COMP432: Machine Learning

Introduction to Machine Learning

Computer Science and Software Engineering (CSSE)
Concordia University, Fall 2024

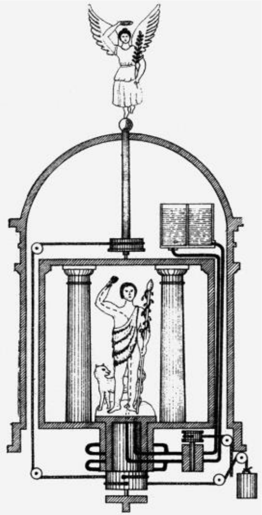
Outline

- What is artificial intelligence?
- What is machine learning?
- Type of problems (*classification*, *regression*)
- Types of learning (*supervised*, *unsupervised*)
- Dataset and features
- Objective functions
- Training
- Examples

What is Artificial Intelligence?

- Artificial Intelligence aims to build **machines** that can mimic **human** intelligence.
- Humanity has long dreamed about creating **intelligent machines**.

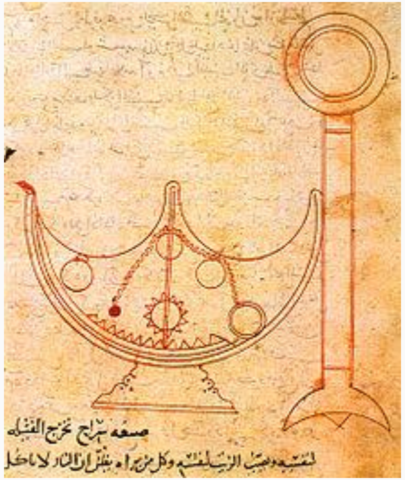
Talos was a giant automaton made of bronze to protect Crete from pirates and invaders.



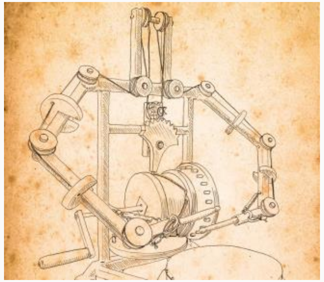
Hero of Alexandria (10 AD – 70 AD) was a Greek mathematician and engineer. He wrote a book entitled "On Automaton-Making". He designed tons of automatic machines, including an "automated theater".

What is Artificial Intelligence?

- Artificial Intelligence aims to build **machines** that can mimic **human** intelligence.
- Humanity has long dreamed about creating **intelligent machines**.



Banu Musa brothers (AD 800-860) published the "Book of Ingenious Devices". Among the other inventions, they designed the water-powered musical machine.



Leonardo da Vinci (1452 – 1519) was an Italian polymath of the renaissance who was active as a painter, draughtsman, engineer, scientist, theorist, sculptor, and architect. He designed many robots and automated systems.

What is Artificial Intelligence?

- Artificial Intelligence aims to build **machines** that can mimic **human** intelligence.
- Humanity has long dreamed about creating **intelligent machines**.

Joseph Möllinger (1715-1772), a dulcimer-playing automaton that mimics human movements to play eight compositions.

Wolfgang von Kempelen (1734 – 1804) “invented” a chess-playing Mechanical Turk (later revealed as a hoax).

https://www.nature.com/articles/d41586-018-05773-y
https://news.stanford.edu/2019/02/28/ancient-myths-reveal-early-fantasies-artificial-life/
https://sitn.hms.harvard.edu/flash/2017/history-artificial-intelligence/

What is Artificial Intelligence?

- Early systems were **mechanical machines** able to perform a sequence of basic operations.

- It was possible to reach higher levels of AI only with the advent of programmable **digital computers** (~1950).

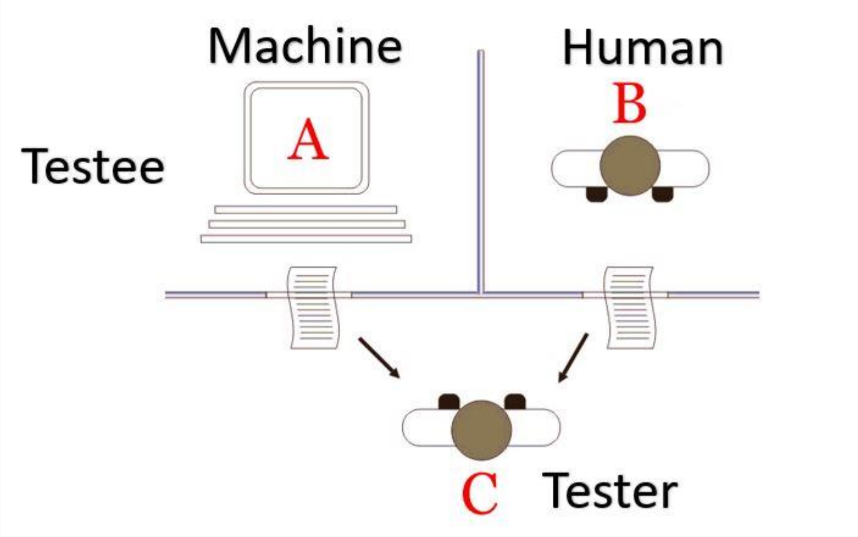
"What we want is a machine that can learn from experience"

"The possibility of letting the machine change its own instructions provides the mechanism for this"

Alan Turing (1912-1954)

Turing Test

- When a machine is "**intelligent**"?



Turing Test

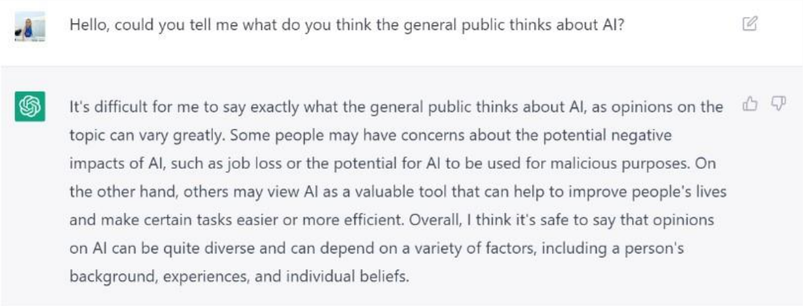
- Let A and B be a **computer** and a **human**, respectively (testee).
- Let C be a **human** (tester) that interrogates both A and B on a specific subject.
- After several questions, C is asked to decide **which respondent is human** and which is a computer.
- The test is repeated many times.
- If the tester (C) **can't recognize which candidate is human and which candidate is a compute**r the computer successfully passed the Turing test.

According to the Turing test, a machine is intelligent if it can engage in a **conversation** with a **human without being detected** as a **machine.**

Turing Test

- Did we ever build a machine that passes the Turing Test?

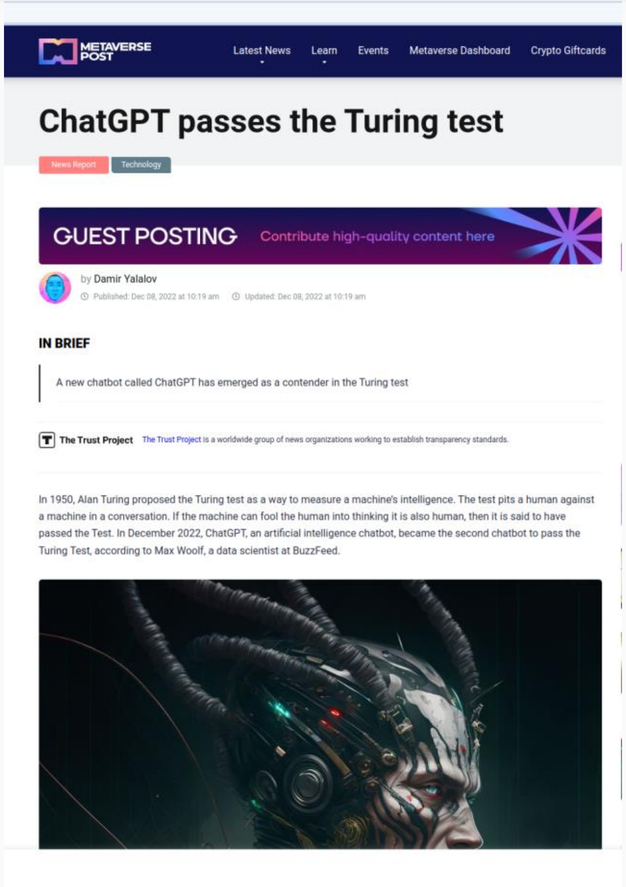
- According to some experts, **ChatGPT** (a famous chatbot launched by OpenAI in November 2022) passes it.


Hello, could you tell me what do you think the general public thinks about AI?

It's difficult for me to say exactly what the general public thinks about AI, as opinions on the topic can vary greatly. Some people may have concerns about the potential negative impacts of AI, such as job loss or the potential for AI to be used for malicious purposes. On the other hand, others may view AI as a valuable tool that can help to improve people's lives and make certain tasks easier or more efficient. Overall, I think it's safe to say that opinions on AI can be quite diverse and can depend on a variety of factors, including a person's background, experiences, and individual beliefs.

- But, are large language models really intelligent?

https://www.washingtonpost.com/technology/2022/06/17/google-ai-lamda-turing-test/


METAVERSE POST
Latest News Learn Events Metaverse Dashboard Crypto Giftcards

ChatGPT passes the Turing test

News Report Technology

GUEST POSTING Contribute high-quality content here

by Damir Yalalov
Published: Dec 08, 2022 at 10:19 am Updated: Dec 08, 2022 at 10:19 am

IN BRIEF

A new chatbot called ChatGPT has emerged as a contender in the Turing test

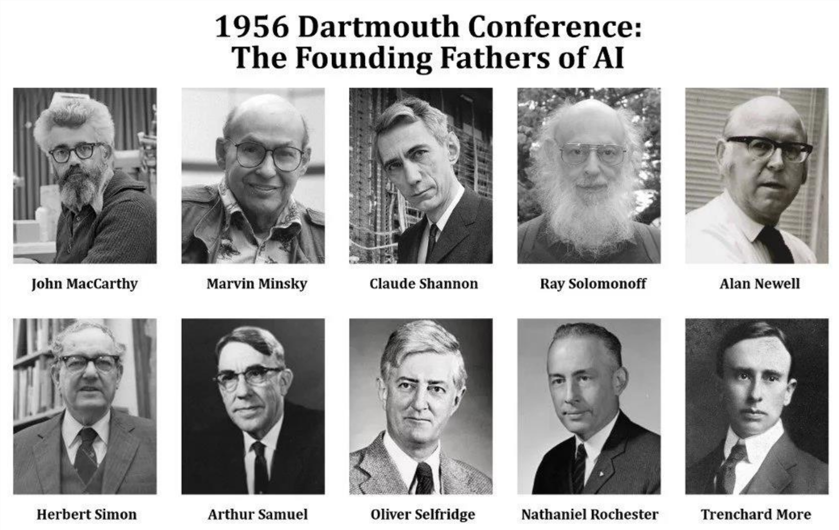
The Trust Project The Trust Project is a worldwide group of news organizations working to establish transparency standards.

In 1950, Alan Turing proposed the Turing test as a way to measure a machine's intelligence. The test pits a human against a machine in a conversation. If the machine can fool the human into thinking it is also human, then it is said to have passed the Test. In December 2022, ChatGPT, an artificial intelligence chatbot, became the second chatbot to pass the Turing Test, according to Max Woolf, a data scientist at BuzzFeed.

https://mpost.io/chatgpt-passes-the-turing-test/

Classical AI

- **Classical AI** is also known as “*symbolic AI*”, “*rule-based AI*,” and “*good old-fashioned AI* (GOFAI)”



- It dominated the field of AI from the mid-1950s until the middle 1990s.

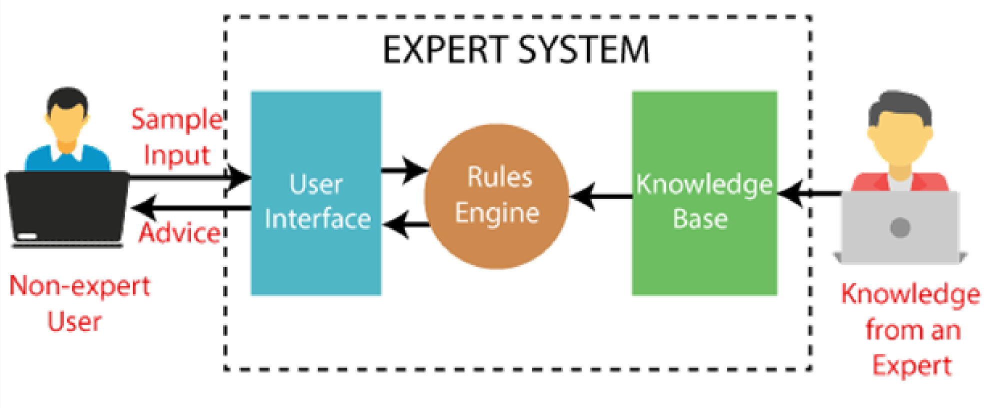
Basic Idea: Hard-code human knowledge using a set of rules added to knowledge bases.

