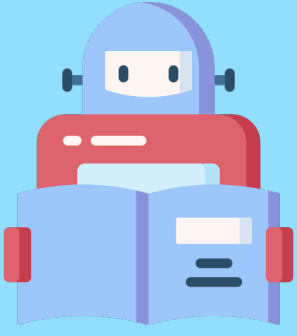
Mathematics for Machine Learning



Special Mathematics Lecture
Nagoya University, Fall 2020

Lecture 0
Course information

https://www.henrikbachmann.com/mml_2020.html

Who is teaching?

Lecturer

Henrik Bachmann

www.henrikbachmann.com

henrik.bachmann@math.nagoya-u.ac.jp

Teaching Assistant

Xuanrui Qi (Ray)

www.xuanruiqi.com

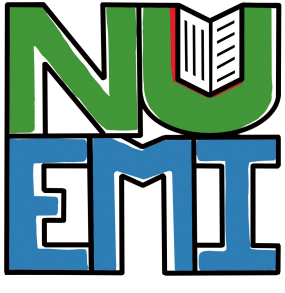
For who is this lecture?

Everyone! 皆様!

- G30 Students of any year who want to learn something outside of the standard curriculum.
- Japanese students who want to attend english lectures.
- Anyone who wants to learn something about mathematics in machine learning.

Basic knowledge in Linear Algebra and Calculus is helpful
(and sometimes necessary to understand everything in detail)

If you are **Japanese student**, please register at the NUEMI project for this lecture:



http://labguide.bio.nagoya-u.ac.jp/NUEMI/registration/

Free extra support & Study sessions

Study sessions

- An additional Zoom session each week
- Organized by students of this lecture
- Without the annoying professor
- Talk again about the content of the lecture
- Will start next week, probably a day after the second lecture.

The day and time is not fixed yet. It will get announced in Discord.

Tomoya Tatsuno ☺ .. 日本語でお願いします

What is this course? References

- This course is a “Special Mathematics Lecture”.
- Can give 2 credit points for ”Optional Courses”.
- Does not count for your final grade.
- You need to be registered in the NU Portal if you want these credits.

“Just for fun” Lecture

Learn together. Try to participate. Suggest topics!

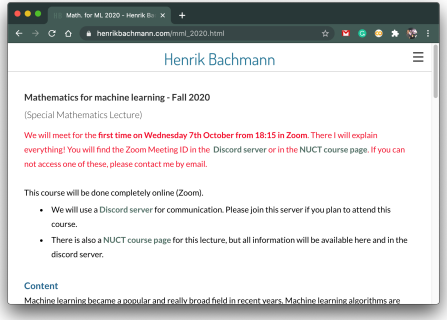
- The beginning of this course is be based on “CS229: Machine Learning” from Stanford university (http://cs229.stanford.edu/)
- Will provide more references during the semester on the homepage.

Goal of this lecture:

- Get an overview of algorithms used in machine learning
- Understand the mathematics used in these algorithms
- Learn how to implement (simple version of) them in python
- Learn to use new tools: Python, LaTeX, Colab, GitHub,…

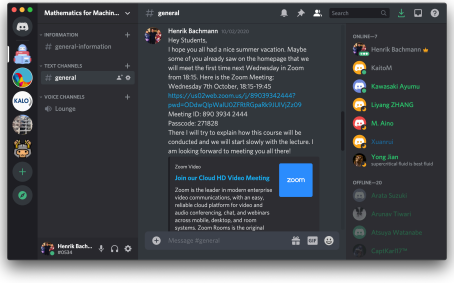
What where?

The lecture will be online every week on Wednesday 18:15.
The Zoom meeting will always be the same (see discord)



Homepage

https://www.henrikbachmann.com/mml_2020.html



Discord server

https://discord.gg/46tAxGX

- Please use it! Discuss! Share nice links/notes/videos
- Ask questions and answer questions of other students
- I will use it to announce everything

We will not use the NUCT course page

Assignments

Your grade will be based on programming & math assignments.

We will use GitHub Classroom (https://classroom.github.com/)

What you should to do first: (if you want to do the assignments)

- Create a GitHub account (www.github.com)
- Send me (by mail or private message in discord) the following information:

FAMILYNAME Firstname, GitHub Account name, Email

- Do the test “Test Assignment”. Link will be posted in Discord & Homepage.
- Play around with python & latex in Google Colab.

Content of the assignments

Don’t worry if you have no experience at all in programming!

Don’t worry if you have a lot of experience in programming!

- We will try to do a mix of mathematics exercises (using LaTeX) and programming exercises (Python)
- Try to make basic programming exercises, which can be made fancy by the more experienced students, but which can also be solved by students with not so much experience.

I did not decide on an exact grading model, but grading will not be strict!

Zoom & Questions

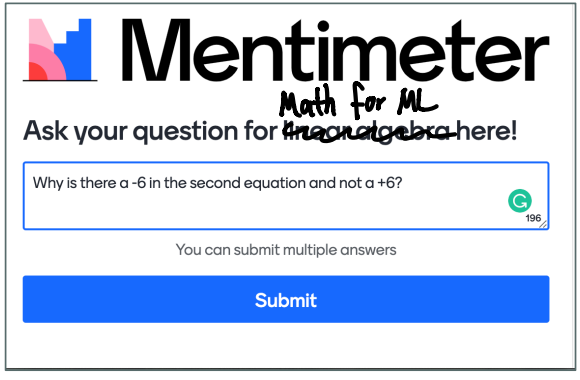
What makes professors happy:

- Seeing the faces of the students during the lecture.
- Getting a lot of questions.

