# Deep Network Developments

Lecture #1

Viktor Varga
Department of Artificial Intelligence, ELTE IK

Who should I contact if I have an administrative question?

Lecturers:

Imre Molnár
Boss

Viktor Varga
(just helping out)

The classic approach

The classic approach to solving problems using computers:

Writing an algorithm…

```java
void insertionSort(int[] ar) {
    for (int i=1; i < ar.length; i++) {
        int index = ar[i]; int j = i;
        while (j > 0 && ar[j-1] > index) {
            ar[j] = ar[j-1];
            j--;
        }
        ar[j] = index;
    }
}
```

The classic approach

Writing an algorithm might not be ideal when…

1. … we need to solve **many** slightly different **varieties** of the same task

The classic approach

Writing an algorithm might not be ideal when…

2. … an **algorithmic solution is unknown** or hard to compute

The classic approach

Writing an algorithm might not be ideal when…

3. … the task can **only be formulated by** showing **examples**

E.g., **"Is there a dog in the image?"**
   – **"Yes."**
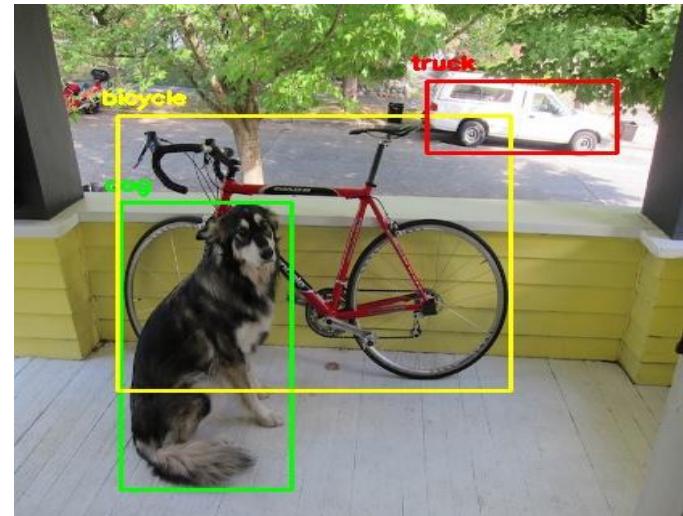
The classic approach

Writing an algorithm might not be ideal when…

3. … the task can **only be formulated by** showing **examples**

E.g., **"Is there a dog in the image?"**
       – **"Yes."**

How does a dog look like?
→ Impossible to describe accurately with rules.
Instead, we can show examples.
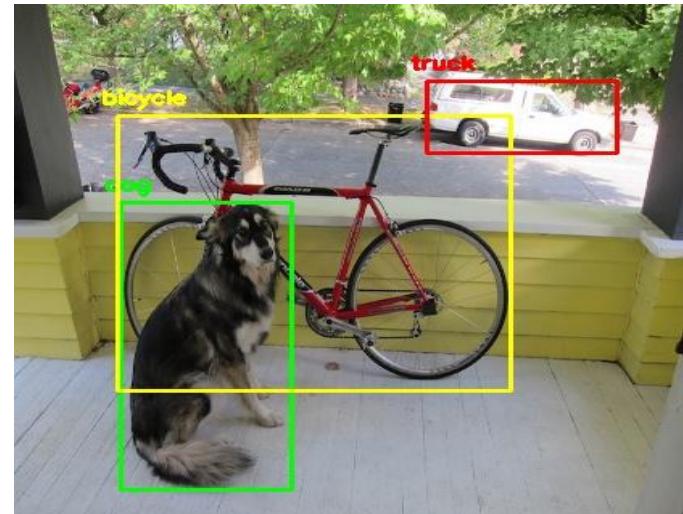
The essence of machine learning

We **learn to solve the problem** without knowing the specific algorithm for the solution.

The essence of machine learning

We **learn to solve the problem** without knowing the specific algorithm for the solution.

- We define a machine learning **model with parameters**.
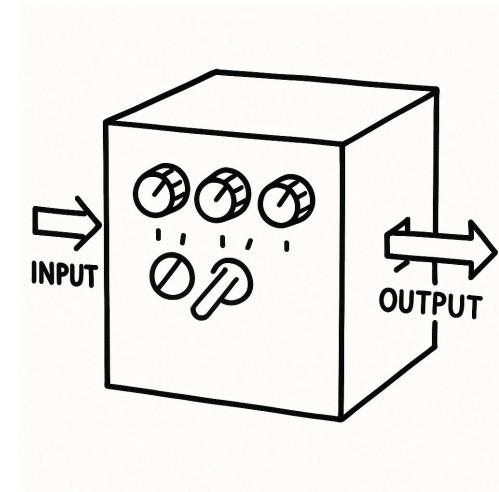
The essence of machine learning

We **learn to solve the problem** without knowing the specific algorithm for the solution.

- We define a machine learning **model with parameters**.
- We try to **find such parameters** that will make our model **solve the task** as **well** as possible.

The essence of machine learning

We **learn to solve the problem** without knowing the specific algorithm for the solution.

- We define a machine learning **model with parameters**.
- We try to **find such parameters** that will make our model **solve the task** as **well** as possible.
- We **define the task** and its correct solution **by showing examples**.

INPUT

OUTPUT

"a dog"

❌
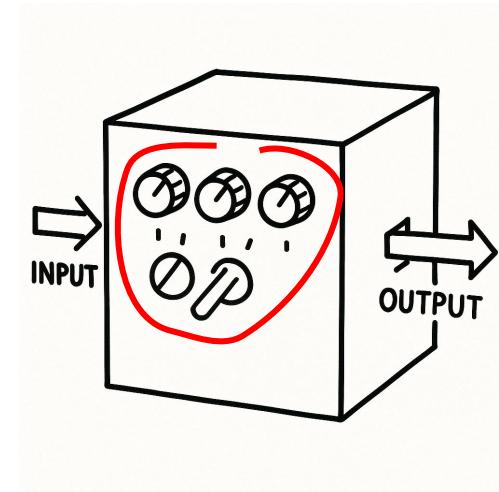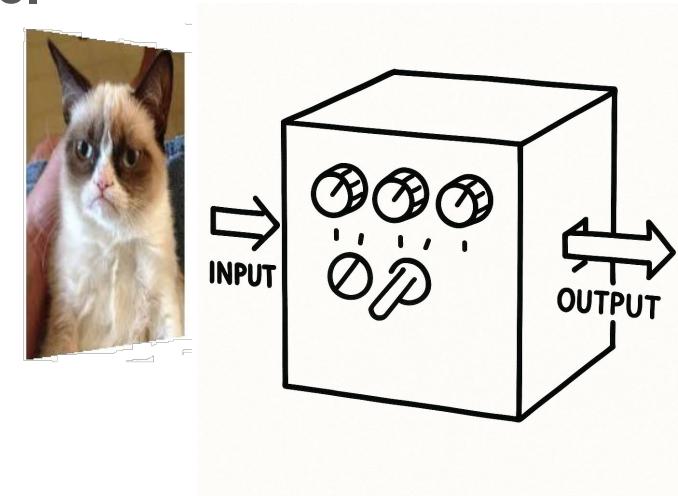
The essence of machine learning

We **learn to solve the problem** without knowing the specific algorithm for the solution.

- We define a machine learning **model with parameters**.
- We try to **find such parameters** that will make our model **solve the task** as **well** as possible.
- We **define the task** and its correct solution **by showing examples**.

The essence of machine learning

We **learn to solve the problem** without knowing the specific algorithm for the solution.

- We define a machine learning **model with parameters**.
- We try to **find such parameters** that will make our model **solve the task** as **well** as possible.
- We **define the task** and its correct solution **by showing examples**.

INPUT

OUTPUT

"a cat"
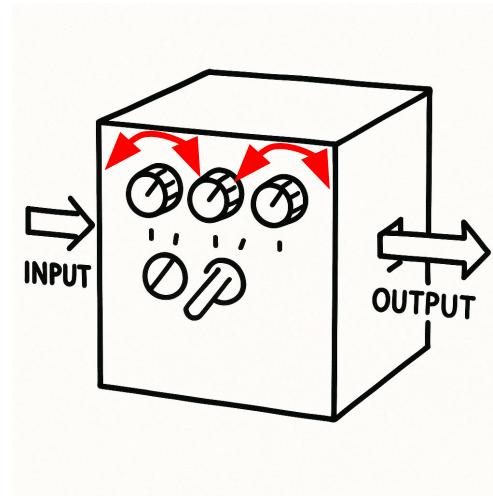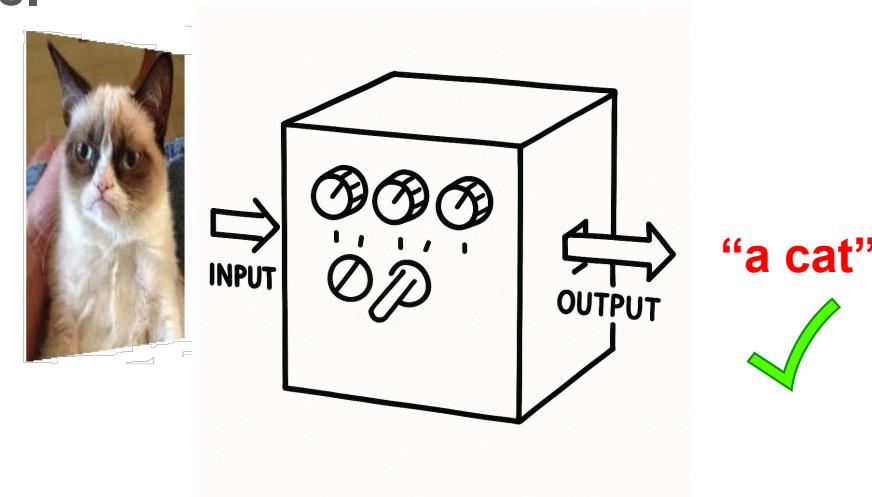
Tasks not suitable for machine learning

- If an **efficient solution** to the problem **is known** and can be easily algorithmized.
- If an **approximate result is not satisfactory**.

Examples:

- **Lossless compression** algorithms.
- **Sorting** algorithms.
- Most software products that are defined by a **complex set of rules**.