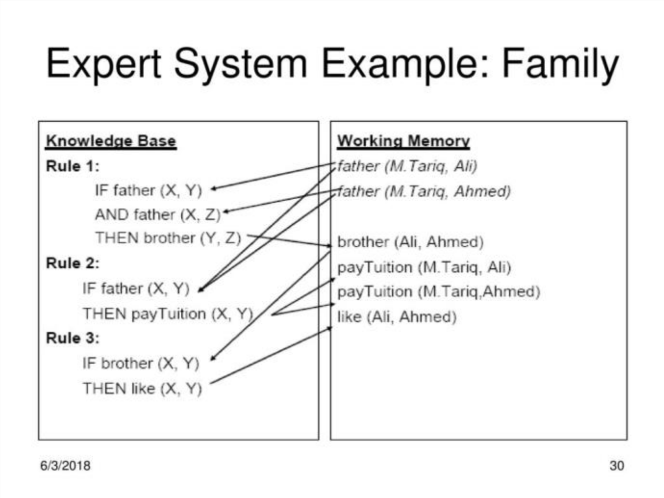
Symbols are used to define everything: things (e.g., cat, car, airplane, etc.), people (teacher, police, salesperson), and abstract concepts (bank transaction).

Rules define relations across symbols (through IF-THEN statements). The set of rules is hard-coded in a knowledge base.

Classical AI

- **Example: Expert Systems**





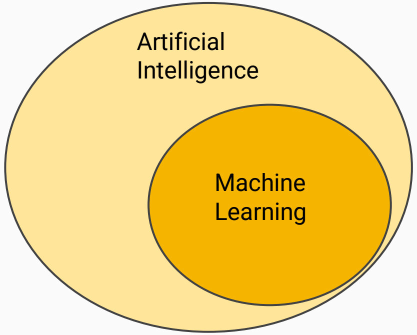
Symbolic artificial intelligence is convenient for problems that can be described by a list of **formal rules.**

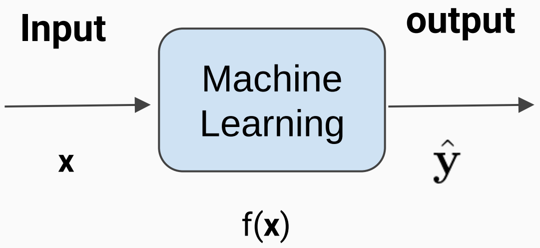
Many tasks cannot be easily formalized with a set of rules (e.g., recognizing faces in images, recognizing spoken words, etc.).

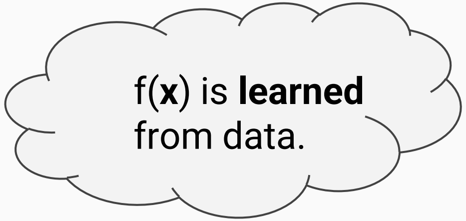
Can we acquire knowledge from data directly?

What is machine learning?

Machine learning aims to build machines that **learn from data** (or, more in general, from **experience**).







- We want a function *f(x)* that maps the input ***x*** into the desired output ***y***:

$$\boxed{\hat{\mathbf{y}} = f(\mathbf{x})} \quad \mathbf{x} \in \mathbb{R}^D, \hat{\mathbf{y}} \in \mathbb{R}^K \quad f : \mathbb{R}^D \to \mathbb{R}^K$$

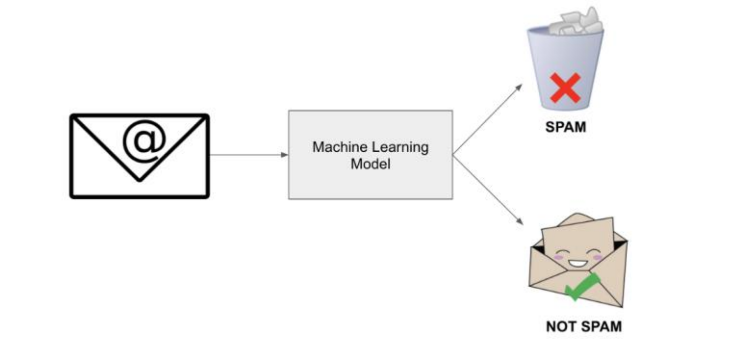
- In practice, we show to machine many **input-output examples**. The algorithm should learn the mapping between the input and output spaces with these examples.

Types of Problems

Classification: the machine learning algorithm has to specify to which of the *k* categories some input belongs to.

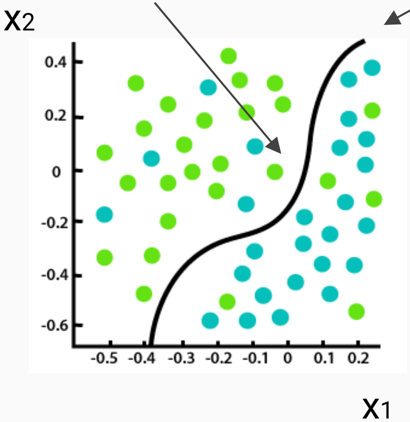
Row-vector (by convention)

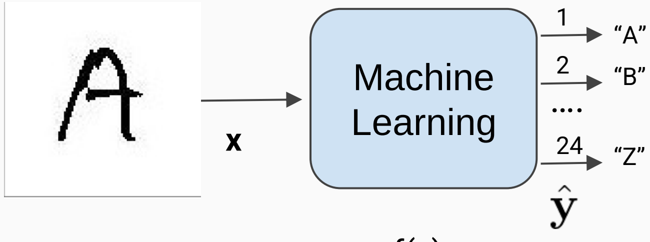
$$\boxed{\hat{y} = f(\mathbf{x})} \quad \mathbf{x} = [x_1, x_2, .., x_D]^T \quad \mathbf{x} \in \mathbb{R}^D \quad \hat{y} = \{1, ..., K\}$$



Decision boundaries

f(**x**)





f(**x**)

Types of Problems

Sequence-to-sequence Classification: the machine learning algorithm converts an input sequence into another one.

Machine Translation

[“Je”, “ne”, “sais”, “pas”] → **French** → Machine Learning → [“I”, “don’t”, “know”] **English**

f(**x**)

Speech Recognition

→ Machine Learning → [“Hi”, “Everyone”]

f(**x**)

Types of Problems

Regression: the machine learning algorithm predicts a continuous value given some input.

$$\boxed{\hat{y} = f(\mathbf{x})} \quad \mathbf{x} = [x_1, x_2, .., x_D]^T \quad \mathbf{x} \in \mathbb{R}^D \quad \hat{y} \in \mathbb{R}$$

$$f : \mathbb{R}^D \rightarrow \mathbb{R}$$

Row-vector
(by convention)

Number of floors: **2**

Number of bedrooms: **3**

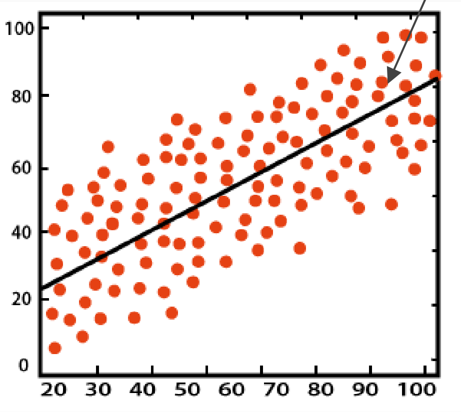
Age: **15**

$\mathbf{x}$ → Machine Learning → **Price** $\hat{y}$ → *665k$*

f(**x**)

f(**x**)

X2



X1

Traditional Programming vs Machine Learning

Traditional Programming

Human Knowledge

Code Writing

Code Use

Machine Learning

Dataset

Model Training

Model Use

Main Differences:

- With machine learning, we replace **human knowledge** with **data**.
- We replace the hand-crafted code with a function f(**x**) **learned from data**.

Traditional Programming vs Machine Learning

Example: max value of a list

Traditional Programming

Human Knowledge

```python
def get_max(lst):
  max_value = - math.inf
  for elem in lst:
      if elem > max_value:
        max_value = elem
  return max_value
```

Code Writing

```python
lst = [2, 10, 5, 6]
print(get_max(lst))
```

```
10
```

Code Use

Machine Learning

```python
inputs  = [[2, 10, 5, 6],
           [1, 4, -5, 6],
           [-1, 0, 0, 1],
           [2, 11, 2, 2]
           ]
```

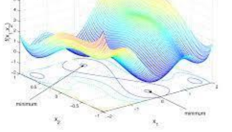
Targets = [10, 6, 1, 11]

Dataset

Model Training

Machine Learning

f(x)=**get_max**(x)



```python
lst = [2, 10, 5, 6]
print(get_max(lst))
```

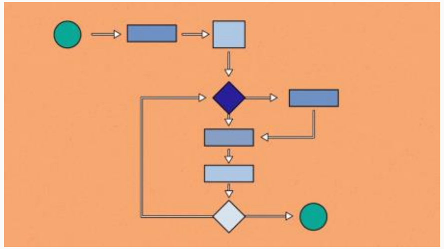
```
10
```

Model Use

Traditional Programming vs Machine Learning

When using traditional programming

- If the problem can be efficiently solved with a well-defined algorithm.



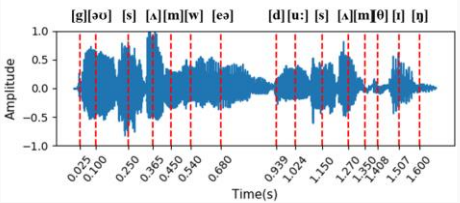
e.g., sorting, search, hashing algorithms.

When using machine learning

- If the problem cannot be solved with a list of formal rules.

- It is easy to collect data to solve the problem.

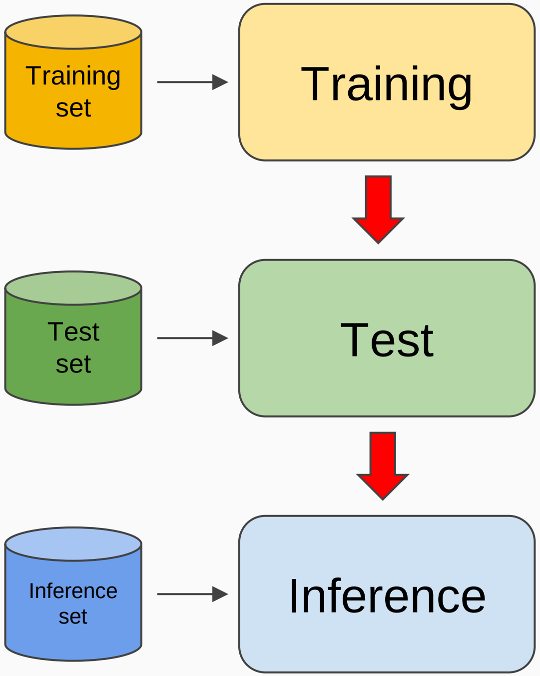
e.g., face recognition, speech recognition



Machine Learning Stages

Machine Learning Stages

- A machine learning algorithm typically goes through the following **stages**:



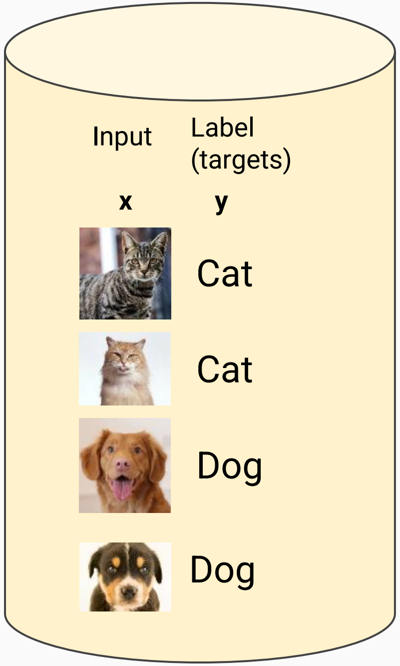
Learn a function that "explains" well the training data.

Evaluate the performance on data never seen before (test set).

Use the model (production).

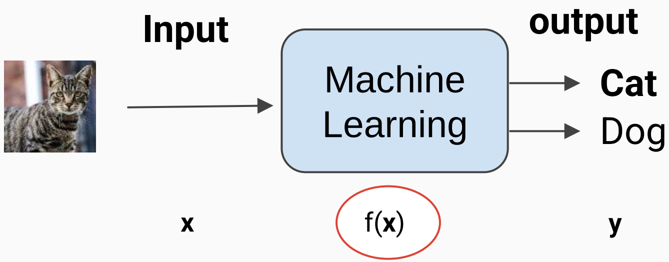
Classification Example

Example: Cat vs Dog classification.

