Learning



Thank you for your attention

Please leave your feedback.

Use the QR code or link:

SJTU Winter School- Tozadore Feedback – Fill in form