Machine learning

Three main groups of machine learning methods:

- **Supervised learning**
- Unsupervised learning
- Reinforcement learning

Supervised learning

Given: The training sample, a set of (input, label) pairs

$$\{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$$

$$x \in X \subset \mathbb{R}^n, \ y \in Y \subset \mathbb{R}^k$$

Task: The estimation of the label (the expected output) from the input

I.e., we search for a (hypothesis-)function h_θ, for which:

$$h_\theta(x) = \hat{y} \approx y$$

Supervised learning

Given: The training sample, a set of (input, label) pairs

$$\{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$$

$$x \in X \subset \mathbb{R}^n, \; y \in Y \subset \mathbb{R}^k$$

Task: The estimation of the label (the expected output) from the input

I.e., we search for a (hypothesis-)function h_θ, for which:

The machine learning model with parameters

$$h_\theta(x) = \hat{y} \approx y$$

- The **true label** is "y".
- Our model h_θ gives an **estimate** for the label: "y hat".

Supervised learning

Given: The training sample, a set of (input, label) pairs

$$\{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$$

$$x \in X \subset \mathbb{R}^n, \ y \in Y \subset \mathbb{R}^k$$
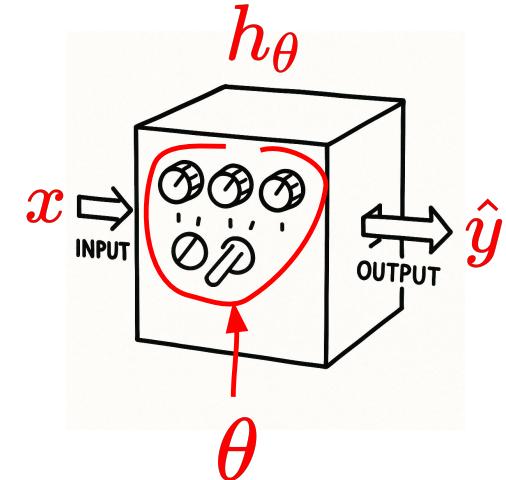
Goal: Find θ that makes $\hat{y}$ be as similar to y as possible!

Task: The estimation of the label (the expected output) from the input

I.e., we search for a (hypothesis-)function h_θ , for which:

$$h_\theta(x) = \hat{y} \approx y$$

Supervised learning

Example: Estimate the number of cars in a particular city given the population of that city.

x: The **population** of a particular city

y: The **number of cars** in a particular city

Cars and residents (Poland 2017)

$$x^{(j)}, y^{(j)} \in \mathbb{R}$$

Supervised learning

Example: Estimate the number of cars in a particular city given the population of that city.

x: The **population** of a
    particular city

y: The **number of cars** in a
    particular city

Cars and residents (Poland 2017)
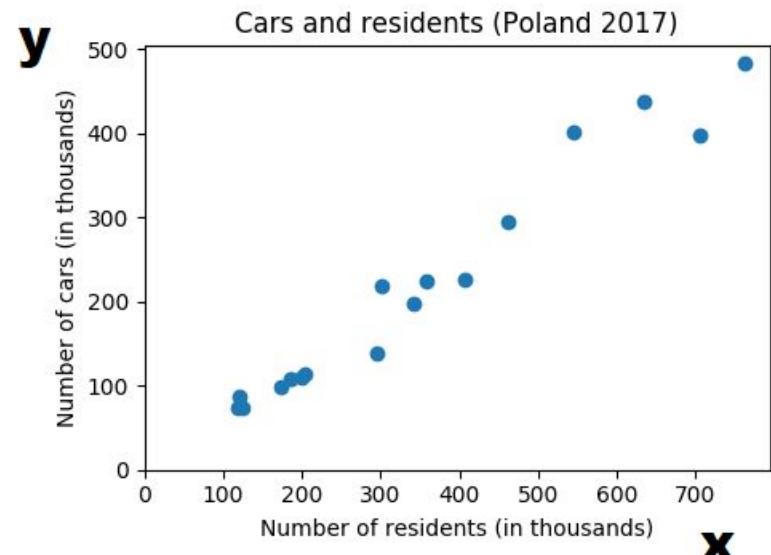
A single input variable and **a label:**
Our labeled examples can be interpreted
as points located in
a two-dimensional vector space (a plane).

$$x^{(j)}, y^{(j)} \in \mathbb{R}$$

Visualizing the data

Cars and residents (Poland 2017)
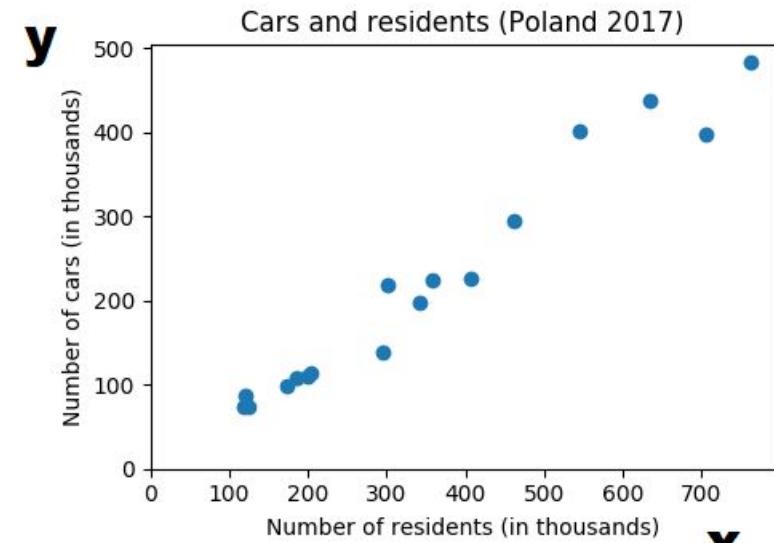
$$x^{(j)}, y^{(j)} \in \mathbb{R}$$
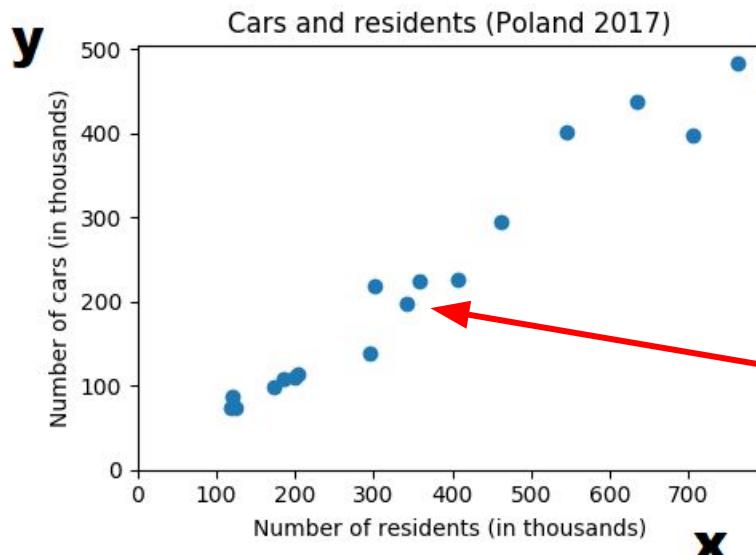
	the input variable (feature)	the label
	x (population, ×1000)	y (n. of cars, ×1000)
Warsaw (j = 1)	1760	910
Krakow (j = 2)	770	465
Lublin (j = 3)	340	198
...	...	...

Supervised learning

How should our model look like?
What type of hypothesis function are we looking for?

The number of cars is roughly proportional to the population of the city...

Supervised learning

What type of hypothesis function are we looking for?

A very simple (linear)
hypothesis function:

$$y \approx \hat{y} = h_\theta(x) = \theta x$$

**θ is the parameter of the hypothesis-
function:** in this case,
this is going to be the **slope of the line**.

Cars and residents (Poland 2017)

$\hat{y} = \theta x$ $\theta = ?$

Number of cars (in thousands)

Number of residents (in thousands)
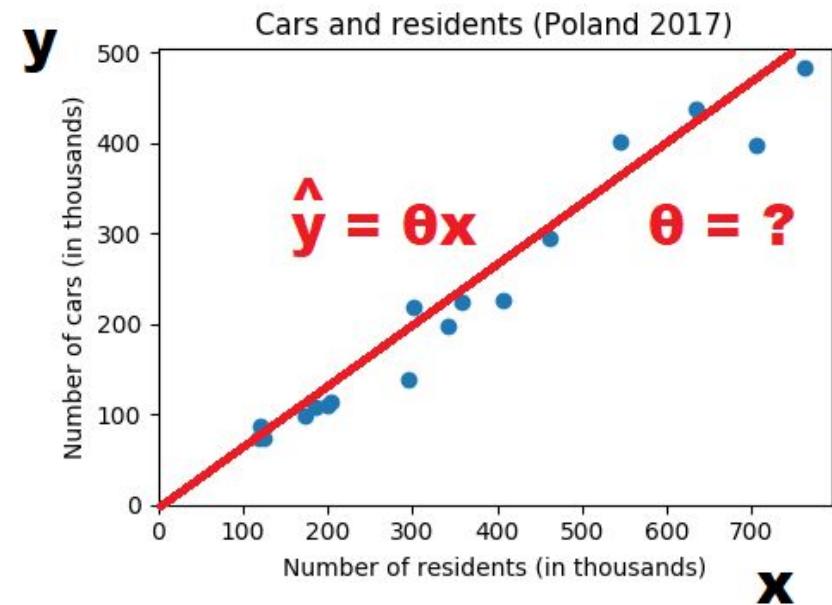
Supervised learning

What type of hypothesis function are we looking for?

A very simple (linear)
hypothesis function:

$$y \approx \hat{y} = h_\theta(x) = \theta x$$

θ is the parameter of the hypothesis-function: in this case,
this is going to be the **slope of the line**.

We are looking for a parameter θ such that h(x) closely approximates the true y labels. For example, the hypothesis function h(x) = 0.65*x fits this particular sample well, so a good parameter is θ = 0.65.

Cars and residents (Poland 2017)

$\hat{y} = \theta x$ θ = ?