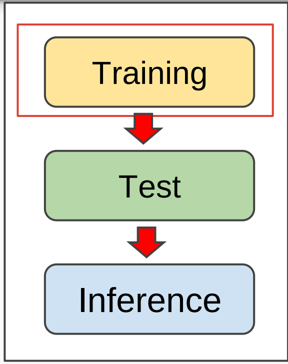


Training Set

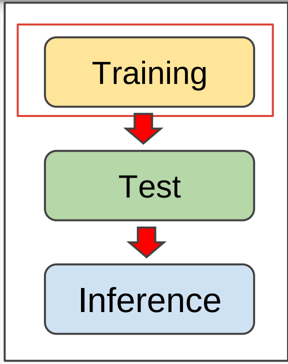
- Phase 1: **Training** (Learning)



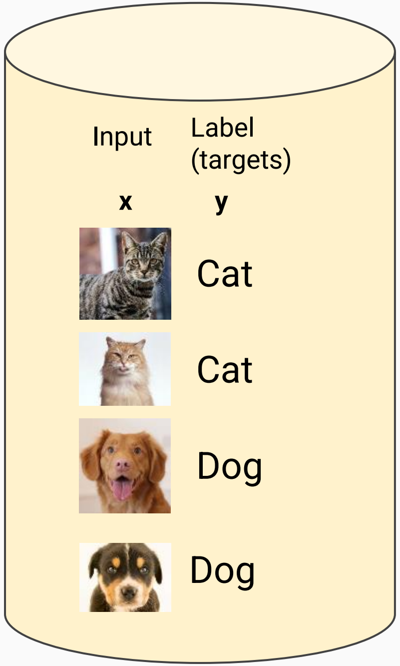
- We show to the machine the input-output examples collected in a **training set**.
- The goal of training is finding f(x) such that it "*explains well*" the training data.



Classification Example

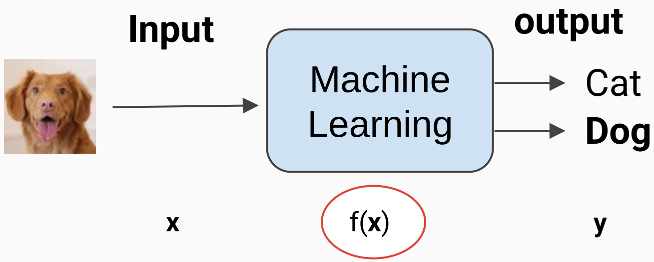


Example: Cat vs Dog classification



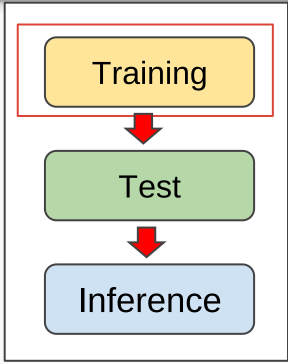
Training Set

- Phase 1: **Training** (Learning)

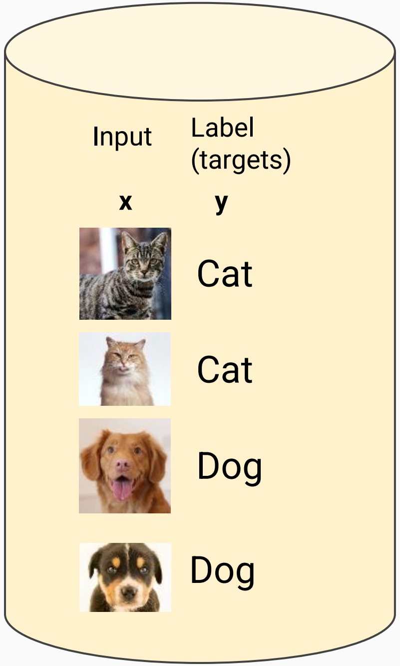


- We show to the machine the input-output examples collected in a **training set**.
- The goal of training is finding f(x) such that it "*explains well*" the training data.

Classification Example

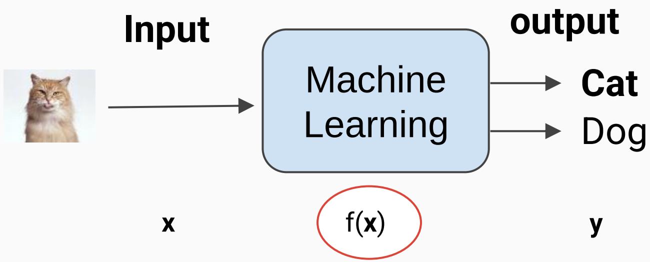


Example: Cat vs Dog classification



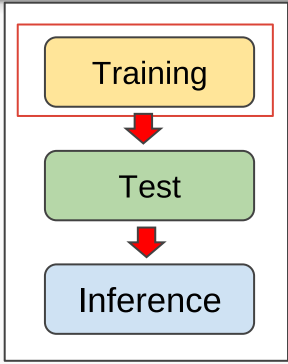
Training Set

- Phase 1: **Training** (Learning)

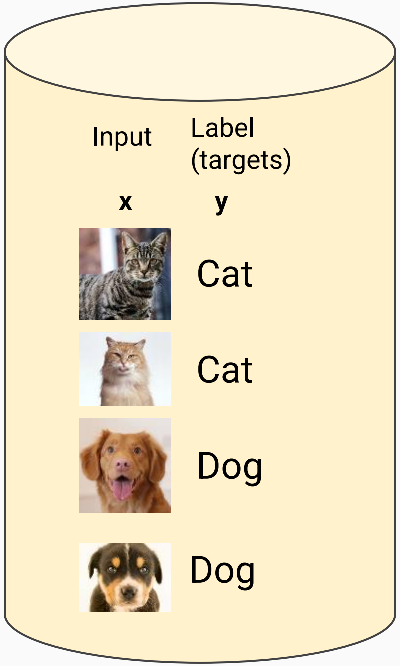


- We show to the machine the input-output examples collected in a **training set**.
- The goal of training is finding f(x) such that it "*explains well*" the training data.

Classification Example

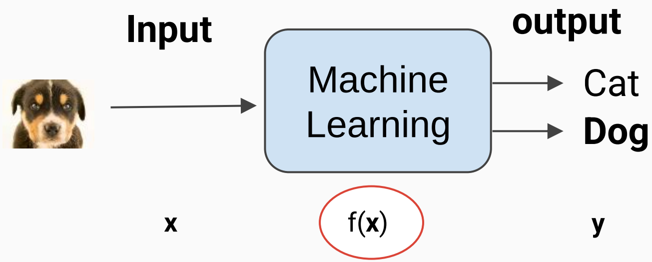


Example: Cat vs Dog classification



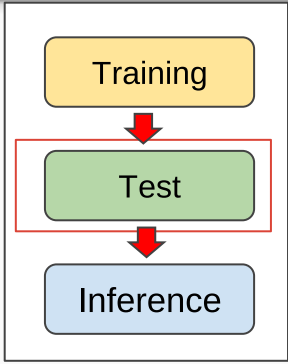
Training Set

- Phase 1: **Training** (Learning)

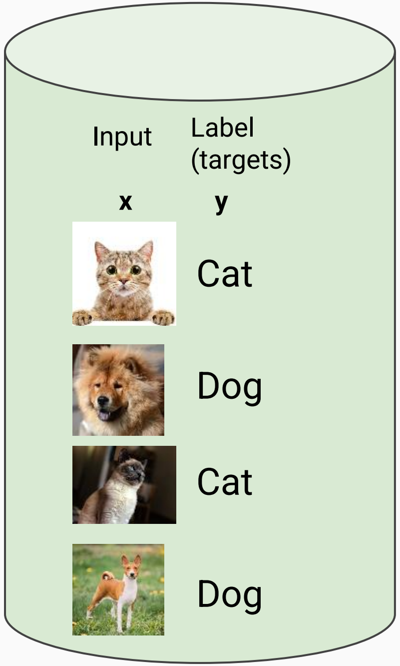


- We show to the machine the input-output examples collected in a **training set**.
- The goal of training is finding a f(x) such that it "*explains well*" the training data.

Classification Example

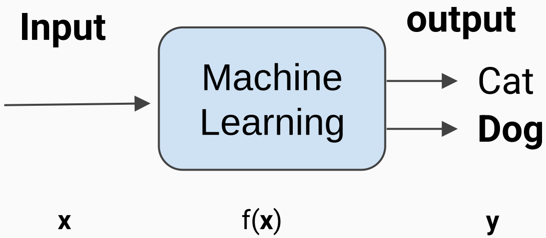


Example1 : Cat vs Dog classification



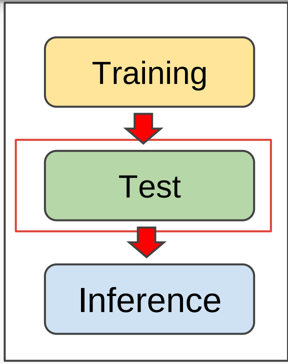
Test Set

- Phase 2: **Test** (Evaluation)

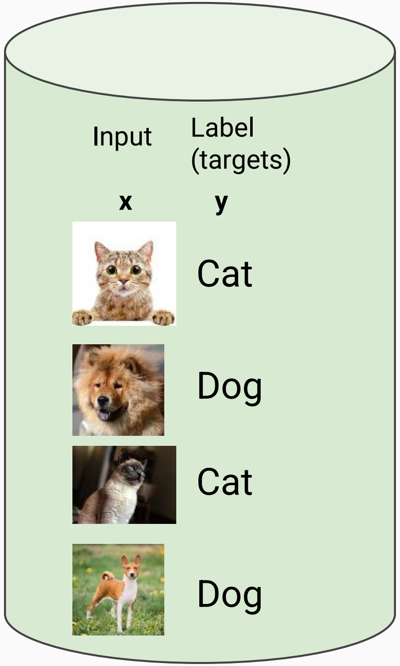


- A good machine learning model should perform well on data never seen before. This ability is called **generalization**.
- The goal of the testing phase (evaluation) is to **measure the performance** on data never seen before (collected in a test set).

Classification Example

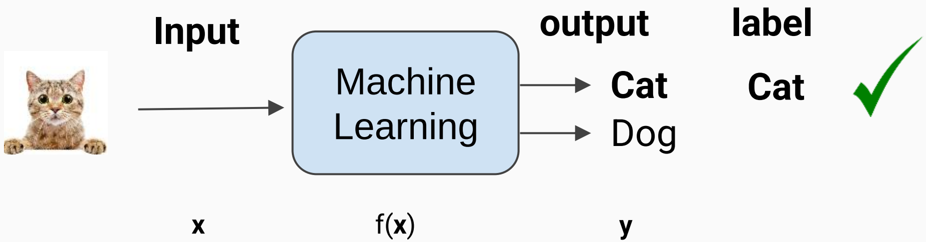


Example: Cat vs Dog classification



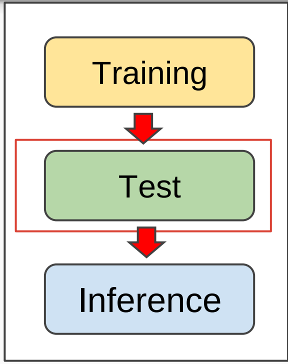
Test Set

- Phase 2: **Test** (Evaluation)

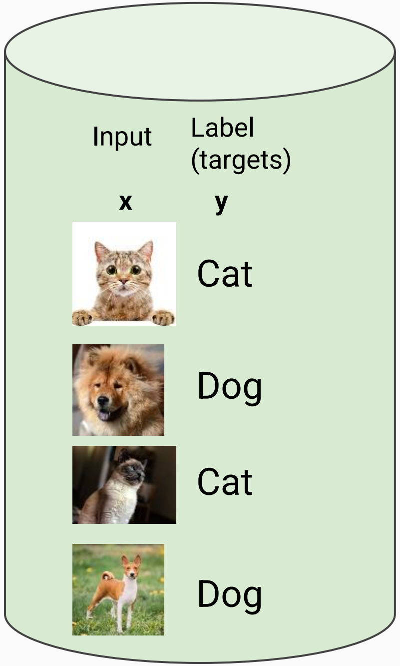


- A good machine learning model should perform well on data never seen before. This ability is called **generalization**.
- The goal of the testing phase (evaluation) is to **measure the performance** on data never seen before (collected in a test set).

Classification Example

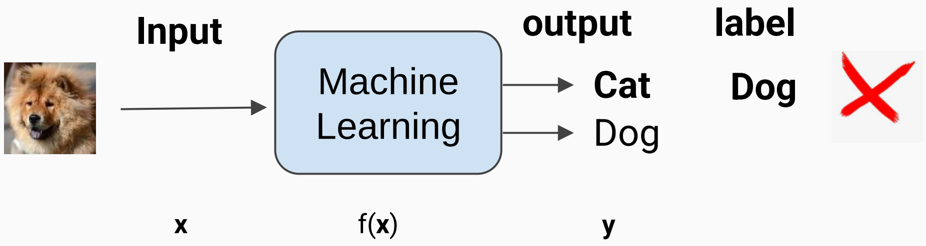


Example: Cat vs Dog classification



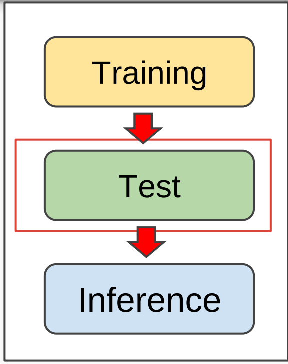
Test Set

- Phase 2: **Test** (Evaluation)

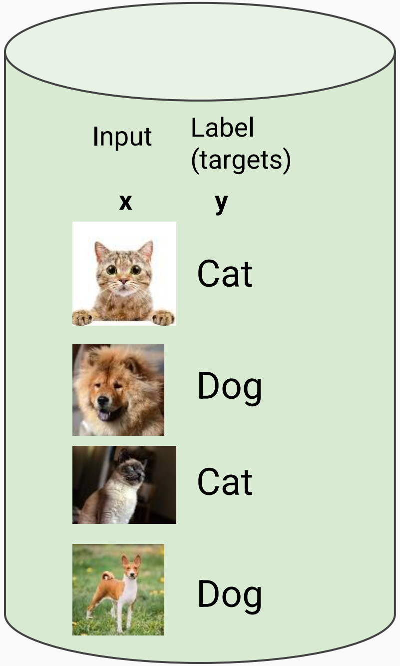


- A good machine learning model should perform well on data never seen before. This ability is called **generalization**.
- The goal of the testing phase (evaluation) is to **measure the performance** on data never seen before (collected in a test set).