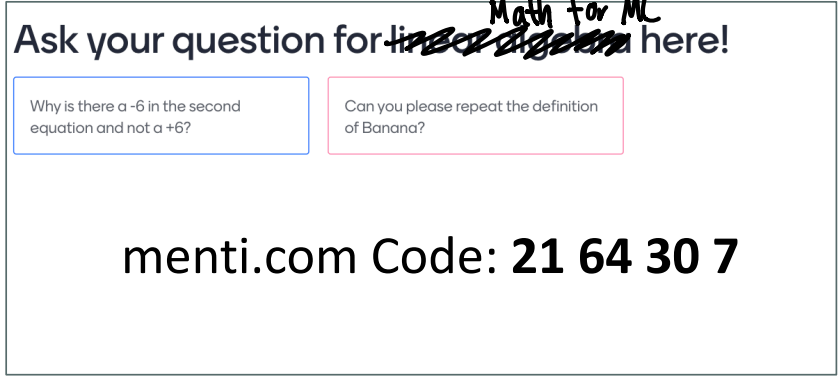
Please ask your questions!

Asking them directly in Zoom by audio is the best!

But I will also offer the possibility of asking anonymous questions at www.menti.com during the Zoom session.

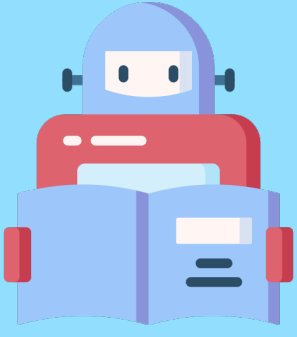


What you see



What I see

Mathematics for Machine Learning



Special Mathematics Lecture
Nagoya University, Fall 2020

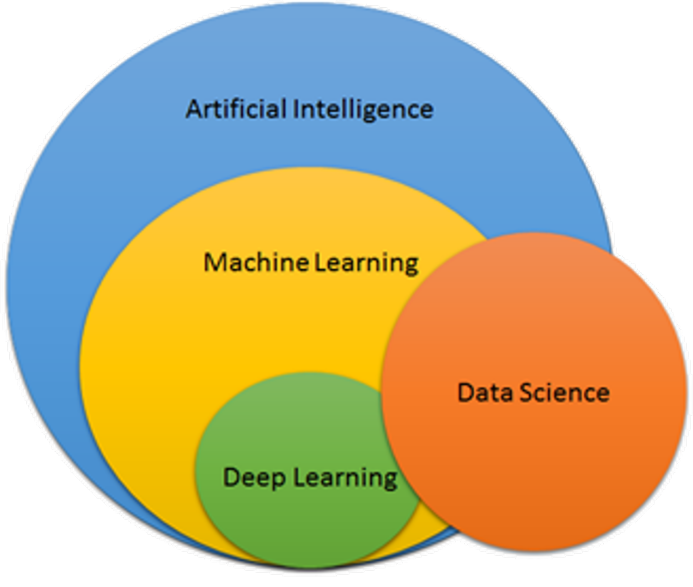
Lecture 1
Introduction & Linear regression

https://www.henrikbachmann.com/mml_2020.html

What is machine learning ?

Common definition:

Machine Learning is the study of computer algorithms that improve automatically through experience



Found its way in a lot of daily life applications, e.g.

- Self driving/flying cars/trains/planes/rockets
- Computer games
- Image & Video & Sound processing
- Speech recognition (Siri, Alexa,…)
- Translation (Google translate)
- Weather forecast
- Data analysis
- Email spam
- Robotics
- and much much more…

Not machine learning: “Classical machines / programs”

Programmer understands the problem and can implement an explicit algorithm, which turns an input into an output.

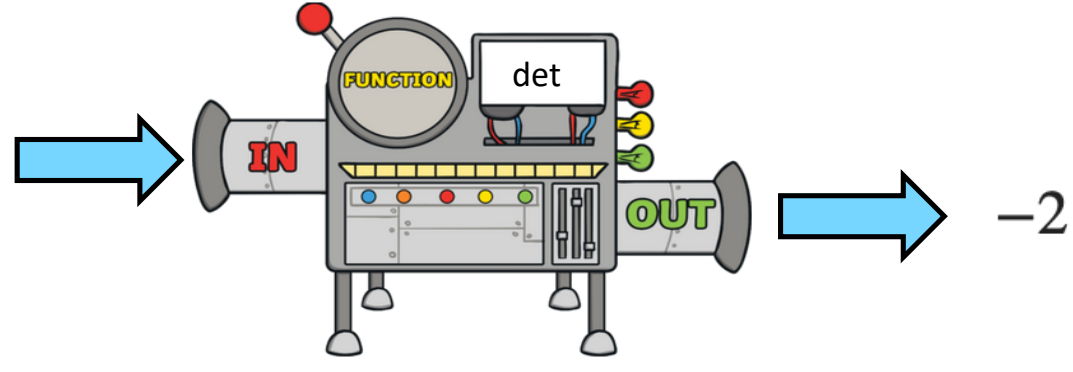
```python
def determinant(A):
    total=0
    indices = list(range(len(A)))
    if len(A) == 2 and len(A[0]) == 2:
        val = A[0][0] * A[1][1] - A[1][0] * A[0][1]
        return val
    for fc in indices:
        As = A
        As = As[1:]
        height = len(As)
        for i in range(height):
            As[i] = As[i][0:fc] + As[i][fc+1:]
        sign = (-1) ** (fc % 2)
        sub_det = determinant(As)
        total += sign * A[0][fc] * sub_det
    return total
```

```python
determinant([[1,2],[3,4]])
```

```
-2
```

Example: Determinant of a matrix

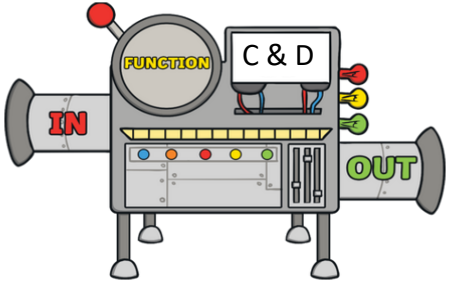
$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$



$$-2$$

What do you see?

Want: Cat & Dog detection machine



Cat & Dog detector

“This is a dog!”

Smart person

Of course, I know how cats and dogs look like. But I do not know how to describe it with a formula or algorithm!

How do we know?