Number of cars (in thousands)

Number of residents (in thousands)
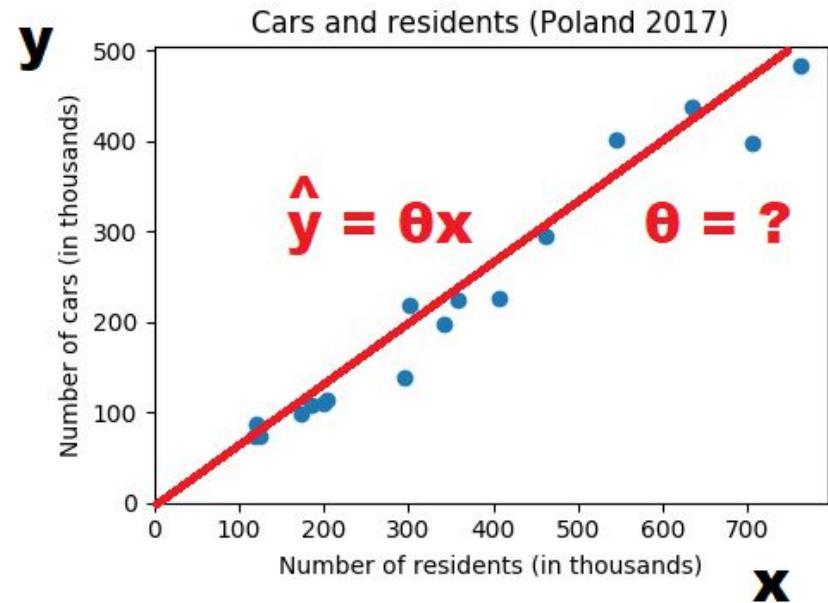
Supervised learning

Examples **do not necessarily consist of a single input variable and a single label...**

Supervised learning

Example: Estimate the age of the people in the pictures!

x: A portrait
(a fixed-resolution color image)

y: The age of the person in the image

$$x^{(j)} \in [0, 255]^{196608}$$

$$y^{(j)} \in \mathbb{R}$$

How many dimensions would the sample plot have in this case?

Supervised learning

Example: Estimate the age of the people in the pictures!

x: A portrait
(a fixed-resolution color image)

y: The age of the person in the image

The brightness of each color channel of each pixel is an input variable:
Examples are points in a 196608 (+1 label) dimensional vector space.

$$x^{(j)} \in [0, 255]^{196608}$$

$$y^{(j)} \in \mathbb{R}$$

Visualizing the data

$x^{(j)} \in [0, 255]^{196608}$

$y^{(j)} \in \mathbb{R}$

input vars	#1 x_1 (pixel #1 "red" bright.)	#2 x_2 (pixel #1 "green" bright.)	...	#196608 x_{196608} (pixel #65535 "blue" brightness)	the label y (age of the person)
Img #1 (j = 1)	255	17	...	175	29
Img #2 (j = 2)	125	0	...	0	9
Img #3 (j = 3)	36	240	...	215	35
...	...	...	...	...	...

Supervised learning

How should our model look like?

**What type of hypothesis function are we looking for?**

The linear hypothesis function might not work well this time…

Supervised learning

What type of hypothesis function are we looking for?

The linear hypothesis function might not work well this time…

Instead, let's use a really complicated function, a deep convolutional neural network! **Neural networks are just functions too...**

$$h_\theta : \quad x \longrightarrow \qquad\qquad\qquad\qquad\qquad \longrightarrow \hat{y}$$

81x81x3
81x81x64
40x40x128
20x20x256
10x10x256
5x5x512
1x1x64
1x1x1024
1x1x1
10x10x256
5x5x512
2x2x512
1x1x256
20x20x128
40x40x128

A neural network can have millions of parameters, so **θ** here is **a vector** with millions of elements.
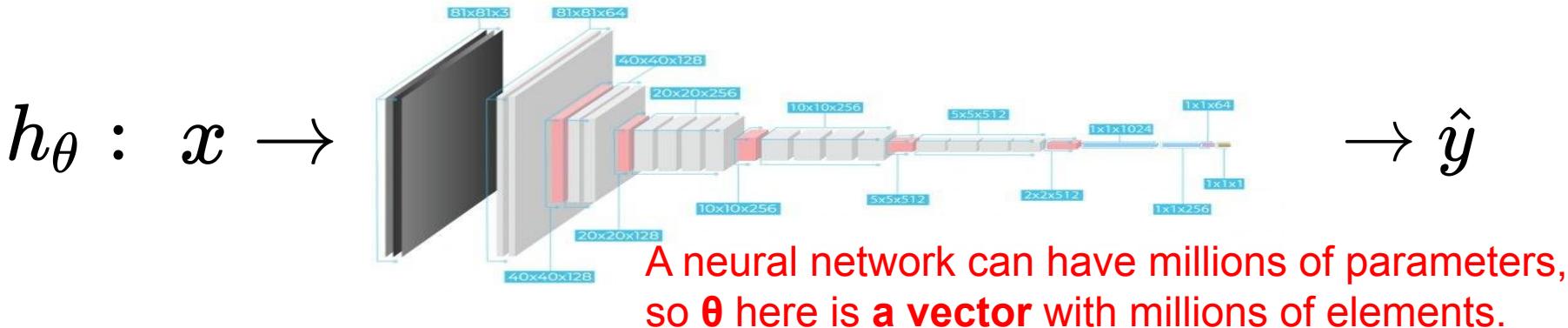
The two main types of tasks in supervised learning

Regression: Continuous labels   (The label set is infinite)

$$|Y| = \infty$$ **Example:** Number of cars or the age of a person

Classification: Discrete labels   (The label set is finite)

$$|Y| < \infty$$ **Example:** Categorization of examples

-   What is the profession of the person in the image?

Supervised learning

How do we determine how good the estimate is?

$$h_\theta(x) = \hat{y} \overset{?}{\approx} y$$
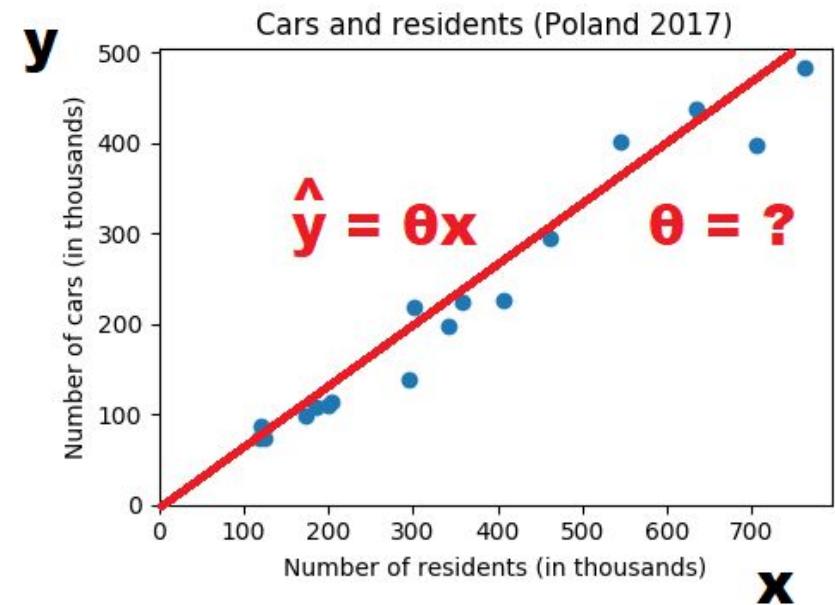
Cars and residents (Poland 2017)

$\hat{y} = \theta x$ $\theta = ?$

Supervised learning

How do we determine how good the estimate is?

With the help of the loss function J.

$$J : \theta \to \mathbb{R}_{\geq 0}$$

The loss function indicates **how much the actual label differs from our estimate** for given parameter values.

A simple example for a loss function:

$$J(\theta) = |\underbrace{h_\theta(x)}_{\hat{y}} - y|$$

Supervised learning

How do we determine how good the estimate is?

With the help of the loss function $J: \theta \to \mathbb{R}$

The loss function indicates **how much the actual label differs from our estimate** for given parameter values.

A simple example for a loss function:

$$J(\theta) = |h_\theta(x) - y|$$
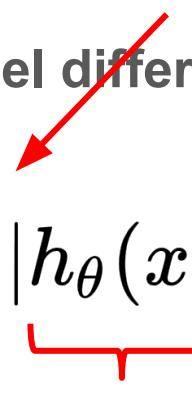
$$\underbrace{\qquad\qquad}_{\hat{y}}$$

The greater the error in our estimate with a given parameter **θ**, the greater the loss is in **θ**.
In the case of a perfect estimate, the loss is 0.

Supervised learning

The goal of supervised learning:

Find the **θ*** parameters that make our hypothesis function $h_\theta(x)$ approximate the true label (**y**) as closely as possible, i.e., minimize our loss **J**.

$$\theta^* = \arg\min_\theta J(\theta)$$

Supervised learning

The goal of supervised learning:

Find the **θ*** parameters that make our hypothesis function $h_\theta(x)$ approximate the true label (**y**) as closely as possible,
i.e., minimize our loss **J**.

$$\theta^* = \arg\min_\theta J(\theta)$$

Steps:

1. **Finding** (learning) **optimal parameters on the training set.**
2. **Estimating labels** for new, unlabeled examples by our trained model.