Classification Example

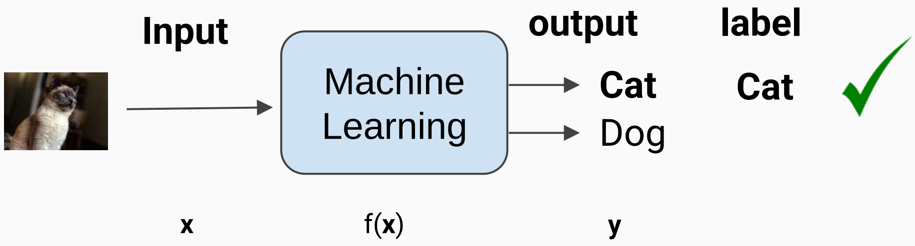


Example: Cat vs Dog classification



Test Set

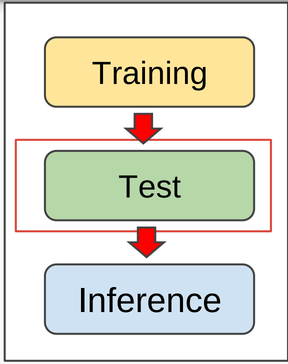
- Phase 2: **Test** (Evaluation)



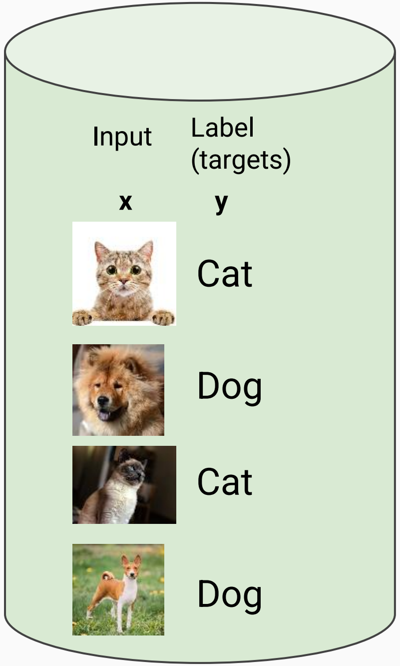
- A good machine learning model should perform well on data never seen before. This ability is called **generalization**.

- The goal of the testing phase (evaluation) is to **measure the performance** on data never seen before (collected in a test set).

Classification Example

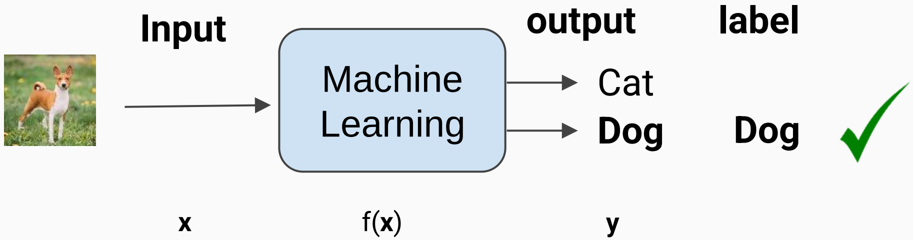


Example: Cat vs Dog classification



Test Set

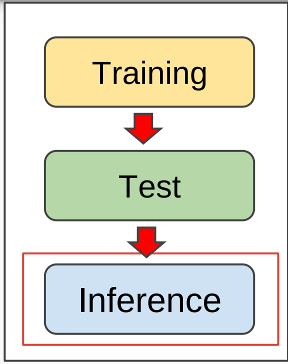
- Phase 2: **Test** (Evaluation)



- We classified correctly 3 of the 4 entries.
- The **test accuracy** of the systems is: $Acc(\%) = \frac{N_{correct}}{N_{tot}} \cdot 100$

$$Acc(\%) = \frac{3}{4} \cdot 100 = 75\%$$

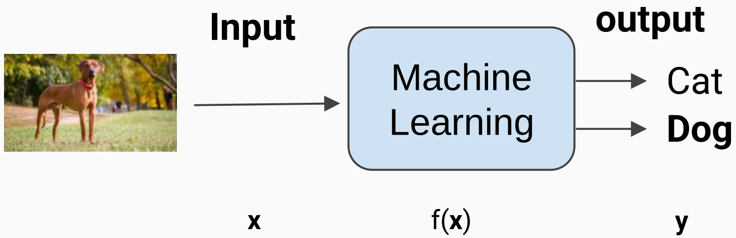
Classification Example



Example: Cat vs Dog classification

Inference Set

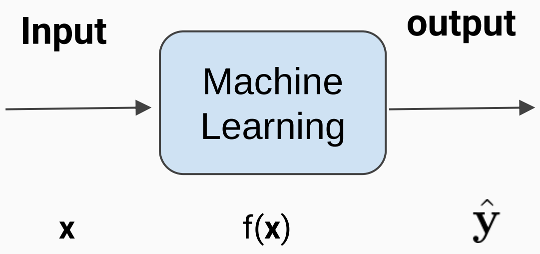
- Phase 3: **Inference**



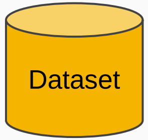
- When we are happy with our machine learning algorithm, we can eventually put it in “production”.
- **Inference** is the process of using a trained machine learning algorithm by **running live data points** (without knowing their label).

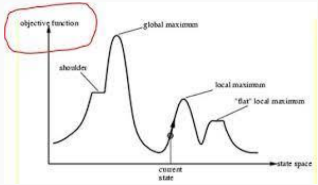
Basic Components

Basic Components



$$\boxed{\hat{\mathbf{y}} = f(\mathbf{x})} \quad \mathbf{x} \in \mathbb{R}^D, \hat{\mathbf{y}} \in \mathbb{R}^K$$



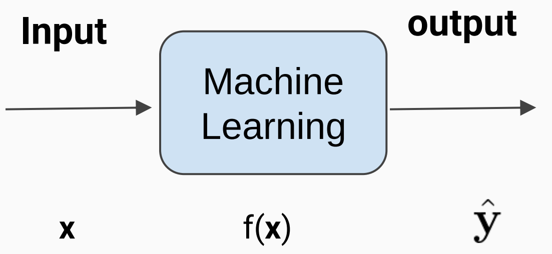


The basic components of a machine learning system are:

- **Datasets**: examples of input-output mappings.
- **Machine learning model**: implements the function f(x) that maps the inputs into the desired outputs.
- **Objective function**: a measure of "how well" the solution f(x) fits the data.
- **Optimization algorithm**: an algorithm that finds the function f(x) among different candidates

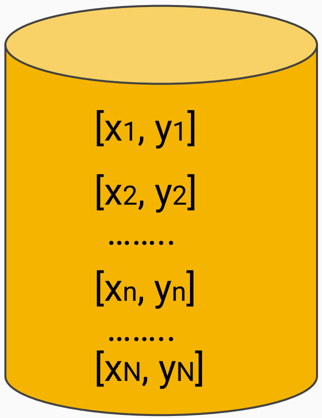
Basic Components:
Datasets

Datasets

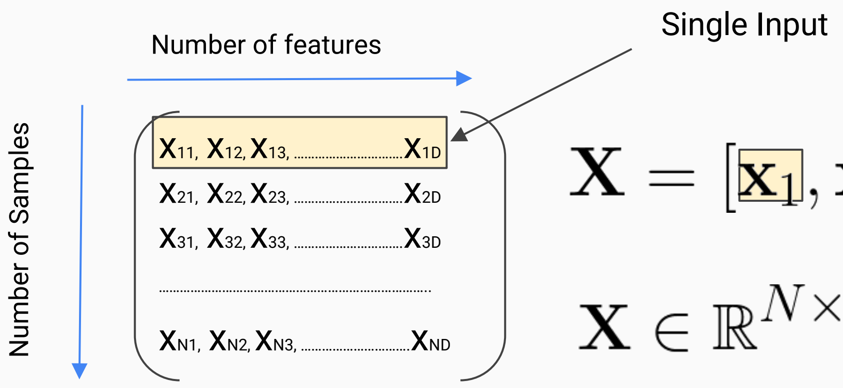


- A dataset is a **collection** of **examples** (sometimes called **data points** or **samples**) containing the desired input-output mappings.

- The inputs **x** are often gathered into a **matrix**:



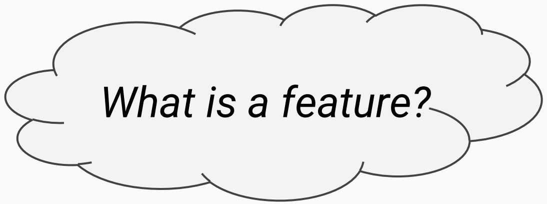
Dataset



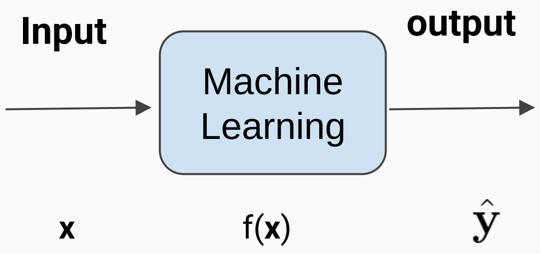
Feature Matrix

$$\mathbf{X} = [\boxed{\mathbf{x}_1}, \mathbf{x}_2, \ldots, \mathbf{x}_N]^T$$

$$\mathbf{X} \in \mathbb{R}^{N \times D}$$



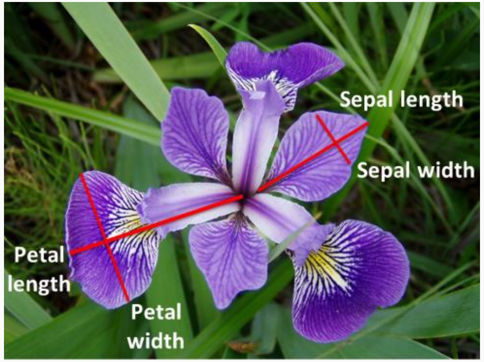
Features



- A **feature** is a measurable property (attributes) potentially relevant for solving a machine learning problem.

- Each example contains a **feature vector,** which is a collection of some relevant measures.

IRIS classification: 3 classes (*Iris setosa*, *Iris virginica*, *Iris versicolor*)

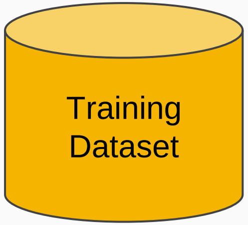


$$
\begin{array}{cccc}
\text{Sepal Length} & \text{Sepal Width} & \text{Petal Length} & \text{Petal Width} \\
\end{array}
$$
$$
\begin{pmatrix}
5.1 & 3.5 & 1.4 & 0.2 \\
4.9 & 3.0 & 1.4 & 0.2 \\
7.0 & 3.2 & 4.7 & 1.4 \\
\ldots\ldots & \ldots\ldots & \ldots\ldots & \ldots\ldots \\
5.8 & 3.3 & 6.0 & 2.5
\end{pmatrix}
$$

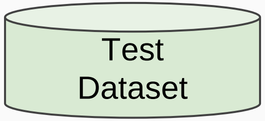
$$\mathbf{X} \in \mathbb{R}^{150\times 4}$$

Training and Test Datasets

- In machine learning, we use at least two distinct datasets:



Training Dataset: the set of examples used to find the desired mapping function $f(\mathbf{x})$.



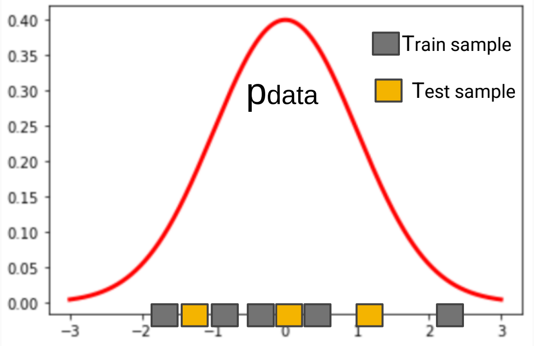
Test Dataset: The set of examples used to assess the performance of the machine learning algorithm (after training).

Training and test sets must contain **different examples**.

If the test samples are already seen during training, we **overestimate** the performance of the system (just like the performance of a student would be much better if the final exam has the same exercises provided as homework).