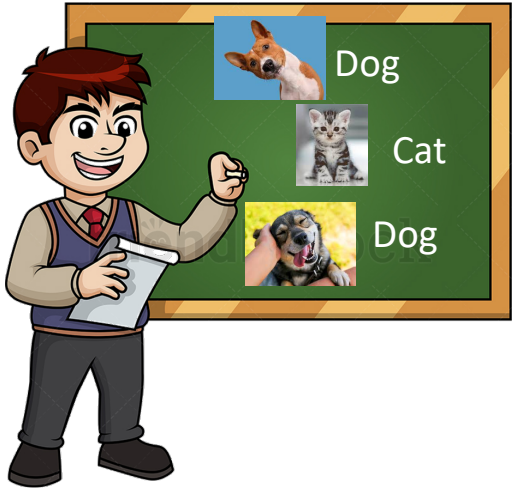
This is a dog because…

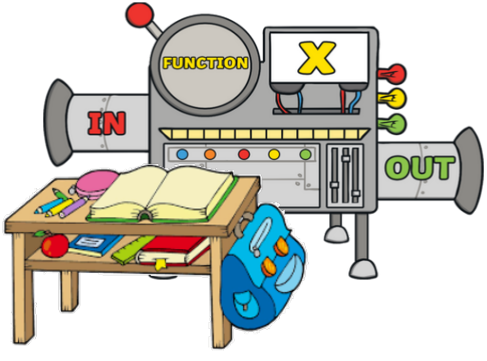
- I saw a lot of dogs & cats in my life.
- Whenever I saw a cat I was told by my parents „This is a cat!“. When I saw a dog they told me “This is a dog.”.
- I **learned** to distinguish between dogs and cats.
- But it is hard for me to explain the exact difference.

Machine learning

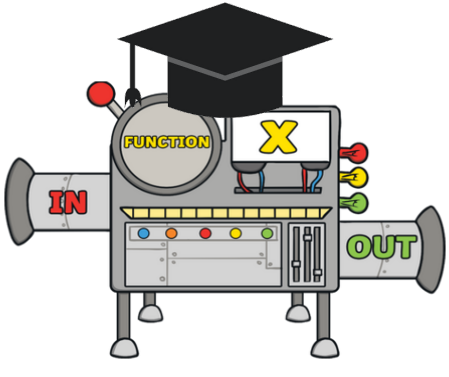
We can create machines/programs which we can teach

Training





After Training

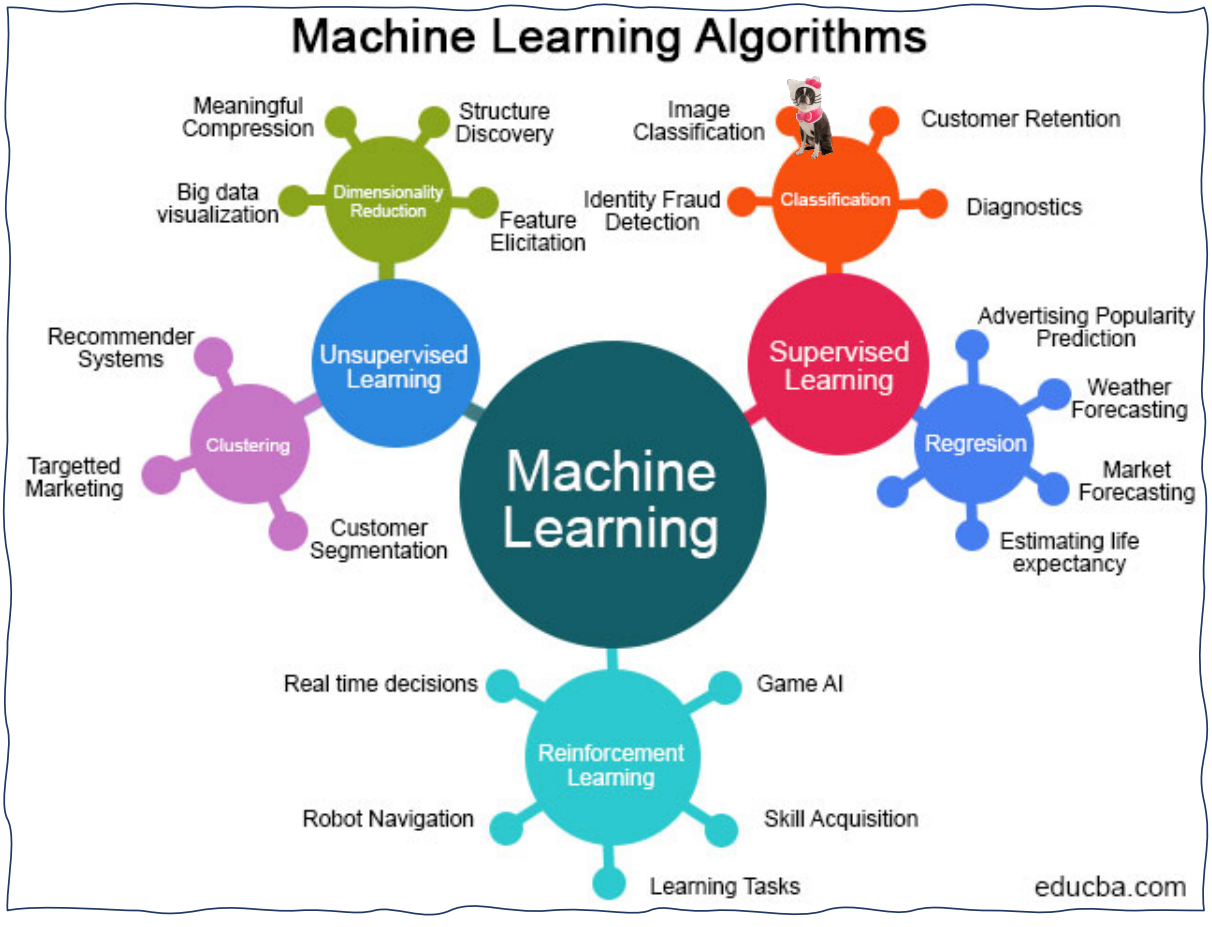


Cat & Dog detector

Dog!