Supervised learning - applications

Supervised learning implemented with neural networks is also **suitable for more complex tasks...**

Supervised learning - applications

Object detection, object segmentation

x: An image

y: The coordinates of the enclosing rectangles (bounding boxes) of objects (bounding box), or the pixel-level outline of objects.
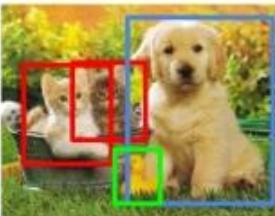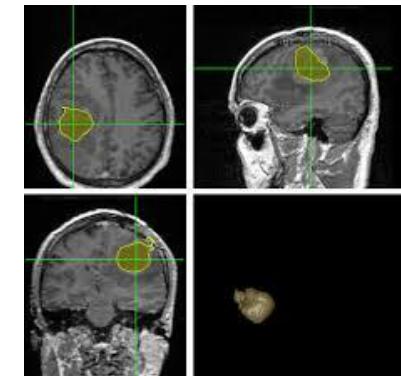
Supervised learning - applications

2D / 3D visual human pose estimation

x: An image

y: The joint coordinates of the persons in the image
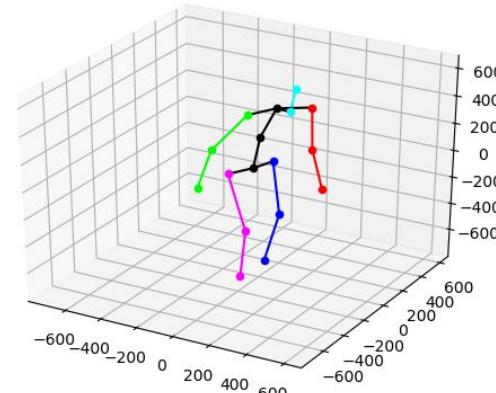
Supervised learning - applications

Natural Language Processing (NLP), Text sentiment analysis

**x:** A sequence of characters / words   **y:** A sentiment category
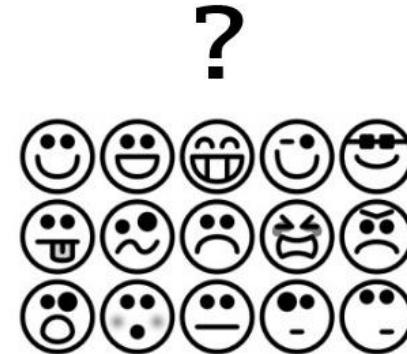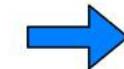
Supervised learning - applications

Natural Language Processing (NLP), Image captioning

x: An image   **y:** A sequence of characters / words describing the content of the image

"man in black shirt is playing guitar."

"construction worker in orange safety vest is working on road."

"two young girls are playing with lego toy."

Supervised learning - applications

Natural Language Processing (NLP), Machine translation

x: A sequence of characters / words in the source language

y: A sequence of characters / words in the target language

ICH MÖCHTE KEINE PERSIMONEN ESSEN

I   NOT   WANT   EAT   PERSIMMON

I NULL NOT WANT NULL EAT PERSIMMON

I DO NOT WANT TO EAT PERSIMMONS

Supervised learning - applications

Natural Language Processing: The problem of the loss function

"man in black shirt is playing guitar."

"construction worker in orange safety vest is working on road."

"two young girls are playing with lego toy."

ICH MÖCHTE KEINE PERSIMONEN ESSEN

I   NOT   WANT   EAT   PERSIMMON

I NULL NOT WANT NULL EAT PERSIMMON

I DO NOT WANT TO EAT PERSIMMONS

Supervised learning - applications

Natural Language Processing: The problem of the loss function

"man in black shirt is playing guitar."

"construction worker in orange safety vest is working on road."

"two young girls are playing with lego toy."

ICH MÖCHTE KEINE PERSIMONEN ESSEN

I   NOT   WANT   EAT   PERSIMMON

I NULL NOT WANT NULL EAT PERSIMMON
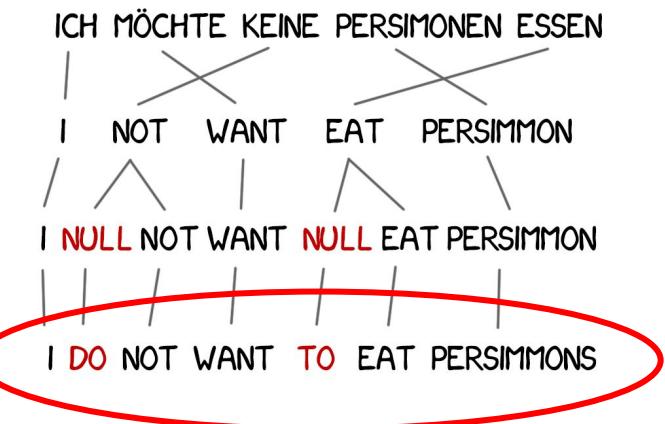
I DO NOT WANT TO EAT PERSIMMONS

The true label is not clear. The same meaning can be expressed in several ways. How do we know how good our estimate is? How do we calculate the error of our estimate (the loss value)?

Supervised learning

A problem of supervised learning

-   **Natural intelligence** (animals/humans) generally **does not require** specific **(x, y) input-output pairs** in order to learn.

Supervised learning

A problem of supervised learning

- **Natural intelligence** (animals/humans) generally **does not require** specific **(x, y) input-output pairs** in order to learn.

- The **production of labeled data** may require a **great deal of human labor**.
  However, there is a vast amount of unlabeled data available.
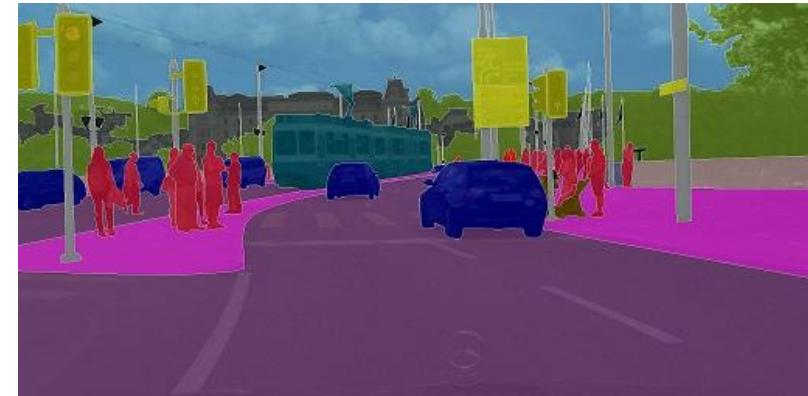
Machine learning

Three main groups of machine learning methods:

- Supervised learning
- **Unsupervised learning**
- Reinforcement learning

Unsupervised learning

Without labels, the task is unclear…

Several specific tasks are possible:

- **Clustering**
- **Compression / dimensionality reduction**
- **Sample generation**
- **...**

Unsupervised learning

Clustering

- **Marketing, targeted advertising, recommendation systems:** grouping similar customers for targeted offers
- **Classic computer vision,** e.g., image segmentation
- **Anomaly** (outlier) **detection**
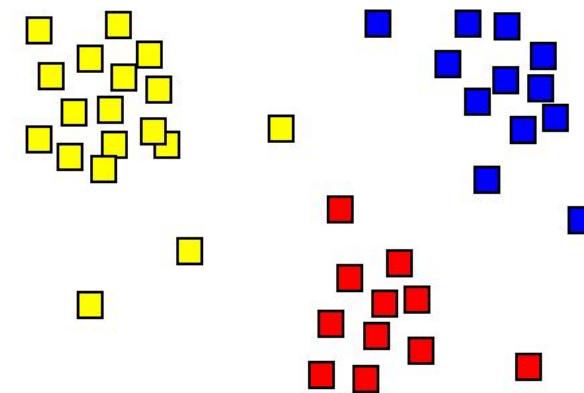
Unsupervised learning

Compression / dimensionality reduction

- Lossy compression
  adapted to work with
  a certain type of data

Unsupervised learning

Principal component analysis (PCA) on human faces. **Reconstruction** (x hat) from the compressed representation, with different compression ratios.

Compression / dimensionality reduction

0.1%

- Lossy compression adapted to work with a certain type of data

Original input (x)

1%

5%

10%

Unsupervised learning

Principal component analysis (PCA) on human faces. **Reconstruction** (x hat) from the compressed representation, with different compression ratios.

## Compression / dimensionality reduction

0.1%

- It only works well on data similar to the training images.

Original input (x)

1%

5%

10%

#efaces=1, res=67.381 #efaces=2, res=66.538 #efaces=5, res=65.311 #efaces=10, res=59.806 #efaces=20, res=57.978

#efaces=40, res=55.138 #efaces=60, res=51.779 #efaces=80, res=49.108 #efaces=100, res=46.238 #efaces=150, res=41.153

#efaces=200, res=38.072 #efaces=300, res=34.22 #efaces=400, res=31.315 #efaces=1000, res=20.659 #efaces=1071, res=19.855
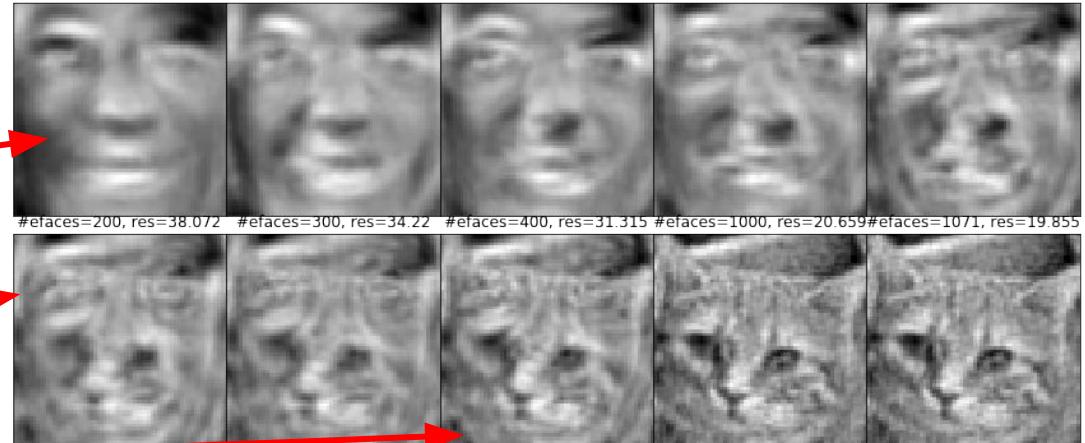
Unsupervised learning

Compression / dimensionality reduction

- **Less storage** space and bandwidth usage.
- A compressed representation (consisting of fewer variables) is often **easier to work with**.
- In a 2-3 dimensional representation, the location of the data points can even be **visualized on a plot** (points on a plane / in 3D space).

Unsupervised learning

Compression / dimensionality reduction

- **Less storage** space and bandwidth usage.
- A compressed representation (consisting of fewer variables) is often **easier to work with**.
- In a 2-3 dimensional representation, the location of the data points can even be **visualized on a plot** (points on a plane / in 3D space).