Training and Test Datasets

- Even though training and test samples are different, a common assumption is that they are sampled from **the same data generation process** p_{data}.



- In this case, for instance, training and test samples are drawn from the same Gaussian distribution.

- **Note**: In real machine learning problems, p_{data} is complex and not accessible. We can only observe samples drawn from it (e.g, images of cats).

- We also assume that each sample is drawn **independently** from any other data point.

- If these conditions hold, the samples are called **independent** and **identically distributed** (*i.i.d*).

Supervised Learning

- **Supervised learning** is **learning from labeled examples**. Training and test examples contain both the input **x** and the desired output y.
- **Classification** and **regression** are supervised learning problems.

Number of features

Number of Samples

$$\begin{pmatrix} x_{11}, & x_{12}, & x_{13}, & \cdots & x_{1D} \\ x_{21}, & x_{22}, & x_{23}, & \cdots & x_{2D} \\ x_{31}, & x_{32}, & x_{33}, & \cdots & x_{3D} \\ \cdots & & & & \\ x_{N1}, & x_{N2}, & x_{N3}, & \cdots & x_{ND} \end{pmatrix}$$

Feature Matrix

$$\mathbf{X} \in \mathbb{R}^{N \times D}$$

Number of Outputs

Number of Samples

$$\begin{pmatrix} y_{11}, & y_{12}, & \cdots & y_{1K} \\ y_{21}, & y_{22}, & \cdots & y_{2K} \\ y_{31}, & y_{32}, & \cdots & y_{3K} \\ \cdots & & & \\ y_{N1}, & y_{N2}, & \cdots, & y_{NK} \end{pmatrix}$$

Target Matrix (labels or supervision)

$$\mathbf{Y} \in \mathbb{R}^{N \times K}$$

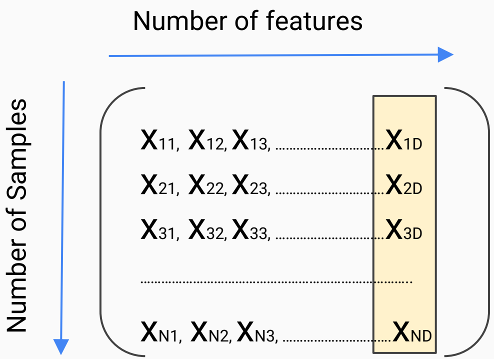
Examples:

- *Linear models*
- *Neural networks*
- *Support vector machine*
- *Naive Bayes*
- *K-nearest neighbor*
- *Random forest*

Unsupervised Learning

- **Unsupervised learning** is about **learning from observation**. Training and test examples only contain the input x.

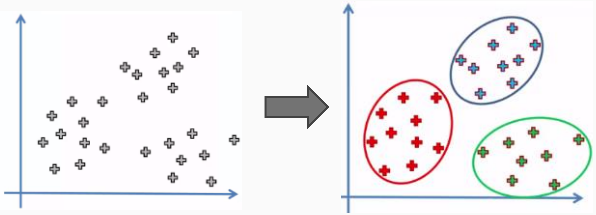
Even if we do not have any labels, we can still **observe the data** and hopefully find **useful properties** in their **structure**.



Feature Matrix

$$\mathbf{X} \in \mathbb{R}^{N \times D}$$

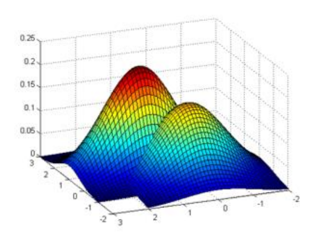
<u>Clustering</u>



We group “close” data into the same cluster.

We cannot give a label to the detected clusters, but likely they contain different types of inputs.

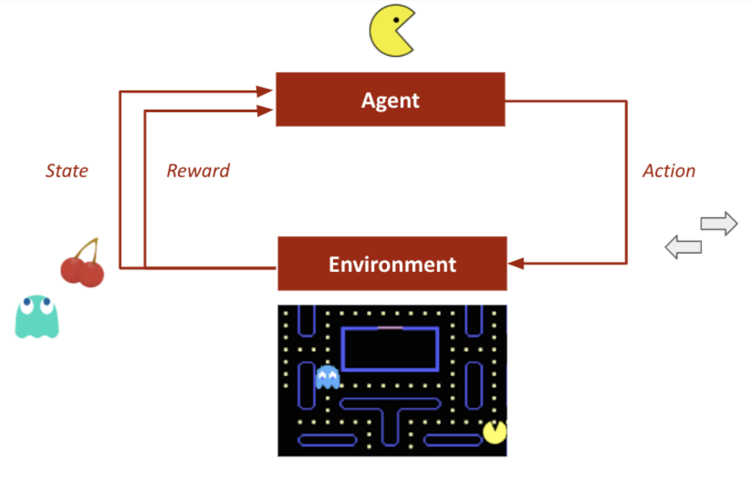
<u>Probability Density Estimation</u>



We use training data to estimate pdata. At test time you can say **how likely** is a certain data point.

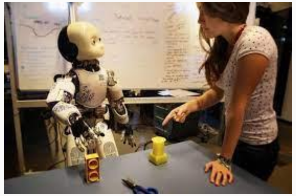
Reinforcement Learning

- **Reinforcement learning** is **learning by interaction**. Training and test examples are not “fixed” but are collected by interacting with an environment.



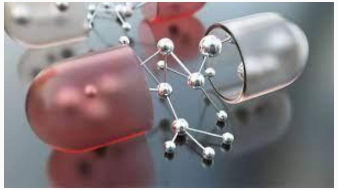
- Our machine learning algorithm is an agent immersed in an **environment**.
- The agent performs **actions**.
- The environment gets either **rewards** or **penalties** for the actions taken.
- The agent is trained to **maximize** the **rewards**.

Automated Game playing



Robotics

Self-Driving Car

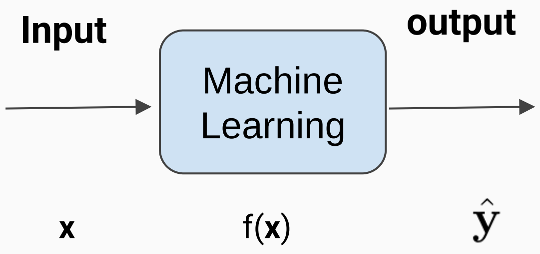


Drug Discovery

Basic Components:
Machine Learning Model

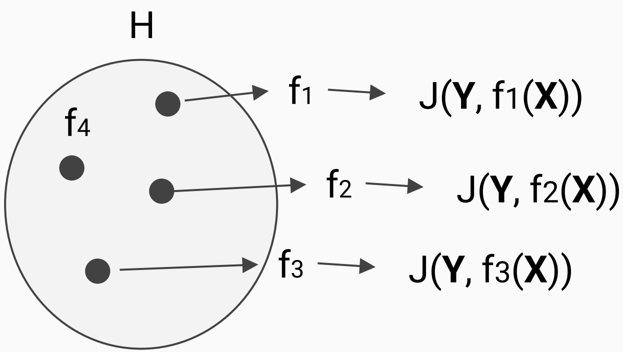
Machine Learning Algorithm

A machine learning algorithm is a **function** f that maps the input into the output.



$$\boxed{\hat{\mathbf{y}} = f(\mathbf{x})} \quad \mathbf{x} \in \mathbb{R}^D, \hat{\mathbf{y}} \in \mathbb{R}^K$$

The set of functions that a machine learning algorithm can implement is the **hypothesis space**.

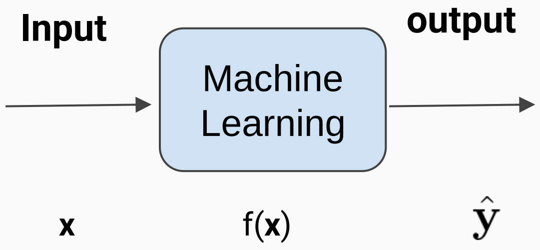


Every function in the hypothesis space is a possible **solution**.

For every solution, we can compute the **objective function** using the training dataset. This tells us "*how good*" is a certain solution.

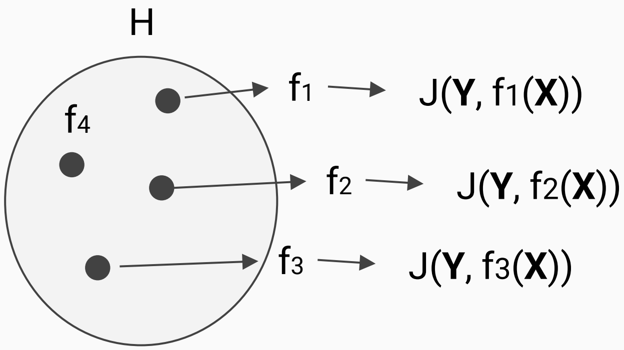
During training, we **explore** the **hypothesis space** until we found a function that **explains well the data**.

Training



$$\boxed{\hat{\mathbf{y}} = f(\mathbf{x})} \quad \mathbf{x} \in \mathbb{R}^D, \hat{\mathbf{y}} \in \mathbb{R}^K$$

Training a machine learning model is about finding a function *f* that explains well the training data:

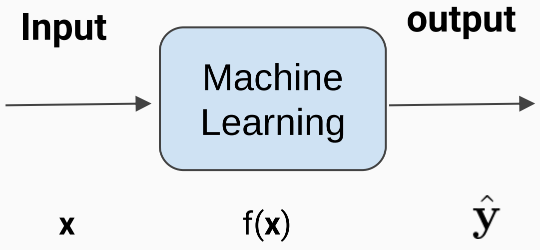


$$\boxed{f^* = \underset{f \in H}{\operatorname{argmin}} \, J(\mathbf{Y}, f(\mathbf{X}))}$$

Training a machine learning model requires solving an **optimization problem**.

Basic Components:
Objective Function

Objective



$$\boxed{\hat{\mathbf{y}} = f(\mathbf{x})} \quad \mathbf{x} \in \mathbb{R}^D, \hat{\mathbf{y}} \in \mathbb{R}^K$$

- Training a machine learning model aim to find a function f that “**fits well**” with my training data.

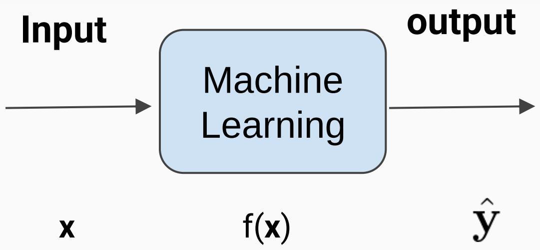
- To quantify “**how good**” are the predictions obtained with the learning function we need to define an **objective function**:

$$\boxed{J(\mathbf{Y}, f(\mathbf{X})) : \mathbb{R}^{N \times K} \to \mathbb{R}}$$

- This function is also called **criterion**.

- By convention, we often want to **minimize** it.

Objective



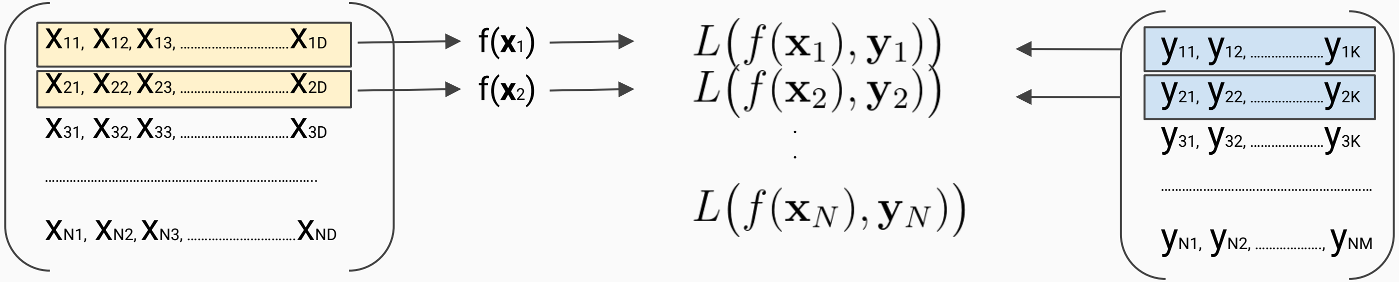
$$\boxed{\hat{\mathbf{y}} = f(\mathbf{x})} \quad \mathbf{x} \in \mathbb{R}^D, \hat{\mathbf{y}} \in \mathbb{R}^K$$

- Often, the objective is written as an average (or sum) over the training samples:

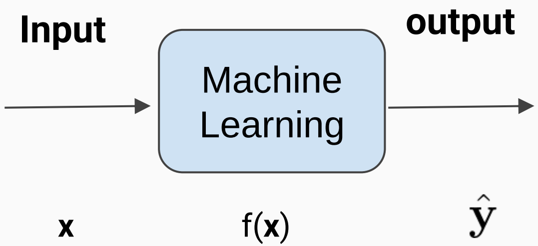
$$\boxed{J(\mathbf{Y}, f(\mathbf{X})) = \frac{1}{N}\sum_{i=1}^{N} L(f(\mathbf{x}_i), \mathbf{y}_i))}$$

The term L is called **Loss**.

The training process based on minimizing such an objective is called **empirical risk minimization**.