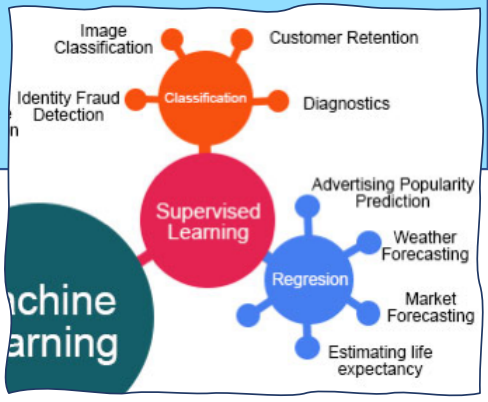
… with 95% certainty!

Machine learning overview



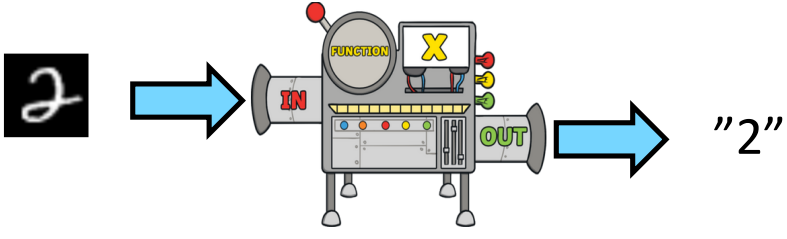
https://developers.google.com/machine-learning/glossary

Supervised learning

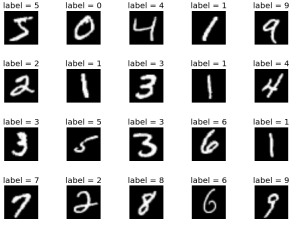


Classification (discrete output)

Output = category (e.g. “dog”, “cat” or “1”,”2”,…,”9”)



”2”

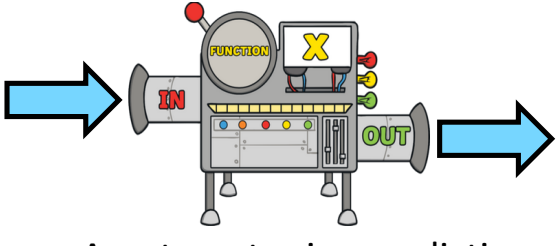


Handwriting recognition

Regression (continuous output)

Output = real number (like “price”, “weight”,…)