The task of compression / dimension reduction is well defined, but effective compression takes advantage of the fact that compressed data is similar in nature. It would be difficult to write unique algorithms for compressing many different types of data.
→ **Ideal task for machine learning**

Unsupervised learning

Compression / dimensionality reduction

The task can be formulated as follows:

x: The input (to be compressed)

h: The hypothesis-function performing
   the compression
   (e.g., an *autoencoder* neural network)

$$h_\theta(x) = \hat{x} \approx x$$

J: The (normed) difference between x and $\hat{x}$

Goal: Minimize J   (improve reconstruction)

Unsupervised learning

Compression / dimensionality reduction

The task can be formulated as follows:

x: The input (to be compressed)

h: The hypothesis-function performing
    the compression
    (e.g., an *autoencoder* neural network)

$$h_\theta(x) = \hat{x} \approx x$$

J: The (normed) difference between x and $\hat{x}$
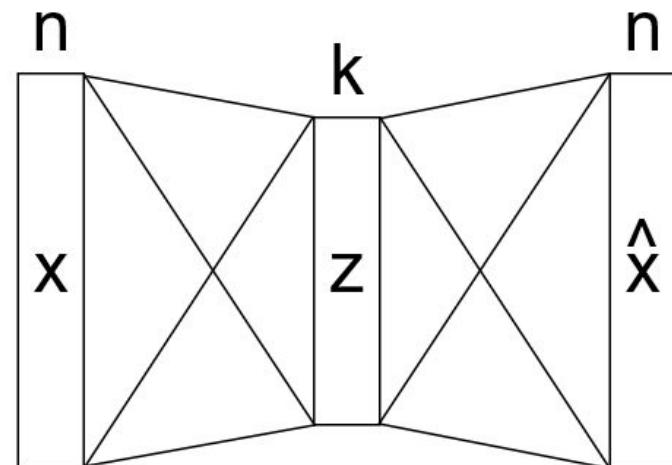
Goal: Minimize J   (improve reconstruction)

Goal: In the absence of a label
(**y**), we learn to recover the
input (**x**) from the compressed
representation (**z**).
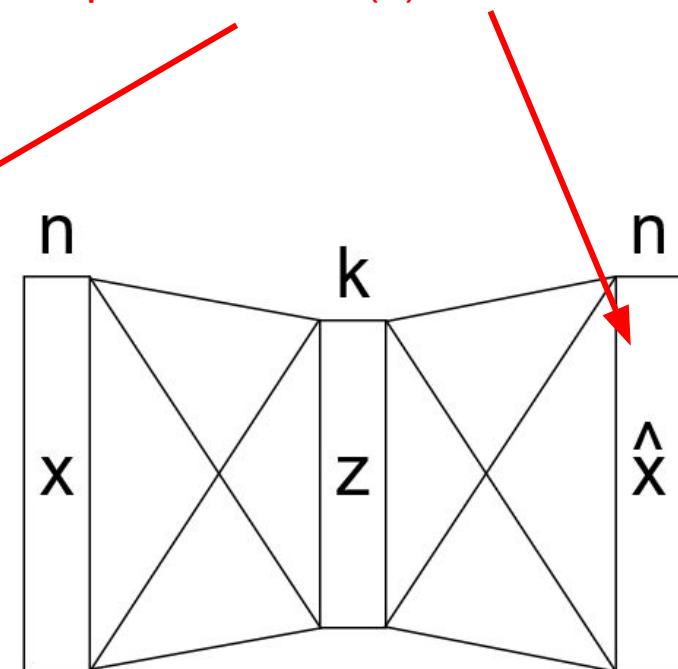
Unsupervised learning - applications

Denoising

Inpainting: Filling in missing parts of an image.
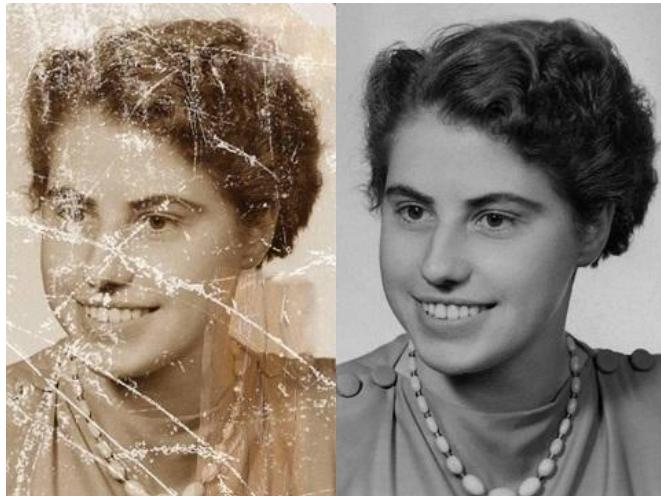
Denoising: Improving image quality

Unsupervised learning

Sample generation (learning the distribution of the data)

Generating new examples similar to those found in the training set.

# **Un**supervised learning

Sample generation (learning the distribution of the data)

Generating new examples similar to those found in the training set.
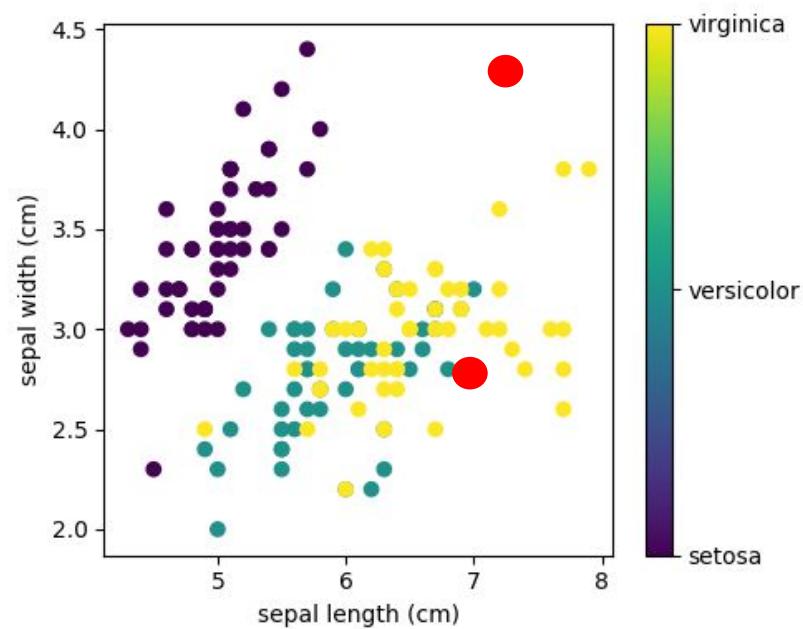
Unsupervised learning

Sample generation (learning the distribution of the data)

Generating new examples similar to those found in the training set.

This new sample element is unlikely: Its petals are too large.

This new sample is good: The petal sizes are similar to those of real flowers.
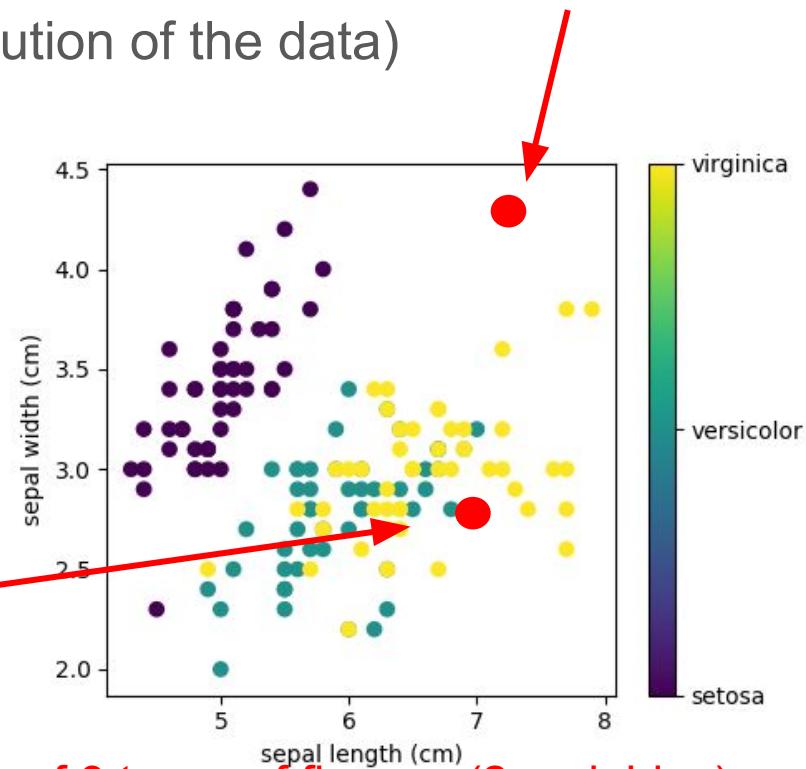
IRIS dataset: The length and width of the petals of 3 types of flowers (2 variables)

Unsupervised learning

Sample generation (learning the distribution of the data)
https://www.youtube.com/watch?v=36lE9tV9vm0