Loss for Regression



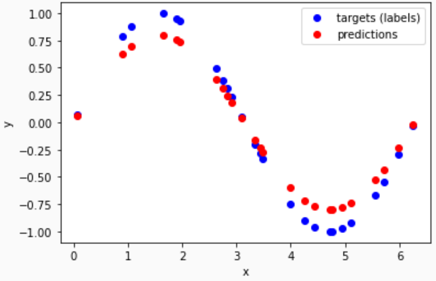
<u>Inputs:</u>

$$\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_i, \ldots, \mathbf{x}_N]^T$$

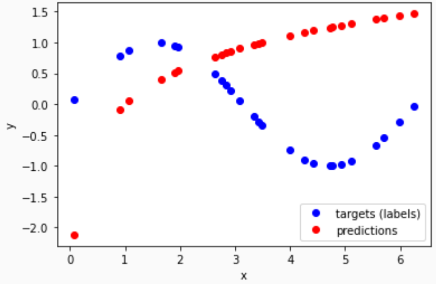
<u>Targets:</u>

$$\mathbf{y} = [y_1, y_2, \ldots, y_i, \ldots, y_N]^T$$

$$y_i \in \mathbb{R}$$



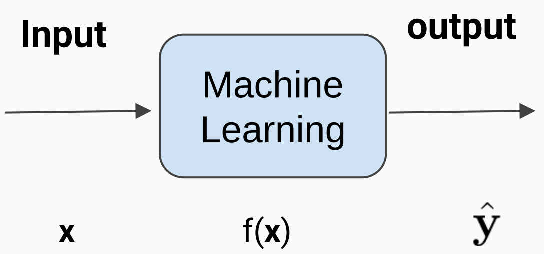
Good Solution



Bad Solution

- How would you measure how good a solution is this case?

Mean Squared Error



Inputs:

$$\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_i, \dots, \mathbf{x}_N]^T$$

Targets:

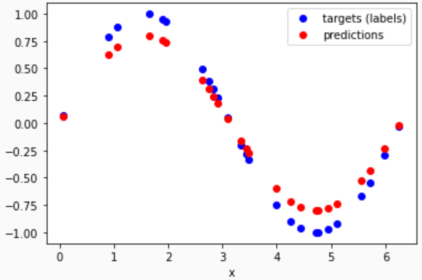
$$\mathbf{y} = [y_1, y_2, \dots, y_i, \dots, y_N]^T$$

$$y_i \in \mathbb{R}$$

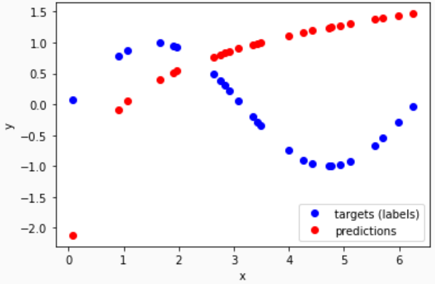
A popular objective used for regression problems is the **Mean Squared Error** (MSE):

$$\boxed{MSE = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - f(\mathbf{x}_i)\right)^2}$$

Intuitively, MSE measures **how far** the predictions are from the targeted ones.



MSE = 0.018

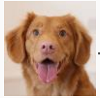


MSE = 2.35

Losses for Classification

For a classification problem, one possible objective is the **classification accuracy**:

$$\boxed{Acc = \frac{N_{correct}}{N_{tot}}} \qquad Err = 1 - Acc$$



$\mathbf{x}_i$

Machine Learning

p(cat)
p(dog)
p(bird)

p($\mathbf{x}_i$)

Output probabilities

argmax

$$\hat{y}_i = \underset{k}{\operatorname{argmax}}\, P(y_i = k \mid \mathbf{x}_i, \mathbf{w}_k)$$

Label

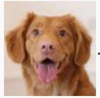
y_i = dog

Prediction

$\hat{y}_i = dog$

Acc

Accuracy



x_i → Machine Learning → p(cat), p(dog), p(bird) → argmax → Prediction

Label

y_i = dog

Prediction

$\hat{y}_i = dog$

→ Acc

	cat	**dog**	**bird**	
y = [	0,	1,	2	]^T
	y_1	y_2	y_3	

X = [  ]^T

$\mathbf{x}_1$ $\mathbf{x}_2$ $\mathbf{x}_3$

Number of Outputs

Number of Samples

$$p(\mathbf{X}) = \begin{pmatrix} \mathbf{0.6} & 0.2 & 0.2 \\ 0.3 & \mathbf{0.5} & 0.2 \\ 0.1 & \mathbf{0.5} & 0.4 \end{pmatrix} \begin{array}{l} p(\mathbf{x}_1) \\ p(\mathbf{x}_2) \\ p(\mathbf{x}_3) \end{array}$$

$$\begin{array}{cccc} & \text{cat} & \text{dog} & \text{bird} \end{array}$$

$$\mathbf{y} = [0, 1, 2]^T$$

$$\begin{array}{cccc} & \text{cat} & \text{dog} & \text{dog} \end{array}$$

$$\hat{\mathbf{y}} = [0, 1, 1]^T$$

$$Acc = \frac{N_{corr}}{N_{tot}} = \frac{2}{3} = 0.66$$

Accuracy

- Accuracy is a simple metric **easy to interpret** (good for test). However, it is a “**hard**” metric and sometimes machine learning algorithms prefer “**soft**” ones.

- To understand why it is a “hard” metric, let’s consider the following examples:

Example 1

$$p(\mathbf{X}) = \begin{array}{c} \begin{pmatrix} \color{blue}{0.6} & 0.2 & 0.2 \\ 0.3 & \color{blue}{0.5} & 0.2 \\ 0.1 & \mathbf{0.5} & 0.4 \end{pmatrix} \end{array} \begin{array}{l} p(\mathbf{x}_1) \\ p(\mathbf{x}_2) \\ p(\mathbf{x}_3) \end{array}$$

$$\mathbf{y} = [\overset{\text{cat}}{0}, \overset{\text{dog}}{1}, \overset{\text{bird}}{2}]^T \qquad \hat{\mathbf{y}} = [\overset{\text{cat}}{0}, \overset{\text{dog}}{1}, \overset{\text{dog}}{1}]^T$$

$$Acc = \frac{N_{corr}}{N_{tot}} = \frac{2}{3} = \boxed{0.66}$$

Example 2

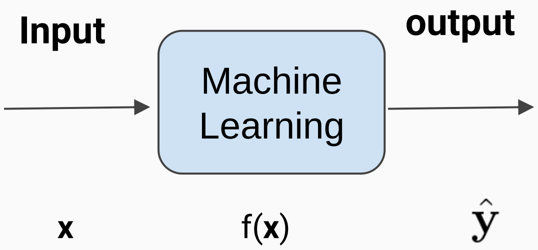
$$p(\mathbf{X}) = \begin{array}{c} \begin{pmatrix} \color{red}{0.9} & 0.1 & 0.0 \\ 0.0 & \color{red}{1.0} & 0.0 \\ 0.1 & \mathbf{0.5} & 0.4 \end{pmatrix} \end{array} \begin{array}{l} p(\mathbf{x}_1) \\ p(\mathbf{x}_2) \\ p(\mathbf{x}_3) \end{array}$$

$$\mathbf{y} = [\overset{\text{cat}}{0}, \overset{\text{dog}}{1}, \overset{\text{bird}}{2}]^T \qquad \hat{\mathbf{y}} = [\overset{\text{cat}}{0}, \overset{\text{dog}}{1}, \overset{\text{dog}}{1}]^T$$

$$Acc = \frac{N_{corr}}{N_{tot}} = \frac{2}{3} = \boxed{0.66}$$

- The Accuracy here is **the same**, but the second case looks better because the classifier is more confident about its predictions.

Categorical Cross-Entropy



Inputs:

$$\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_i, \dots, \mathbf{x}_N]^T$$

Targets:

$$\mathbf{y} = [y_1, y_2, \dots, y_i, \dots, y_N]^T$$

$$y_i \in \mathbb{R}$$

- A "softer" alternative is the categorical **cross-entropy**.

$$\boxed{CCE = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} y_{ik} ln(p_{ik})}$$

Natural Log

Also known as **Negative Log-Likelihood** (NLL)

Labels

$$\mathbf{y} = \begin{array}{c} \text{cat dog bird} \\ [0, 1, 2]^T \end{array} \Rightarrow \begin{array}{cc} & \text{\# Classes K} \rightarrow \\ \text{\# Examples N} \downarrow & \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \end{array}$$

One-hot label representation

Example 1

$$p(\mathbf{X}) = \begin{array}{cc} \begin{pmatrix} 0.6 & 0.2 & 0.2 \\ 0.3 & 0.5 & 0.2 \\ 0.1 & 0.5 & 0.4 \end{pmatrix} & \begin{array}{c} p(\mathbf{x}_1) \\ p(\mathbf{x}_2) \\ p(\mathbf{x}_3) \end{array} \end{array}$$

Cross Entropy = -⅓ *(
1 * ln(0.6) + 0 * ln(0.2) + 0 * ln(0.2) +
0 * ln(0.3) + 1 * ln(0.5) + 0 * ln(0.2) +
0 * ln(0.1) + 0 * ln(0.5) + 1 * ln(0.4)
)
= **0.70**

Categorical Cross-Entropy

Labels

$$\mathbf{y} = \overset{\text{cat}}{[0}, \overset{\text{dog}}{1}, \overset{\text{bird}}{2}]^T \Rightarrow \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

One-hot label representation

Example 1

$$p(\mathbf{X}) = \begin{pmatrix} 0.6 & 0.2 & 0.2 \\ 0.3 & 0.5 & 0.2 \\ 0.1 & 0.5 & 0.4 \end{pmatrix} \begin{matrix} p(\mathbf{x}_1) \\ p(\mathbf{x}_2) \\ p(\mathbf{x}_3) \end{matrix}$$

Cross Entropy = 0.70

Accuracy = 0.66

Example 2

$$p(\mathbf{X}) = \begin{pmatrix} 0.9 & 0.1 & 0.0 \\ 0.0 & 1.0 & 0.0 \\ 0.1 & 0.5 & 0.4 \end{pmatrix} \begin{matrix} p(\mathbf{x}_1) \\ p(\mathbf{x}_2) \\ p(\mathbf{x}_3) \end{matrix}$$

Cross Entropy = 0.34

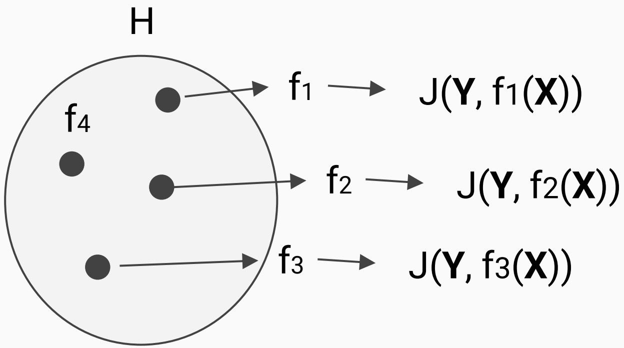
Accuracy = 0.66

- In this example, the two machine learning models have the same accuracy, but different cross-entropy.
- *Example 2* is better than *Example 1* because the classifier is **more confident** about its predictions. The cross-entropy is thus lower.
- We now have a metric much “softer” that the accuracy.
- The categorical cross-entropy ranges from 0 (**perfect solution**) to +inf (**bad solution**).
- We thus want to **minimize** this metric.

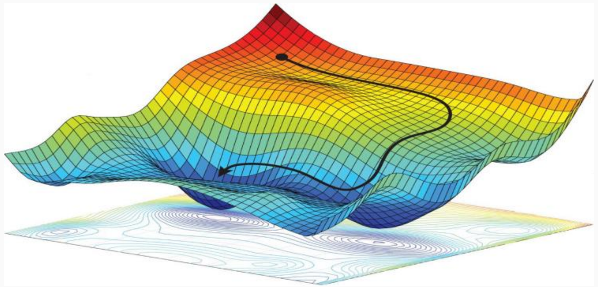
Basic Components:
Optimization

Optimization

- When we have a meaningful objective function, we need a way to find the function f(x) that **minimizes** it.



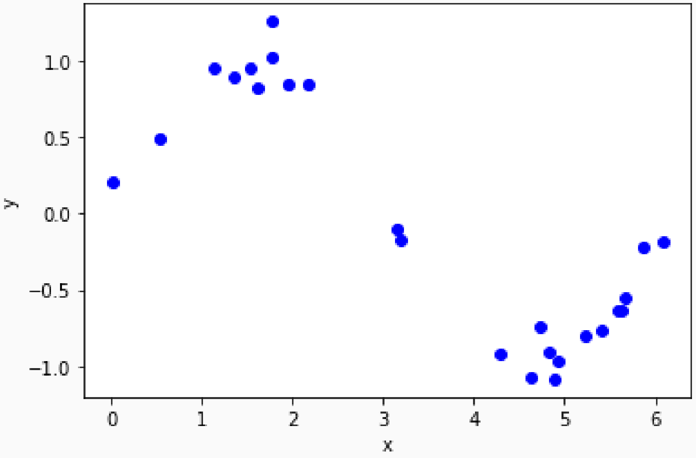
$$f^* = \underset{f \in H}{\operatorname{argmin}}\, J(\mathbf{Y}, f(\mathbf{X}))$$



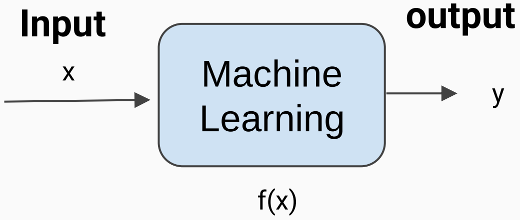
The algorithm that solves this minimization problem is called “**optimizer**”.