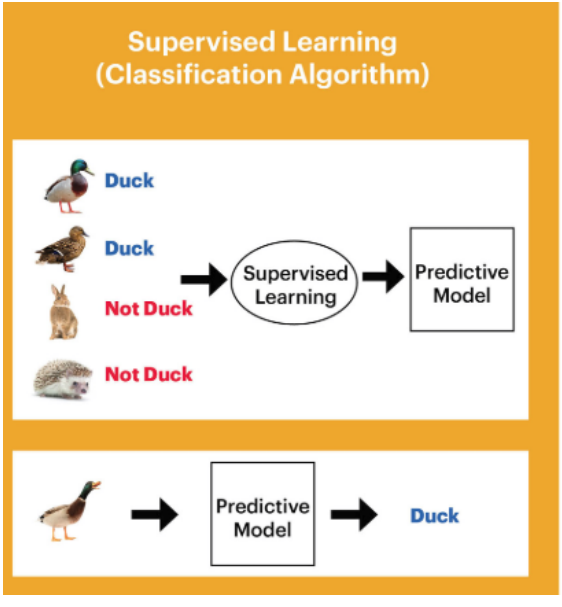
Size: $70m^2$
Rooms: 3

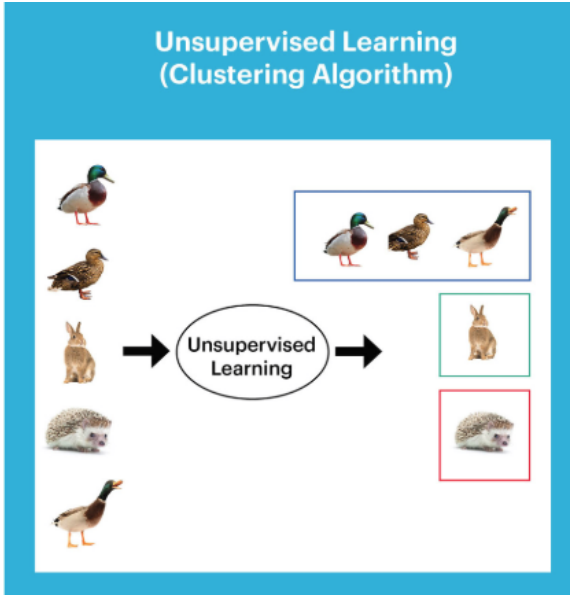


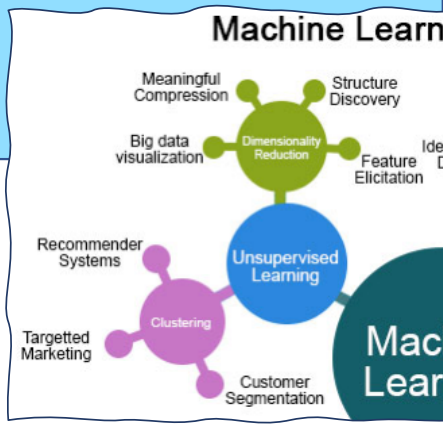
90.000 Yen

Apartment price prediction

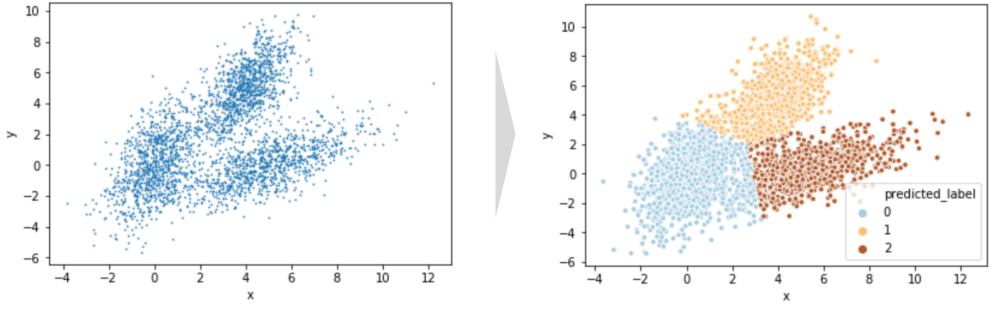
Unsupervised learning



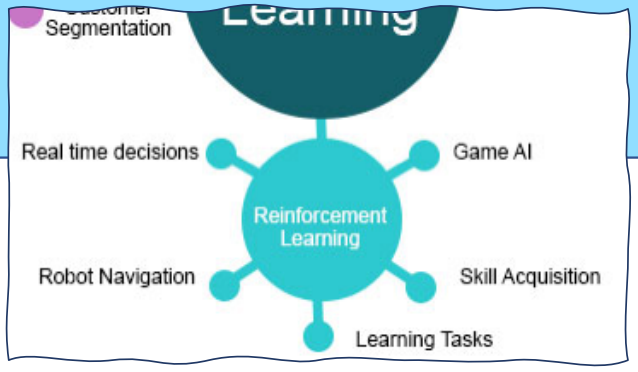


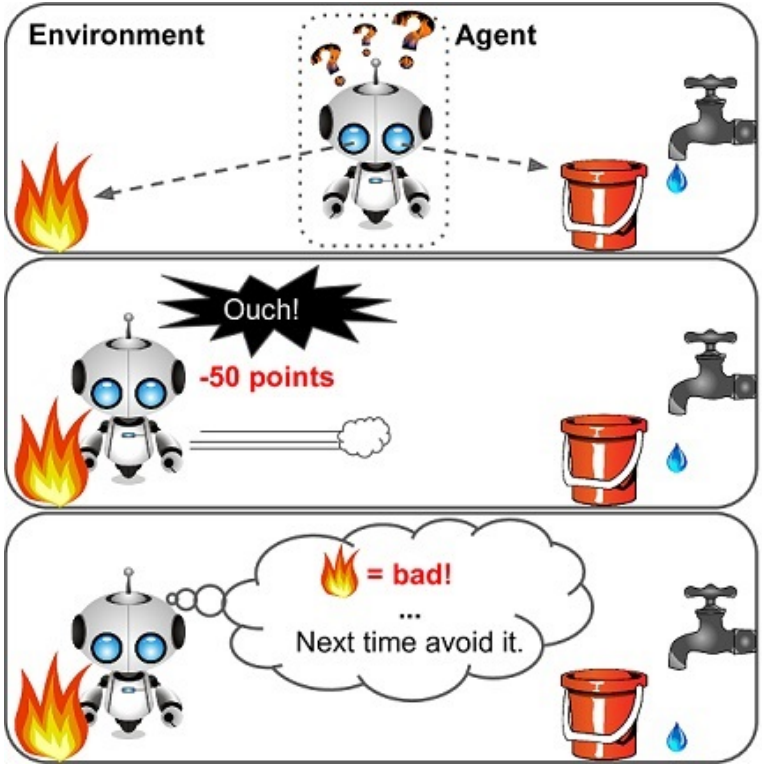


Before k-means clustering After



Reinforced learning





1. Observe
2. Select action using policy
3. Action!
4. Get reward or penalty
5. Update policy (learning step)
6. Iterate until an optimal policy is found

Examples of AI in Chess, Go & Computer games

- (1996: Deep blue (not machine learning))
- 2016: Alpha Go
- 2017: Alpha Zero (plays chess, shōgi & go)
- 2019: AlphaStar (Starcraft 2)

Reinforced learning

1996: Garri Kasparow (Chess world champion) loses against Deep blue.

2016: Lee Sedol (one of the world best Go Players) loses against Alpha Go.

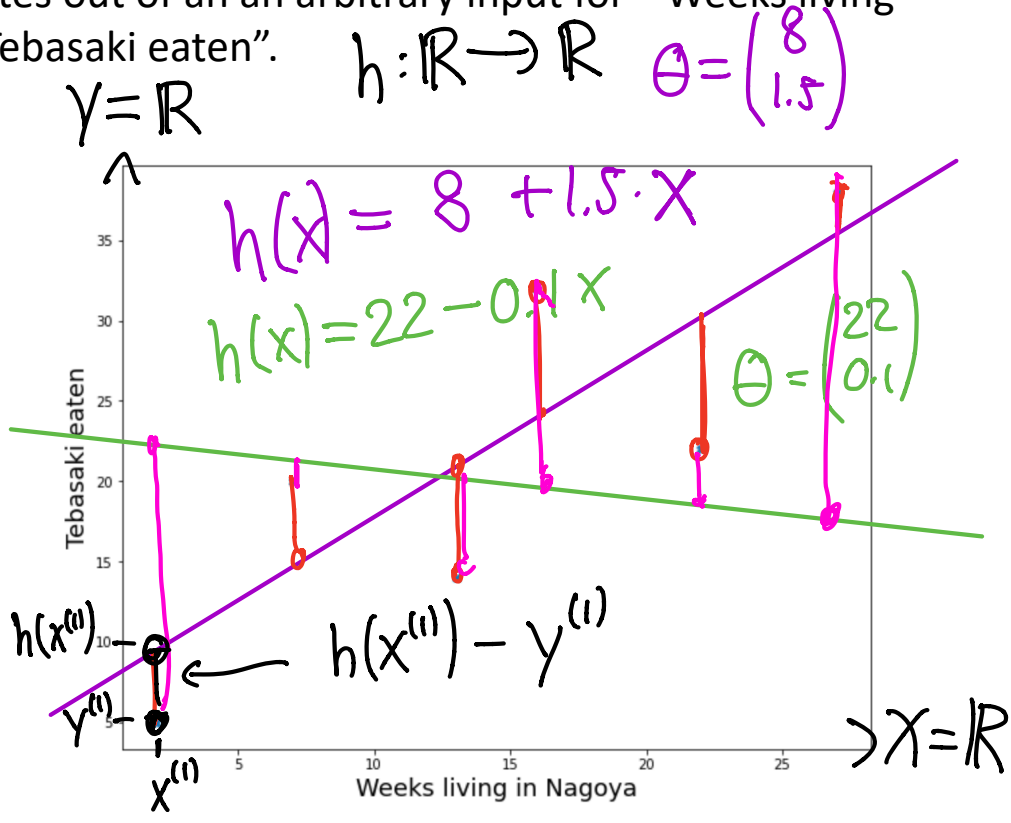
2019: AlphaStar beats Starcraft 2 pro gamers.