Unsupervised learning

Iterative denoising as a generative model (Stable Diffusion)
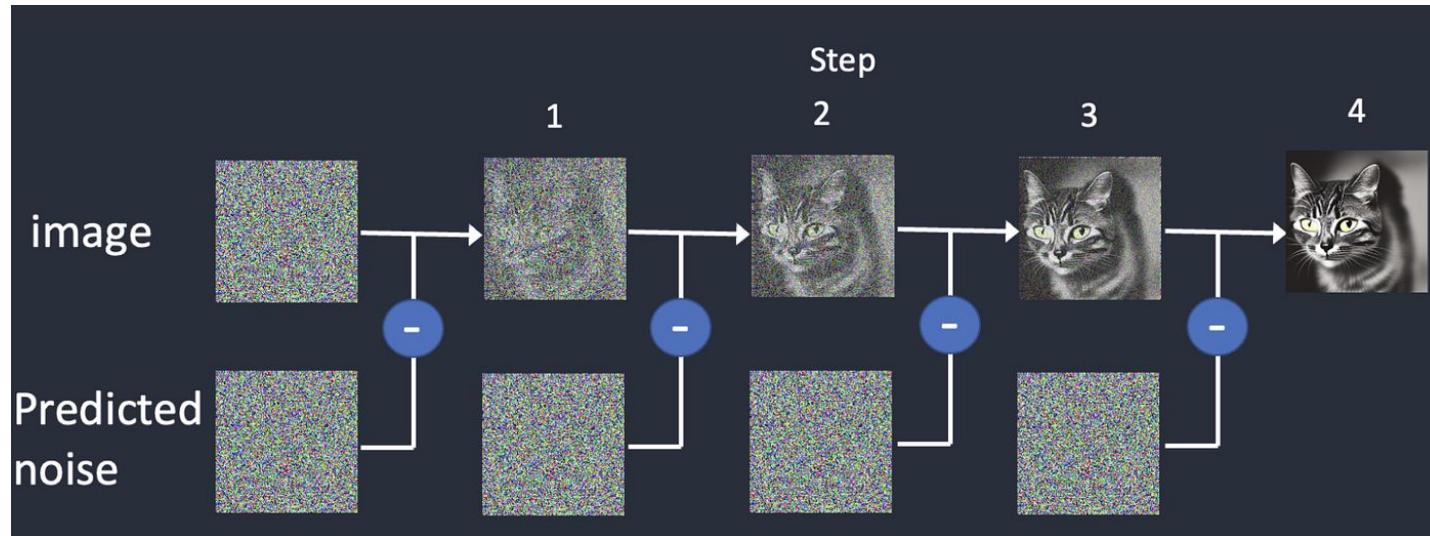
Unsupervised learning

Denoising / next token prediction, as an unsupervised learning task
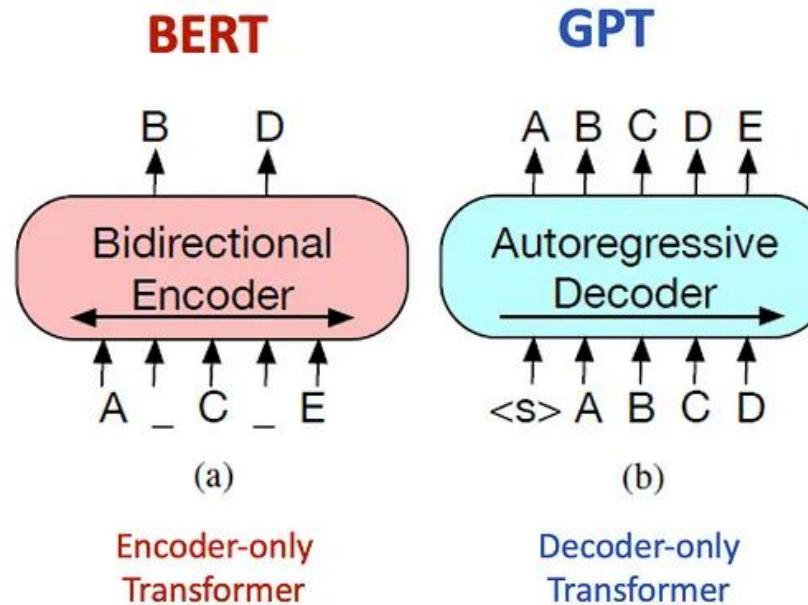
Unsupervised learning

Large Language Models (LLMs)

Unsupervised training on
a large amount of data, **then
fine-tuning** on partially
supervised language tasks.
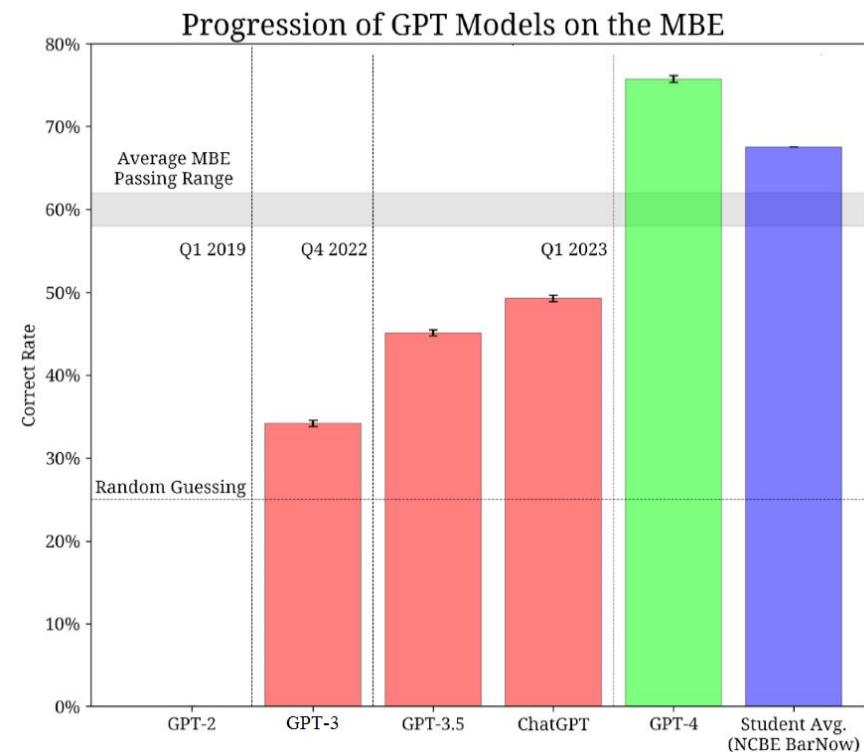
Progression of GPT Models on the MBE

Machine learning

Three main groups of machine learning methods:

- Supervised learning
- Unsupervised learning
- **Reinforcement learning**

Reinforcement learning

Natural intelligence rarely encounters supervised learning tasks...

More common: We perform a series of actions,
and if they lead to objectively assessable results,
we receive some kind of **positive or negative feedback.**

→ **Reinforcement Learning** (RL)

Reinforcement learning

- The **agent** navigates the **environment** (and influences it).
- The **agent** observes the **state** of the environment and then modifies it by performing some **action**.
- If certain predefined goals are met in the environment, the agent receives a (possibly negative) **reward**.
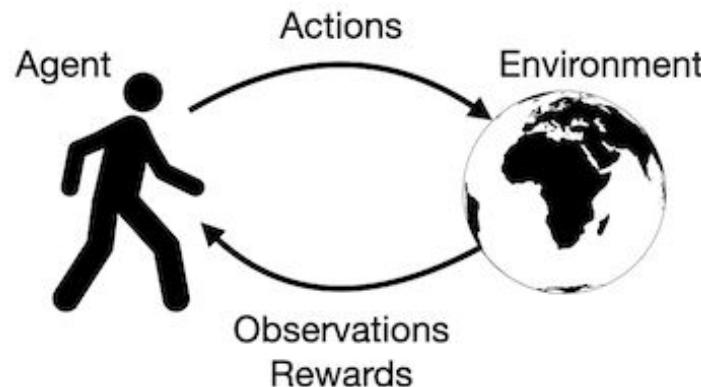
Reinforcement learning

Goal: Choosing actions that are expected to **maximize** the total amount of **rewards** received in the future.

- The **agent** navigates the **environment** (and influences it).
- The **agent** observes the **state** of the environment and then modifies it by performing some **action**.
- If certain predefined goals are met in the environment, the agent receives a (possibly negative) **reward**.
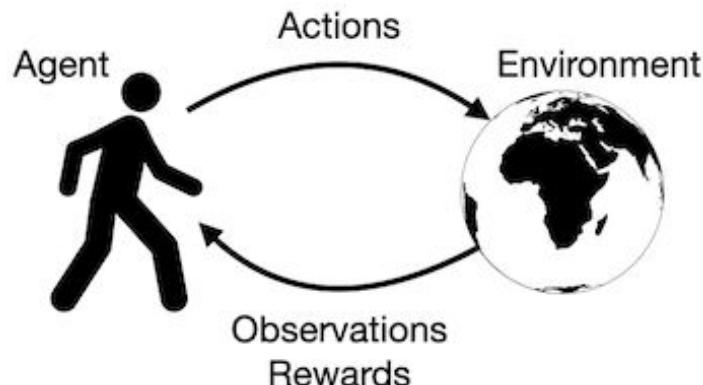
Reinforcement learning

Sparse feedback. The *Credit Assignment problem*.

$r_1, \ldots, r_{51} = 0$

1.d4 ♞f6 2.c4 e6 3.♞f3 b6 4.g3 ♝a6 5.b3 ♝b4+ 6.♝d2 ♝e7 7.♞c3 c6 8.e4 d5 9.♛c2 dxe4 10.♞xe4 ♝b7 11.♞eg5 c5 12.d5 exd5 13.cxd5 h6 14.♞xf7 ♚xf7 15.O-O-O ♝d6 16.♞h4 ♝c8 17.♖e1 ♞a6 18.♖e6 ♞b4 19.♝xb4 cxb4 20.♝c4 b5 21.♝xb5 ♝e7 22.♞g6 ♞xd5 23.♖xe7+ ♞xe7 24.♝c4+ ♚f6 25.♞xh8 ♛d4 26.♖d1 ♛a1+ 27.♚d2 ♛d4+ 28.♚e1 ♛e5+ 29.♛e2 ♛xe2+ 30.♚xe2 ♞f5 31.♞f7 a5 32.g4 ♞h4 33.h3 ♖a7 34.♖d6+ ♚e7 35.♖b6 ♖c7 36.♞e5 ♞g2 37.♞g6+ ♚d8 38.♚f1 ♝b7 39.♖xb7 ♖xb7 40.♚xg2 ♖d7 41.♞f8 ♖d2 42.♞e6+ ♚e7 43.♞xg7 ♖xa2 44.♞f5+ ♚f6 45.♞xh6 ♖c2 46.♝f7 ♖c3 47.f4 a4 48.bxa4 b3 49.g5+ ♚g7 50.f5 b2 51.f6+ ♚h7 52.♞f5 1-0