Optimization (Naive) Example

Example: Curve Fitting Problem



Training Set



We have a training set composed of:

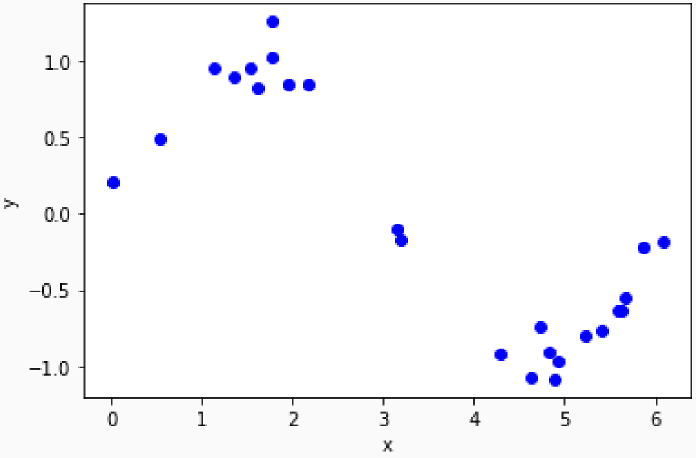
- Inputs $\mathbf{X} = [x_1, x_2, ..., x_i, ..., x_N]$
- Labels $\mathbf{Y} = [y_1, y_2, ..., y_i, ..., y_N]$

$x_i, y_i \in \mathbb{R}$

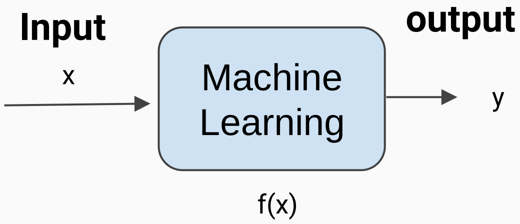
Goal: find a function $f(x) : \mathbb{R} \to \mathbb{R}$ that fits well with my dataset.

Optimization (Naive) Example

Example: Curve Fitting Problem



Training Set



Naive approach: try all the functions of the hypothesis space and take the one that better explains the training data.

Just for this toy example, let's assume that the hypothesis space is composed of the following functions:

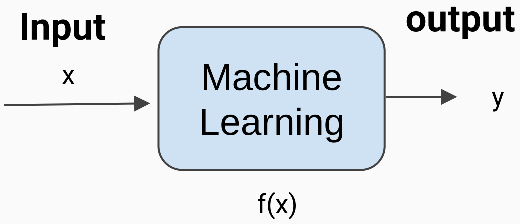
H

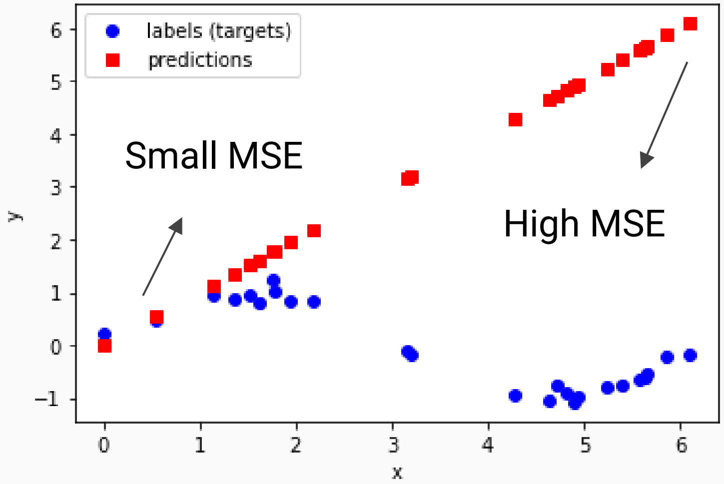
- Candidate 1: $f(x) = x$
- Candidate 2: $f(x) = e^x$
- Candidate 3: $f(x) = \sin(x)$
- Candidate 4: $f(x) = \cos(x)$

How well do they fit the training dataset?

Optimization (Naive) Example

Candidate 1 : $f(x) = x$





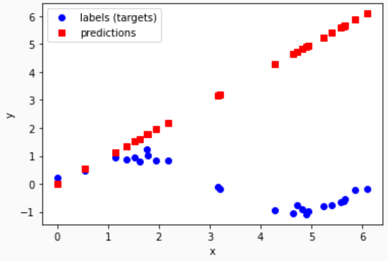
We can use the **Mean Squared Error** (MSE) as an objective:

$$MSE = \frac{1}{N}\sum_{i=0}^{N}(y_i - f(x_i))^2$$

- MSE is “*low*” when the predictions approach the targets and “*high*” when the predictions are far away from the targets.

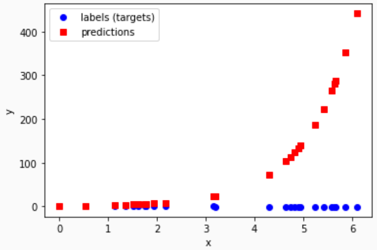
Optimization (Naive) Example

Candidate 1 : $f(x) = x$



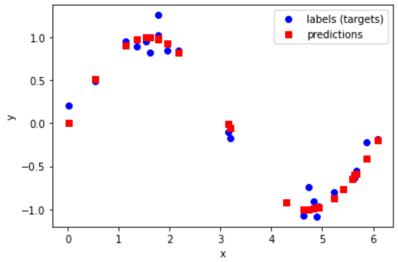
MSE = 19.49

Candidate 2 : $f(x) = e^x$



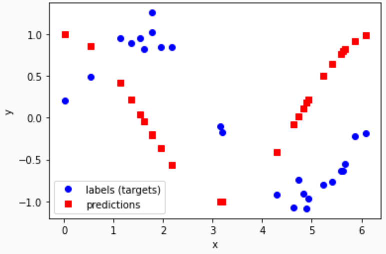
MSE = 28891.95

Candidate 3 : $f(x) = \sin(x)$



MSE = 0.013

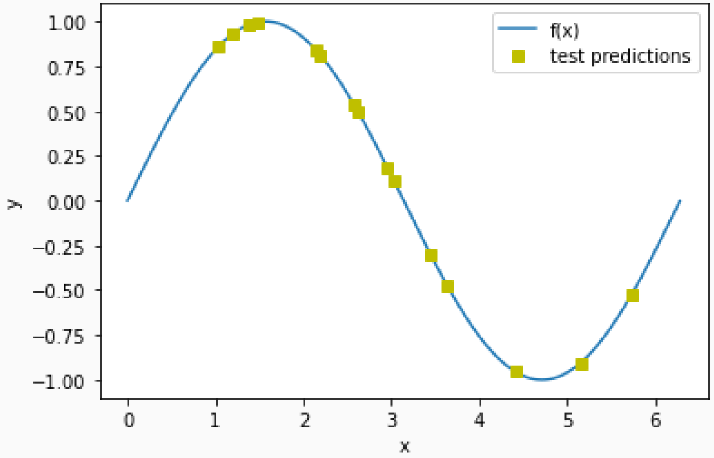
Candidate 4 : $f(x) = \cos(x)$



MSE = 1.18

Optimization (Naive) Example

- Our function f(x) = sin(x) **fits well** with the **training data**.
- However, we are more interested to see how well this mapping works on data never seen before (**generalization**).
- Let's thus compute the MSE on the **test set**:

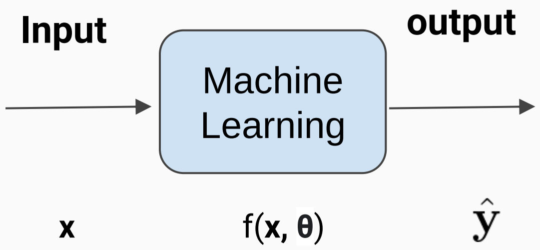


MSE = 0.010

- The function discovered during training fits well the test data: the **mean square error is close to zero**.
- This evidence suggests that we learned a function that **generalizes** well on new data points.

Parameter Optimization

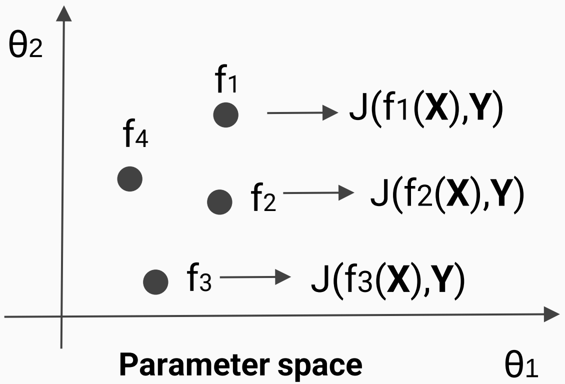
The function implemented by a machine learning algorithm often depends on some **parameters θ**



$$\boxed{\hat{\mathbf{y}} = f(\mathbf{x}, \boldsymbol{\theta})} \quad \mathbf{x} \in \mathbb{R}^D, \hat{\mathbf{y}} \in \mathbb{R}^K \qquad f : \mathbb{R}^D \to \mathbb{R}^K$$

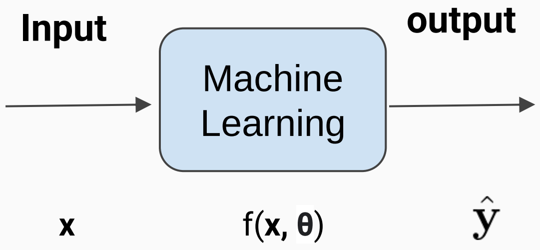
$$\boldsymbol{\theta} = [\theta_1, ..., \theta_P]^T \quad \boldsymbol{\theta} \in \mathbb{R}^P$$

- For each **parameter** configuration, the machine learning model implements a **different function**.



- In this case, the hypothesis space is equivalent to the **parameter space**.
- For every point of the parameter space, we can compute the **objective function** using the training dataset.
- During training, we **explore** the **parameter space** until we found a function that explains well the data.

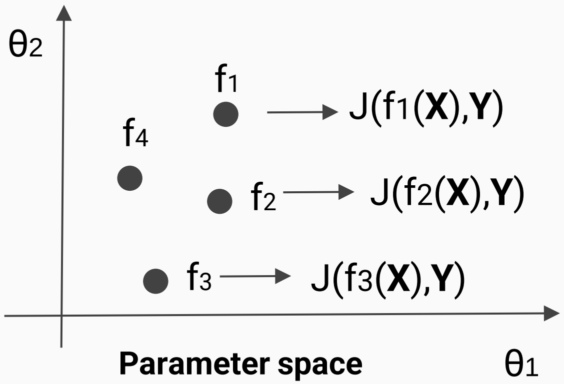
Parameter Optimization



$$\boxed{\hat{\mathbf{y}} = f(\mathbf{x}, \boldsymbol{\theta})} \quad \mathbf{x} \in \mathbb{R}^D, \hat{\mathbf{y}} \in \mathbb{R}^K \qquad f : \mathbb{R}^D \rightarrow \mathbb{R}^K$$

$$\boldsymbol{\theta} = [\theta_1, ..., \theta_P]^T \quad \boldsymbol{\theta} \in \mathbb{R}^P$$

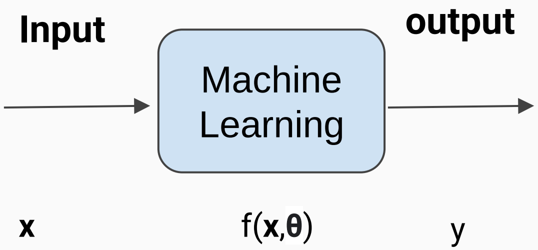
When we have parameters, training is about solving the following optimization problem:



$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta} \in \mathbb{R}^P}{\operatorname{argmin}} \, J\big(\mathbf{Y}, f(\mathbf{X}, \boldsymbol{\theta})\big)$$

Parameter Optimization (Vanilla Example)

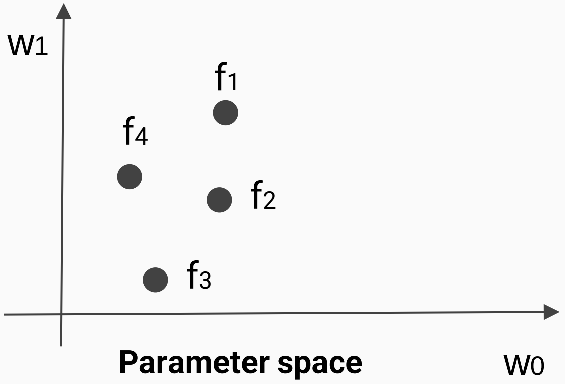
For instance, our machine learning model can implement **linear functions**

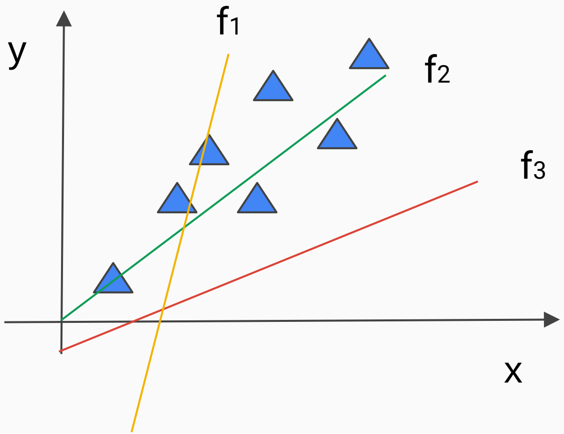


$$f(x, \mathbf{w}) = w_0 + w_1 x$$

$$\boldsymbol{\theta} = \mathbf{w} = [w_0, w_1]^{T}$$

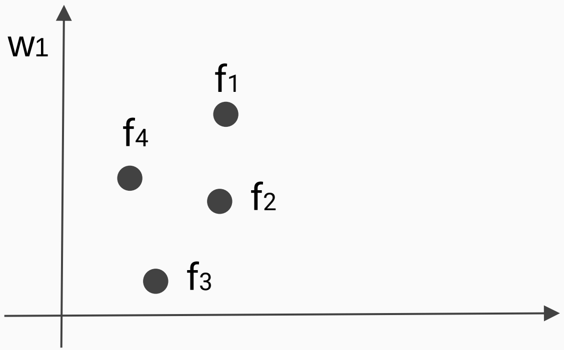
Depending on the values of w_0 and w_1, we implement different linear functions.