1 Supervised learning

Let us start!

Have: Some data of ”Weeks living in Nagoya” and “Tebasaki eaten”.

Want: A functions, which creates out of an an arbitrary input for “Weeks living in Nagoya” a prediction for “Tebasaki eaten”.



1 Supervised learning

output

input

Training set

feature

label (target)

income

6

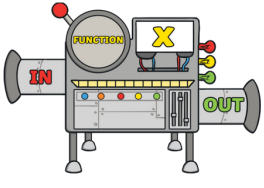
...

Training example

Weeks living in Nagoya	Tebasaki eaten
2	5
7	20
...	....

Learning algorithm

Weeks living in Nagoya



Predicted number of eaten Tebasaki

hypothesis

1 Supervised learning – Notation

- Input values (Feature space): $\mathcal{X}$
- Output value (Label space): $\mathcal{Y}$
- Trainings example: $(x, y) \in \mathcal{X} \times \mathcal{Y}$.

Dog & cat example:
$\mathcal{X} = \{$ pictures of cats & dogs $\}$
$\mathcal{Y} = \{$"cat", "dog"$\}$.

- Trainings set (with n training examples): $\mathcal{T} = \left((x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})\right) \in (\mathcal{X} \times \mathcal{Y})^n$.
- hypothesis: A function $h : \mathcal{X} \to \mathcal{Y}$. ← goal
- Learning algorithm: An algorithm to create a hypothesis h out of a trainings set $\mathcal{T}$.

Tebasaki example: $\mathcal{X} = \mathcal{Y} = \mathbb{N}$

$\mathcal{T} = \left((2,5), (7,20), \ldots\right)$

$\mathcal{X} = \mathbb{R}^d \ni x^{(1)} = \begin{pmatrix} x_1^{(1)} \\ x_2^{(1)} \\ \vdots \\ x_d^{(1)} \end{pmatrix}$

1 Supervised learning – Linear Regression

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{pmatrix} \in \mathbb{R}^d$$

$Y = \mathbb{R}$

Learning Algorithm: Linear Regression

(idea)

Let $\mathcal{X} = \mathbb{R}^d$, i.e. we have d features, and $\mathcal{Y} = \mathbb{R}$. As an Ansatz for the hypothesis we set

$$h_\theta(x) := \theta_0 + \theta_1 x_1 + \cdots + \theta_d x_d = \sum_{i=0}^{d} \theta_i x_i \,,$$

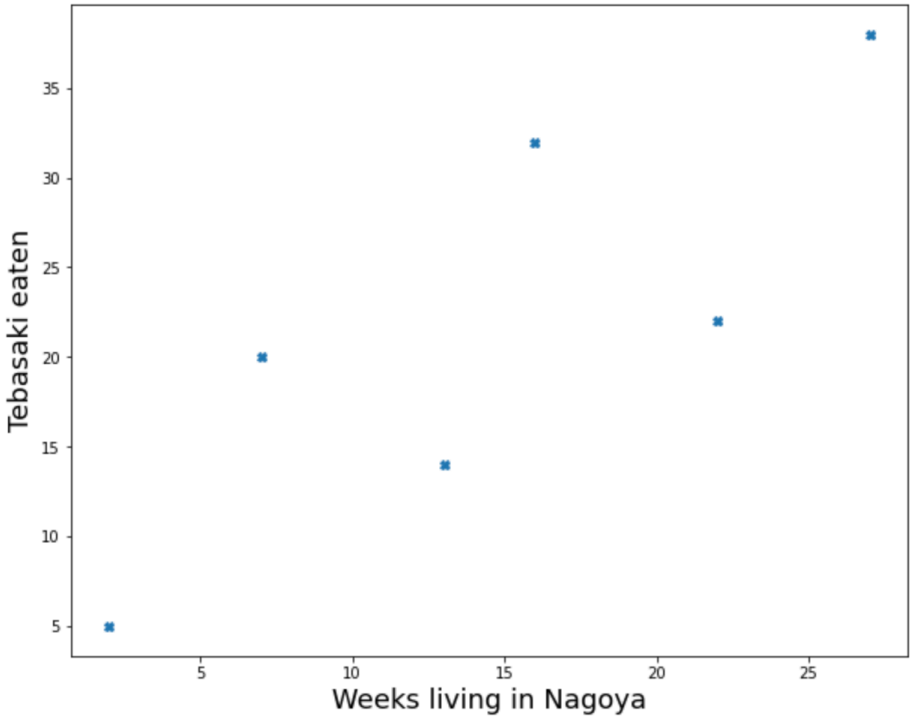
with **parameters (weights)** $\theta = (\theta_0, \theta_1, \ldots, \theta_d)^T \in \mathbb{R}^{d+1}$. In the second equation we set $x_0 := 1$.

(Tebasaki : $d=1$ just one feature: Weeks living in Nagoya)

$\theta = \begin{pmatrix} \theta_0 \\ \theta_1 \end{pmatrix}$ $\quad h_\theta(x) = \theta_0 + \theta_1 x$

Goal: Determine the "best "parameters for a given trainings set.