$r_{52} = 100$

Reinforcement learning

AlphaGo / AlphaZero: Superhuman level Go skills.
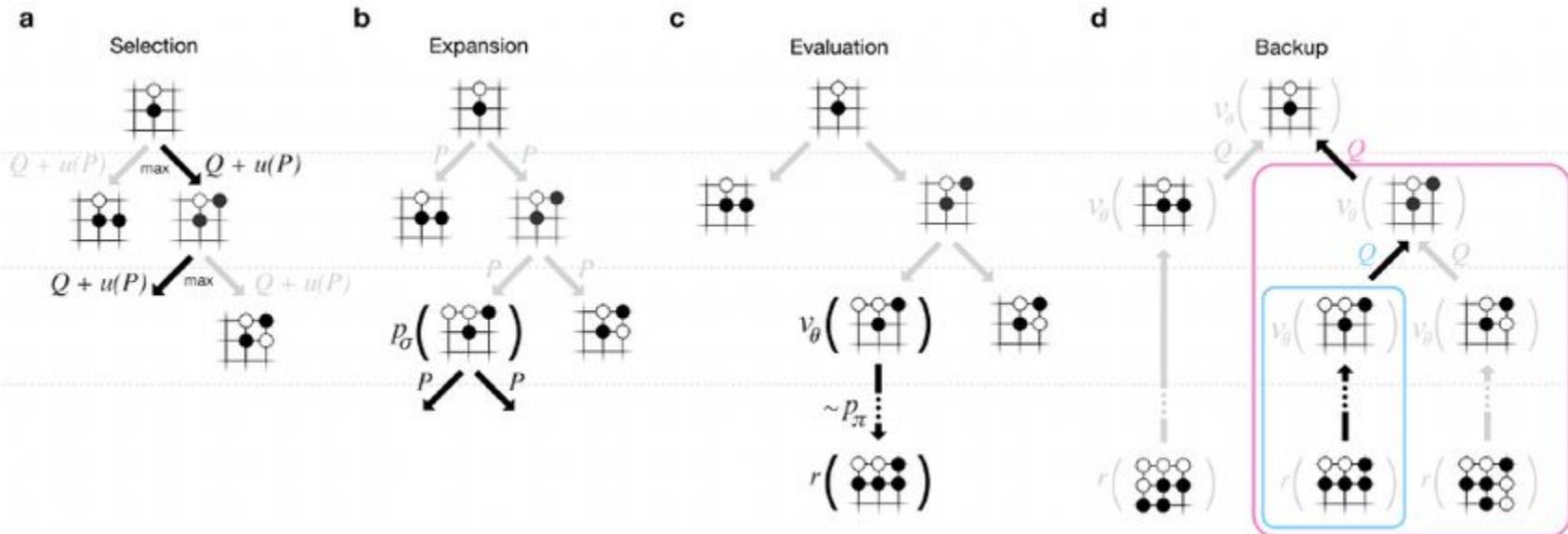
Key questions of artificial intelligence

The ability to generalize

Does our trained model memorize the training examples one by one, or does it actually learn "useful" things?
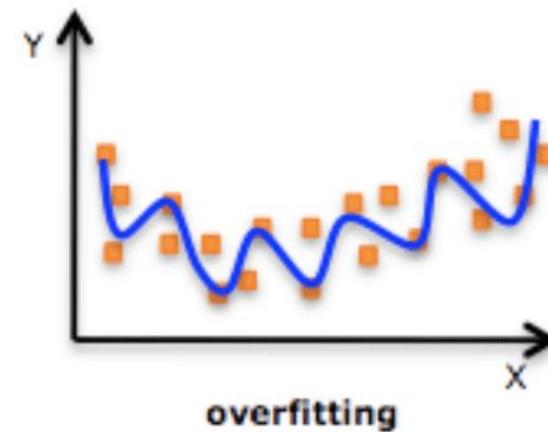
Just right!

overfitting

Key questions of artificial intelligence

Explainability

Some classic machine learning
methods learn easily
explainable / interpretable models.

Decision trees are one example:

Is a Person Fit?

Age < 30 ?

Yes?   No?

Eat's a lot   Exercises in
of pizzas?   the morning?

Yes?   No? Yes?   No?

Unfit!   Fit   Fit   Unfit!

Key questions of artificial intelligence

Explainability

Some classic machine learning
methods learn easily
explainable / interpretable models.

Decision trees are one example:

Is a Person Fit?

Age < 30 ?

Yes?   No?

Eat's a lot   Exercises in
of pizzas?   the morning?

Yes?   No?   Yes?   No?

Unfit!   Fit   Fit   Unfit!
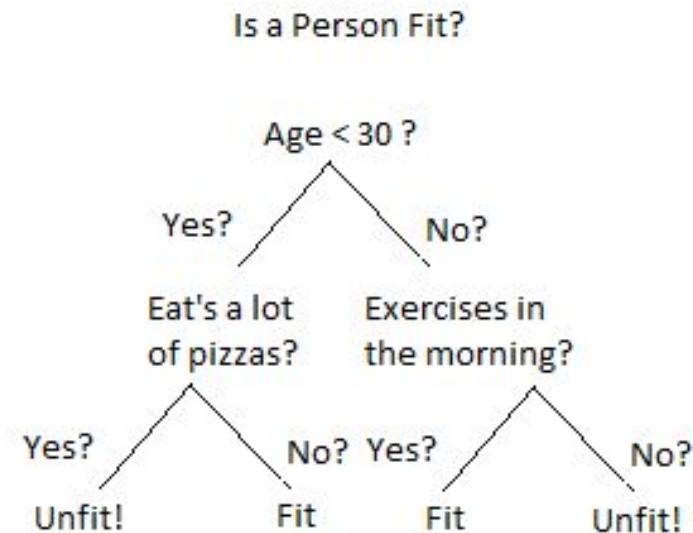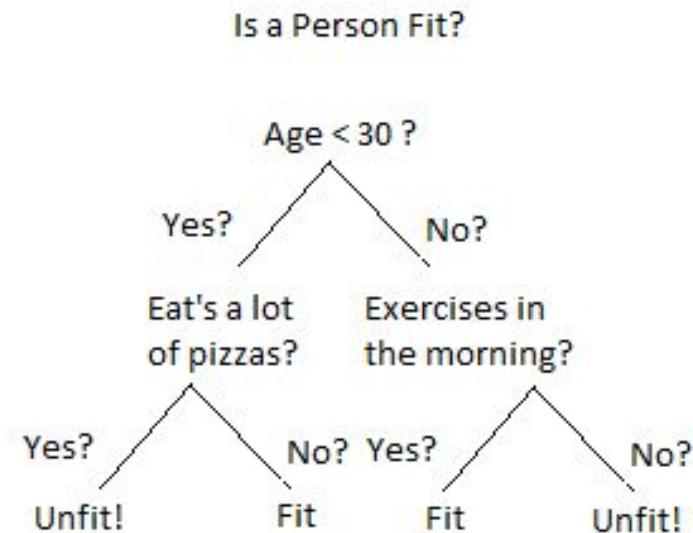
They are very limited, unfortunately…

Key questions of artificial intelligence

Explainability

The knowledge learned by neural networks is difficult to interpret.

Key questions of artificial intelligence

Explainability

The knowledge learned by neural networks is difficult to interpret.

(c)   (d)   (e)   (f)   (g)

Estimates of a neural network categorizing objects (animals): Depending on the horse's environment, the network estimates the labels "horse," "cow," and "dog" for a picture of a horse. The neural network has therefore learned to estimate the label better from the appearance of its environment than from the appearance of the animal itself...

Key questions of artificial intelligence

Explainability

Software used in critical applications
is thoroughly tested and formally verified.

Deep learning methods can only be tested
empirically.
**We cannot say whether they
made their decisions in a "logical way"...**

Key questions of artificial intelligence

Safety - Adversarial attacks

Key questions of artificial intelligence

Safety - Adversarial attacks

"pig" + 0.005 x = "airliner"
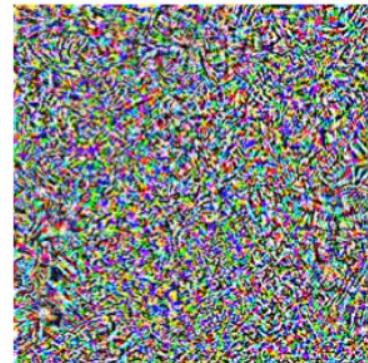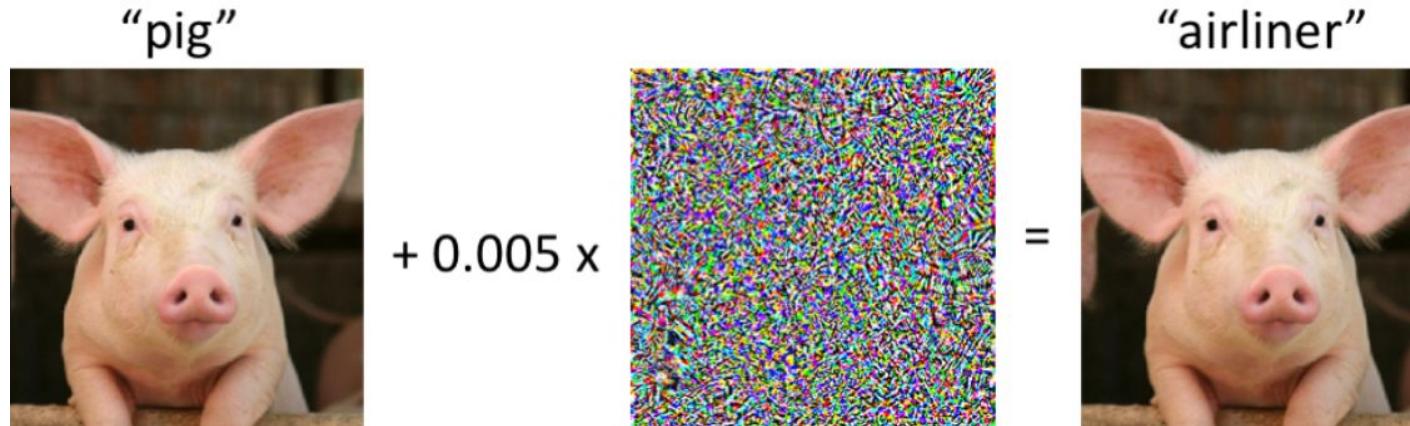
Adversarial attacks: Even without the knowledge of the neural network architecture and parameters, a specific small perturbation of the image can be found, that changes the label estimates dramatically.