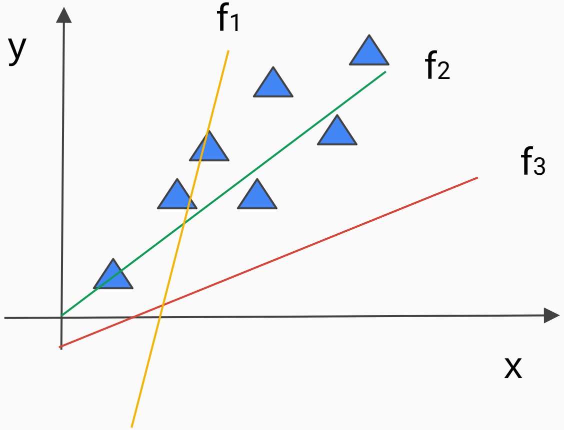




The goal of training is to find the parameter configuration that explains well the training data (triangle points).

Parameter Optimization (Vanilla Example)

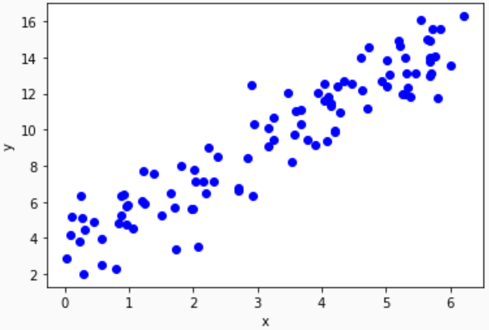


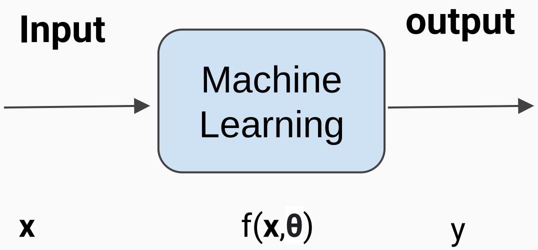


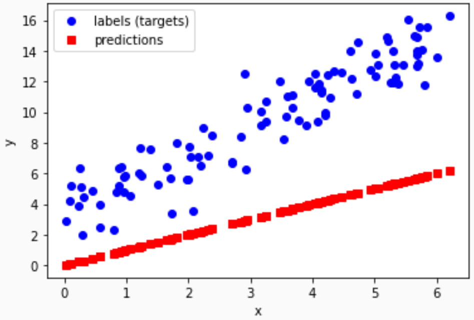
The parameters are **continuous**.

We have an **infinite** number of solutions! → We cannot try all!

Parameter Optimization (Vanilla Example)







We can start with **random parameters** and evaluate how the corresponding function performs on the training set.

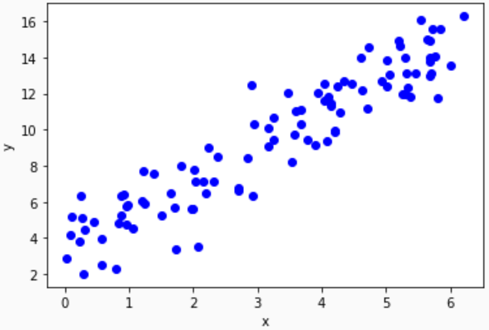
Let's start for instance with $w_0=0$ and $w_1=1$.

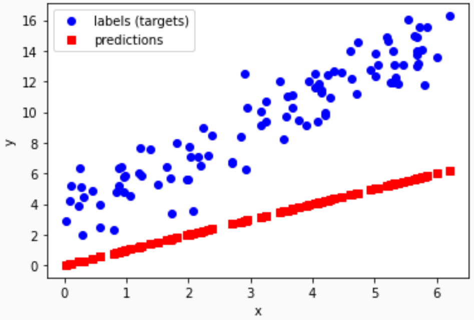
MSE = 40.60

$w_0=0$, $w_1=1$

That's pretty high. The current function does not explain well the training data.

Parameter Optimization (Vanilla Example)





MSE = 40.60

$w_0=0$, $w_1=1$

$$f(x, \mathbf{w}) = w_0 + w_1 x$$

- Let's now do a little step forward and backward for all the parameters, and let's monitor how the performance changes:

$$MSE(\mathbf{Y}, f(\mathbf{X}, \boxed{w_0+0.05}, \boxed{w_1+0.05})) = \boxed{19.28}$$

$$MSE(\mathbf{Y}, f(\mathbf{X}, w_0+0.05, w_1-0.05)) = 62.17$$

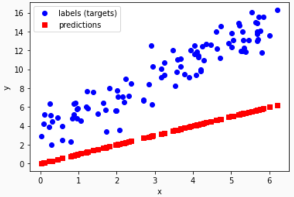
$$MSE(\mathbf{Y}, f(\mathbf{X}, w_0-0.05, w_1+0.05)) = 28.52$$

$$MSE(\mathbf{Y}, f(\mathbf{X}, w_0-0.05, w_1-0.05)) = 77.83$$

Ok, the best MSE is observed when increasing a bit both the parameters. Let's do a step in this direction!

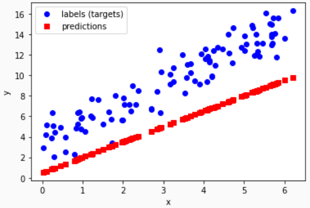
Parameter Optimization (Vanilla Example)

- We now have a slightly better function, but we are still unhappy.
- To further improve it, we can repeat this game **multiple times**.



MSE = 40.60
$w_0=0$, $w_1=1$

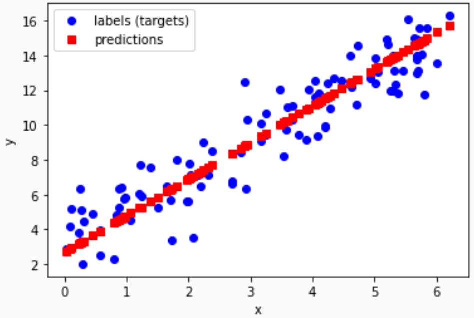
Epoch 0



MSE = 19.28
$w_0=0.05$, $w_1=1.05$

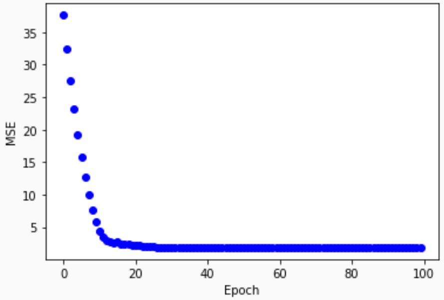
Epoch 1

……………



MSE = 1.92

$w_0=2.7$, $w_1=2.1$



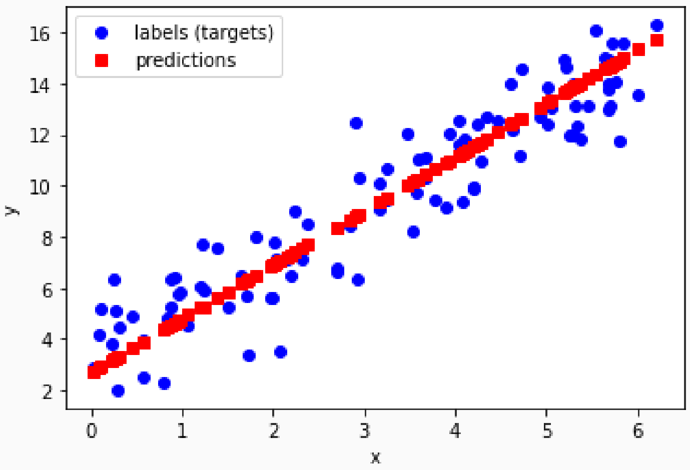
- The MSE decreases fast in the first epochs and then converges to a value close to 0.

Parameter Optimization (Vanilla Example)

We found a function that matches reasonably well with the **training data**.

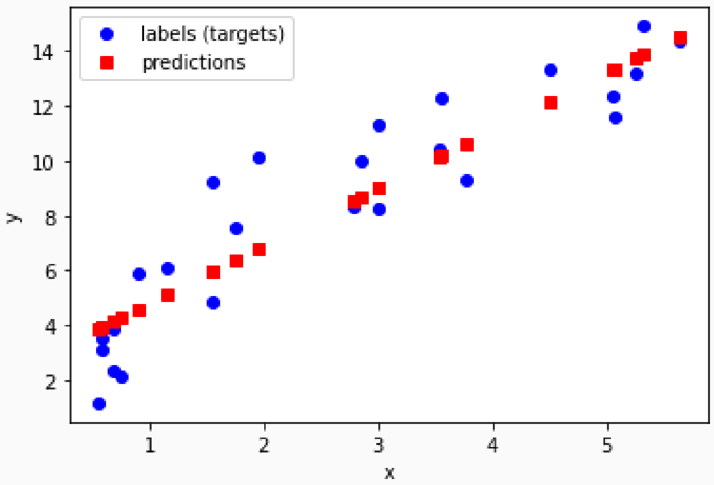
But what about the performance of the **test set**?

Training



MSE = 1.92

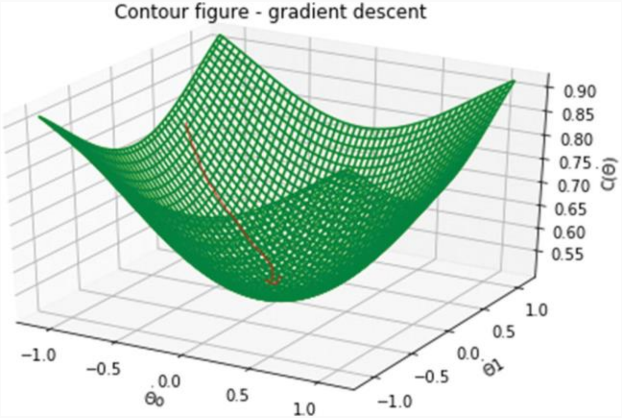
Test



MSE = 2.52

Parameter Optimization (Vanilla Example)

- We have seen a simple way to train a linear regressor.
- We trained it by applying **small variations** to our parameters and choosing the parameter configuration that maximized the MSE.
- As we will see in the next lecture, this "**little step**" in the direction that optimizes the cost function is what we will call "***gradient***".



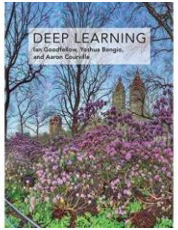
Additional Material

- Check out the tutorials (with google Colab) on:

  [Mean_squared_error.ipynb]() colab

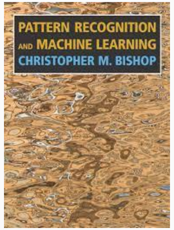
  [Categorical_cross_entropy.ipynb]() colab

  [Linear_Regression_Example.ipynb]() colab

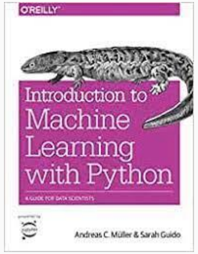


Chapter 2: Linear Algebra

Chapter 5: Machine Learning Basics



Introduction (pages 1-11)



Introduction (pages 1-27)

Lab Session

- During the weekly lab session, we will do:

Tutorial on Python

Tutorial on NumPy

Python and NumPy exercises

1st Lab Assignment Deadline:
Sunday 11:59PM, September 17th, 2023
(Submission from Moodle)