Tebasaki example:



1 Supervised learning – Linear Regression

Measure how good parameters are:

n = number of trainings examples

For a given training set $\mathcal{T} = \left((x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})\right)$ we define the **cost function** by

$$J(\theta) = \frac{1}{2} \sum_{j=1}^{n} (h_\theta(x^{(j)}) - y^{(j)})^2 .$$

The cost function is a function $J : \mathbb{R}^{d+1} \to \mathbb{R}$, which we want to minimize.

j	$x^{(j)}$	$y^{(j)}$
1	2	5
2	·	·
3		
4		
5		
6		

Goal rephrased: Minimize the cost function for a given trainings set.

Tebasaki $d=1$ $n=6$

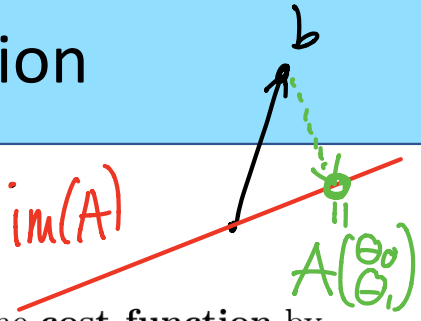
$h_\theta(x) = \theta_0 + \theta_1 x$

$J : \mathbb{R}^2 \to \mathbb{R}$

$\theta = \begin{pmatrix} \theta_0 \\ \theta_1 \end{pmatrix} \mapsto \frac{1}{2} \sum_{j=1}^{6} (\underbrace{\theta_0 + \theta_1 x^{(j)}}_{h_\theta(x^{(j)})} - y^{(j)})$

There are several different choices for cost functions. The above choice corresponds is the “least-squares cost function“.

1 Supervised learning – Linear Regression

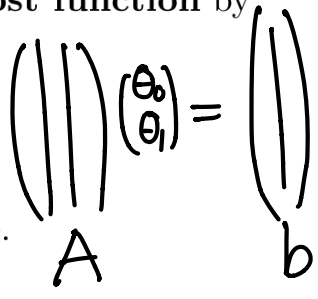


Measure how good parameters are:

For a given training set $\mathcal{T} = \big((x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})\big)$ we define the **cost function** by

$$J(\theta) = \frac{1}{2}\sum_{j=1}^{n}(h_\theta(x^{(j)}) - y^{(j)})^2 \,.$$

The cost function is a function $J : \mathbb{R}^{d+1} \to \mathbb{R}$, which we want to minimize.



Goal rephrased: Minimize the cost function for a given trainings set.

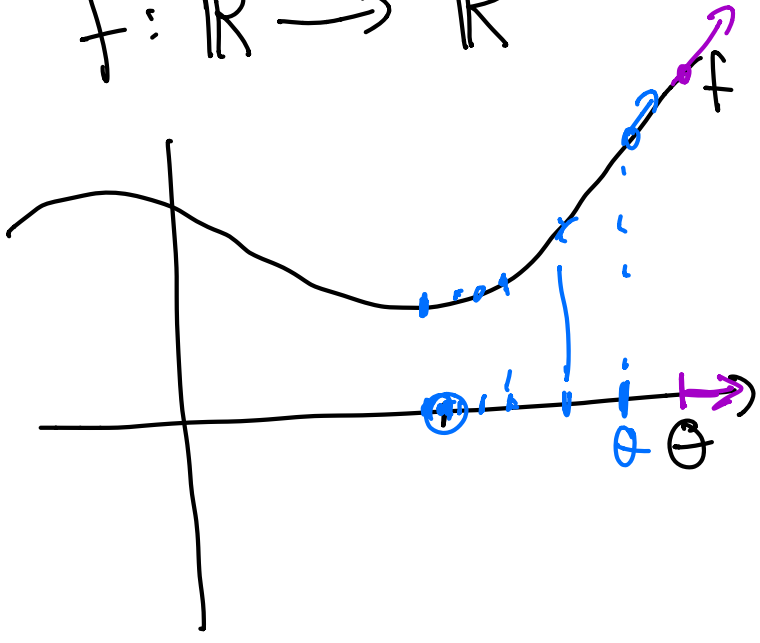
Linear Algebra I: There are explicit ways (orthogonal projection) to minimize the above cost function.

But we will use an algorithm which will also be used for other learning algorithms: **Gradient descent.**

1 Supervised learning – Linear Regression

Gradient descent main idea:

$f: \mathbb{R} \to \mathbb{R}$



Want to minimize a fct

$J: \mathbb{R}^{d+1} \to \mathbb{R}$

$$\theta := \theta - \alpha \cdot f'(\theta)$$

learning factor.

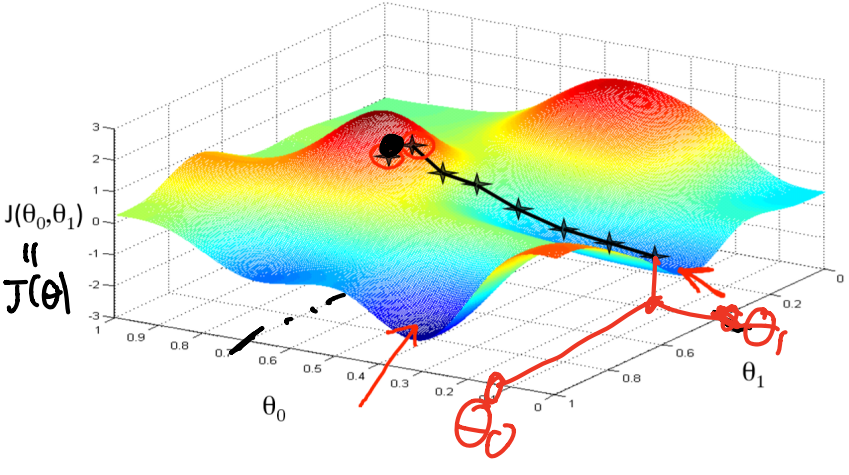
1 Supervised learning – Linear Regression