Key questions of artificial intelligence

Safety - Adversarial attacks

The attacks work in reality, as well…

Key questions of artificial intelligence

Explainability

Knowledge learned by artificial neural networks **cannot be explained** well.

However, can humans explain how they recognize a familiar face or voice?

Software

Python

- High level
- Automatic memory management
- Runtime type inference and checking
- Interpreted language
- Simple syntax, limited language features
- Portable
- Lots of tools in many available libraries
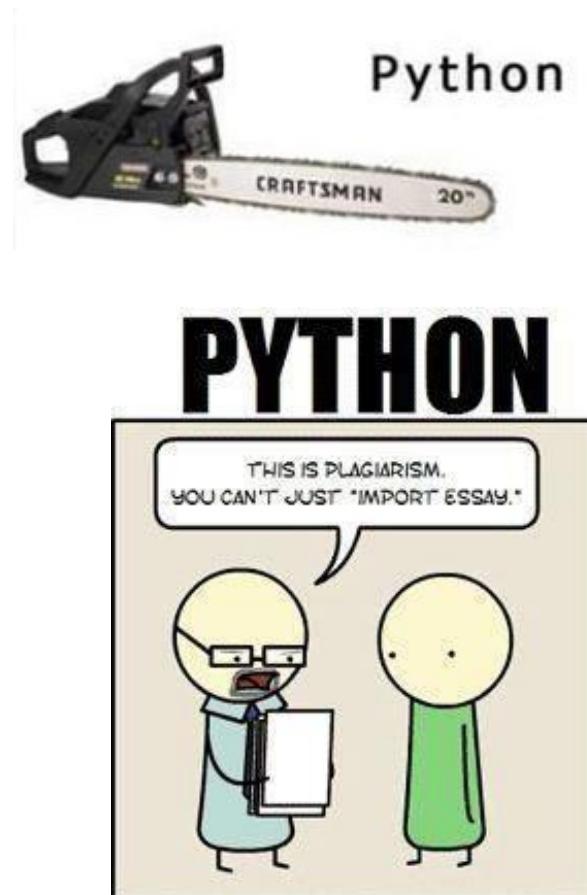
Software

Python

- High level
- Automatic memory management
- Runtime type inference and checking
- Interpreted language
- Simple syntax, limited language features
- Portable
- Lots of tools in many available libraries

**Convenient.
Rapid development,
experimentation.**

Software

Python

- High level
- Automatic memory management
- Runtime type inference and checking
- Interpreted language
- Simple syntax, limited language features
- Portable
- Lots of tools in many available libraries

Slow execution.
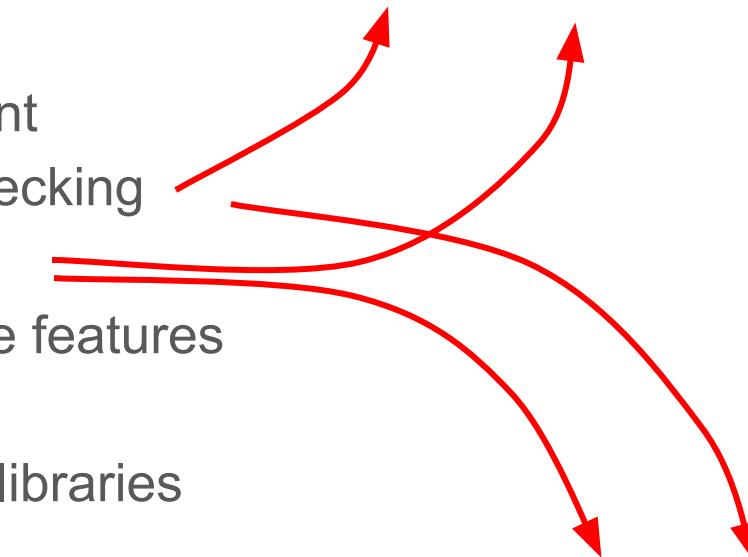
Most errors are only revealed at runtime.

Software

Python

"Slow execution".

If Python is not efficient, why has it become popular in (computationally intensive) **machine learning?**

Software

Python

"Slow execution".

If Python is not efficient, why has it become popular in (computationally intensive) **machine learning?**

We perform our computationally intensive operations efficiently using libraries written in other languages.
We only use Python for scripting!

Software

NumPy - Generating and multiplying two random matrices (400 x 400)

Naive Python: ?

```python
matrix1 = [[random.random() for col_idx in range(matrix_dim)] for row_idx in range(matrix_dim)]
matrix2 = [[random.random() for col_idx in range(matrix_dim)] for row_idx in range(matrix_dim)]
matrix_out = [[0. for col_idx in range(matrix_dim)] for row_idx in range(matrix_dim)]

for row1_idx in range(matrix_dim):
  for col2_idx in range(matrix_dim):
    for item_idx in range(matrix_dim):
      matrix_out[row1_idx][col2_idx] += matrix1[row1_idx][item_idx] * matrix2[item_idx][col2_idx]
```

NumPy: ?

```python
matrix1 = np.random.rand(matrix_dim, matrix_dim)
matrix2 = np.random.rand(matrix_dim, matrix_dim)


return np.matmul(matrix1, matrix2)
```

Software

NumPy

NumPy - Generating and multiplying two random matrices (400 x 400)

Naive Python: **14.79 sec**

```
matrix1 = [[random.random() for col_idx in range(matrix_dim)] for row_idx in range(matrix_dim)]
matrix2 = [[random.random() for col_idx in range(matrix_dim)] for row_idx in range(matrix_dim)]
matrix_out = [[0. for col_idx in range(matrix_dim)] for row_idx in range(matrix_dim)]

for row1_idx in range(matrix_dim):
  for col2_idx in range(matrix_dim):
    for item_idx in range(matrix_dim):
      matrix_out[row1_idx][col2_idx] += matrix1[row1_idx][item_idx] * matrix2[item_idx][col2_idx]
```

NumPy: **0.019 sec**

```
matrix1 = np.random.rand(matrix_dim, matrix_dim)
matrix2 = np.random.rand(matrix_dim, matrix_dim)

return np.matmul(matrix1, matrix2)
```

Software

NumPy example: Count elements divisible by 3 or 5!

Naive Python:

```python
c = 0
for elem in my_list:
  if (elem % 3 == 0) or (elem % 5 == 0):
    c += 1
```

NumPy:

```python
c = np.count_nonzero((my_array % 3 == 0) | (my_array % 5 == 0))
```

Software

NumPy example: Count elements divisible by 3 or 5!

Naive Python:

```python
c = 0
for elem in my_list:
  if (elem % 3 == 0) or (elem % 5 == 0):
    c += 1
```

NumPy:

```python
c = np.count_nonzero((my_array % 3 == 0) | (my_array % 5 == 0))
```

Array operations: %, ==, |

Software

NumPy logo

NumPy example: Given several sets of three numbers. Specify which ones can be used to form triangles, and then specify the area and perimeter of these triangles.

```python
def triangle_area_perimeter(edges):
  # edges shape: (n_triples, 3)
  valid_tr_mask = (edges[:,0] + edges[:,1] > edges[:,2]) & (edges[:,0] + edges[:,2] > edges[:,1]) &\
                  (edges[:,1] + edges[:,2] > edges[:,0]) & np.all(edges >= 0., axis=1)
  valid_triangle_idxs = np.where(valid_tr_mask)[0]
  edges = edges[valid_triangle_idxs]
  perimeter = np.sum(edges, axis=1)
  semi_per = perimeter*.5
  area = np.sqrt(semi_per*(semi_per-edges[:,0])*(semi_per-edges[:,1])*(semi_per-edges[:,2]))
  area_perimeter = np.stack([area, perimeter], axis=1)
  return valid_triangle_idxs, area_perimeter
```

Software

NumPy example: Given several sets of three numbers. Specify which ones can be used to form triangles, and then specify the area and perimeter of these triangles.

N dimensional arrays, indices

```python
def triangle_area_perimeter(edges):
    # edges shape: (n_triples, 3)
    valid_tr_mask = (edges[:,0] + edges[:,1] > edges[:,2]) & (edges[:,0] + edges[:,2] > edges[:,1]) &\
                    (edges[:,1] + edges[:,2] > edges[:,0]) & np.all(edges >= 0., axis=1)
    valid_triangle_idxs = np.where(valid_tr_mask)[0]
    edges = edges[valid_triangle_idxs]
    perimeter = np.sum(edges, axis=1)
    semi_per = perimeter*.5
    area = np.sqrt(semi_per*(semi_per-edges[:,0])*(semi_per-edges[:,1])*(semi_per-edges[:,2]))
    area_perimeter = np.stack([area, perimeter], axis=1)
    return valid_triangle_idxs, area_perimeter
```

Indexing with all elements of an array at once

Executing operations across entire axes of an array

Software

Array programming

- Execution of a **single operation on a large amount of data** at once.
- Code implemented with array operations is much **easier to translate into parallel machine code**.
- In machine learning, we mostly use operations that are well suited for **parallelization**.
- Code written with array programming is generally much more **compact**.

Software

PyTorch - Library for implementing neural networks, with GPU support

```python
import torch
import torch.nn as nn

model = nn.Sequential(
    nn.Linear(28, 50),
    nn.ReLU(),
    nn.Linear(50, 10)
)

loss_fn = nn.MSELoss()
optimizer = torch.optim.Adam(model.parameters(), lr=0.001)

for n in range(num_epochs):
    y_pred = model(X)              # X: inputs from training set, y_pred: estimation of labels
    loss = loss_fn(y_pred, y)      # y: true labels from training set
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
```

High-level components for building neural networks

Software

Google Colab

https://colab.research.google.com

Synonyms in ML terminology

- Example ≈ (data) sample ≈ data point ≈ observation
- (Input) variable ≈ (input) feature
- Training ≈ fitting
- Prediction ≈ estimation
- (True) label ≈ (expected) output ≈ target variable ≈ ground truth
- Predicted/estimated label ≈ predicted/actual output ≈ estimates
- (Model) parameters ≈ (model) weights
- Loss function ≈ cost function

θ: theta