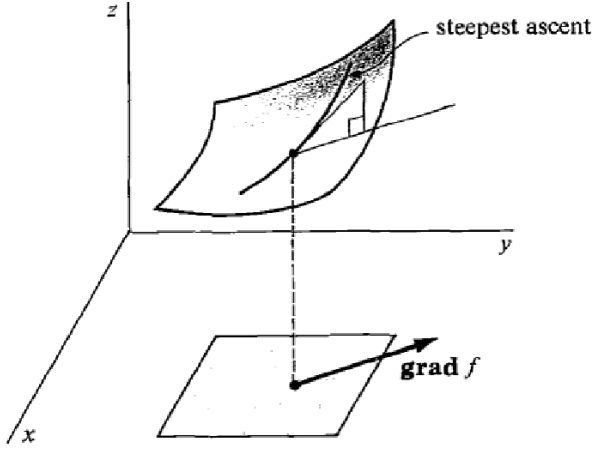
Gradient descent main idea:

$d = 1$ $\quad J : \mathbb{R}^2 \longrightarrow \mathbb{R}$

$\theta = \begin{pmatrix} \theta_0 \\ \theta_1 \end{pmatrix} \mapsto J(\theta)$

$$\nabla J = \begin{pmatrix} \frac{\partial}{\partial \theta_0} J \\ \frac{\partial}{\partial \theta_1} J \end{pmatrix}$$





Fact: The gradient shows in the direction of the steepest ascent

1 Supervised learning – Linear Regression

The **gradient of** J is defined by $\nabla J = \begin{pmatrix} \frac{\partial}{\partial \theta_0} J \\ \frac{d\partial}{\partial \theta_1} J \\ \vdots \\ \frac{\partial}{\partial \theta_d} J \end{pmatrix}$

Goal: Find $\theta \in \mathbb{R}^{d+1}$, such that $J(\theta)$ is minimal.

Gradient descent algorithm (rough version).

i) Start with a random starting value for the parameters, e.g. $\theta = 0 = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}$.

ii) Change the parameters in the opposite direction of the steepest ascent, i.e. opposite direction of the gradient. This means we want to subtract the gradient from the current parameters, weighted by a factor $\alpha \in \mathbb{R}$, the **learning rate**.

The new parameters θ are therefore given by:

$$\theta := \theta - \alpha \nabla J(\theta).$$

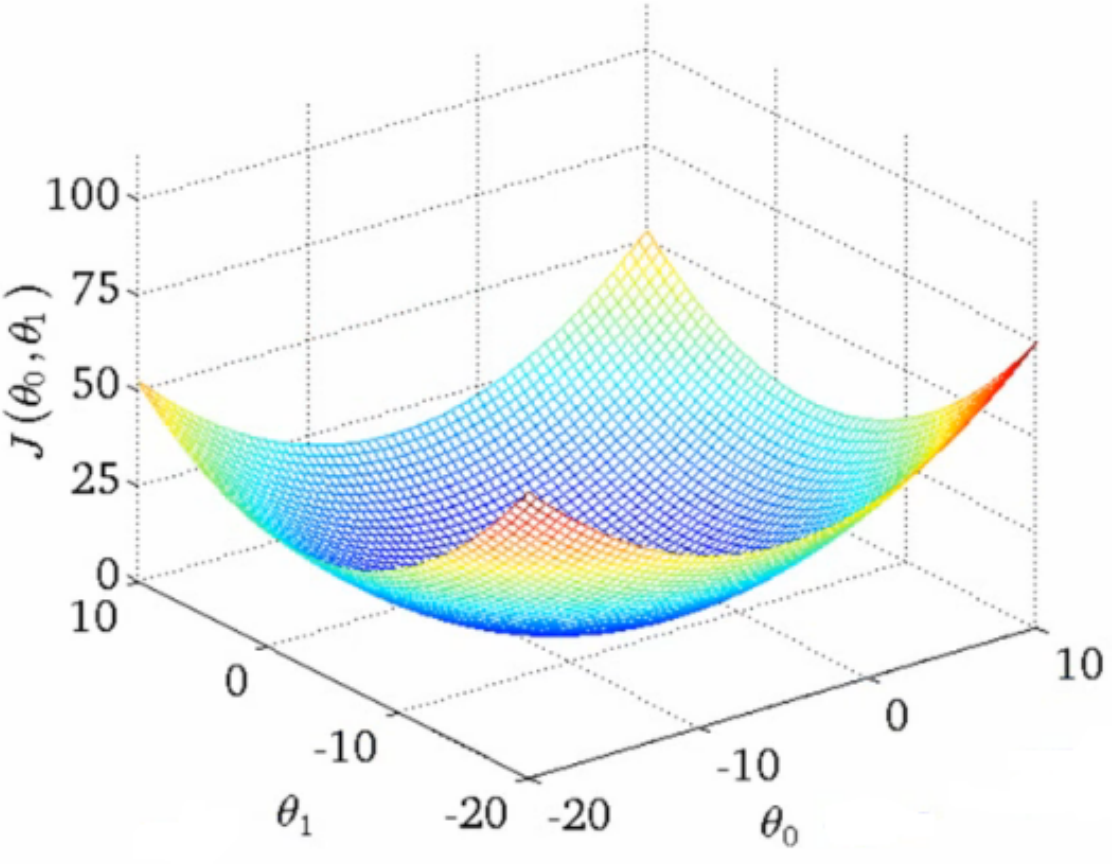
iii) Repeat step ii) until the value $J(\theta)$ does not change anymore.

1 Supervised learning – Linear Regression

Gradient descent: d=1 case

1 Supervised learning – Linear Regression

Gradient descent: d=1 case



1 Supervised learning – Linear Regression

Next time:

- Calculate the gradient for J in general.
- Do some Python examples.

Until then try to find out how to use Google Colab!

Plan for Assignment 1: Implementation gradient descent for linear regression. Details will come next week!