Introduction to Machine Learning. Basic Concepts and Learning Paradigms.

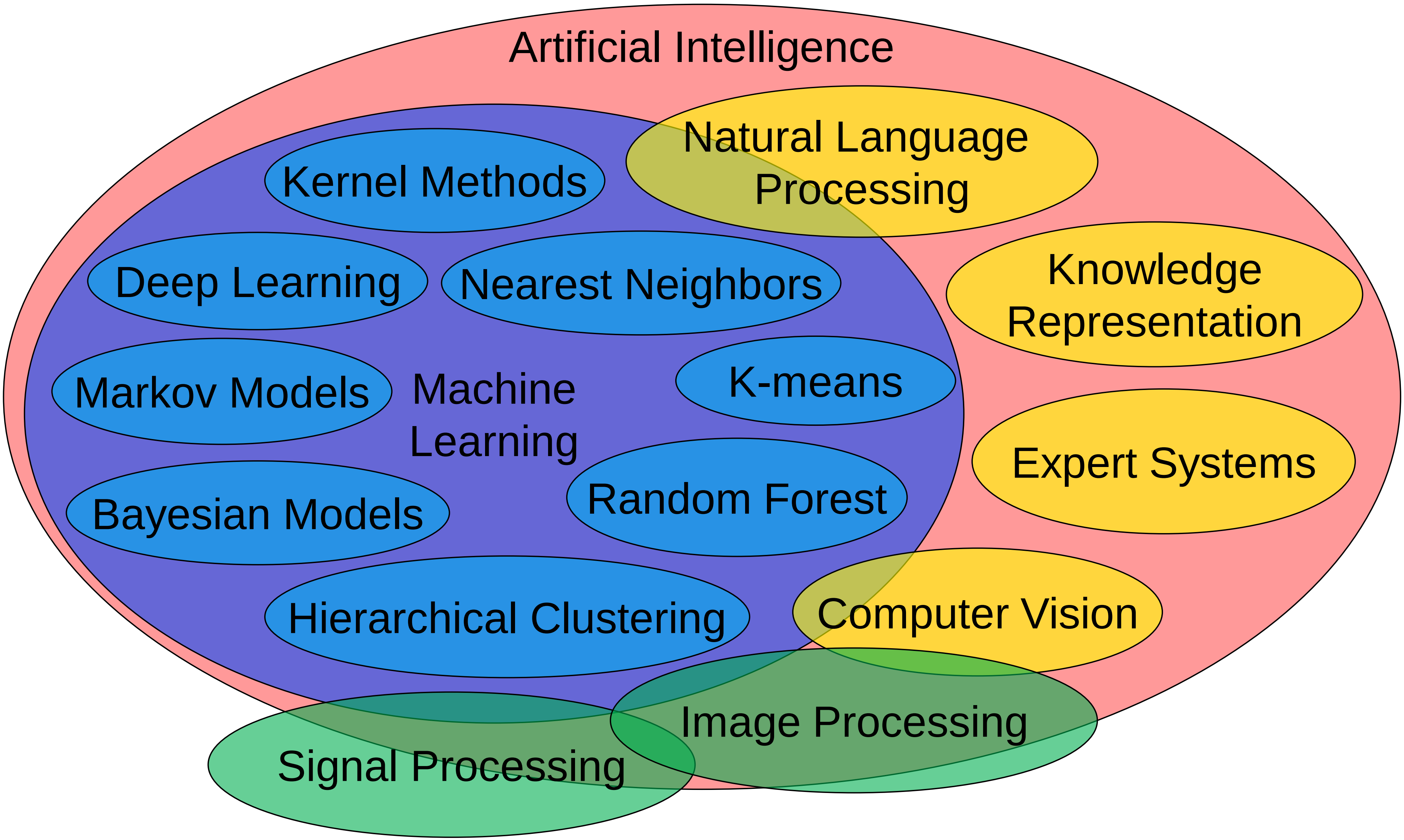
Radu Ionescu, Prof. PhD.

raducu.ionescu@gmail.com

Faculty of Mathematics and Computer Science

University of Bucharest

Machine Learning



Instructors

- Lectures:

➢Radu Ionescu (raducu.ionescu@gmail.com)

- Labs:

➢Adriana Costache (adriana16costache@gmail.com)

➢Silviu Gheorghe (ghesil@gmail.com)

➢Mădălina Poșchină (madalinaposchina@gmail.comaaa)

➢Vlad Hondru (vlad.hondru25@gmail.com)

- Website:

https://practical-ml-fmi.github.io/ML/

- Team code: 1jr5sik

Grading System

- Your final grade* is composed of:
  - 20% for Project 1 (minimum 10%)
  - 20% for Project 2 (minimum 10%)
  - 60% for ORAL exam (minimum 30%)

  (*) subject to passing the minimum grade for each!
- Both projects are individual!
- Each project consists of employing machine learning methods on a specific data set
- Project 1 is about participating in a Kaggle competition
  - The competition will be launched in a couple of weeks
- Project 2 is about comparing two unsupervised approaches
  - There are many datasets out there, so no overlap allowed among students!
  - Methods and data sets must be chosen beforehand!

Grading System

- Projects must be presented no later than the day of the “exam”
- There will be no paper exam, only ORAL exam!
- The projects consist of the code implementation in Python (any library is allowed) and a PDF report including (0.5 points):
  - a description of the data set (for project 2 only)
  - a description of the implemented machine learning methods
  - figures and / or tables with results / hyperparameter tuning
  - comments / interpretation for the results
  - conclusion

Grading System

- First project consists of implementing some machine learning method(s) for the proposed Kaggle challenge (TBA)
- The grades will be proportional to your model’s accuracy:

- Top 1-20 => your grade can be up to 2

- Top 21-50 => your grade can be up to 1.8

- Top 51-80 => your grade can be up to 1.6

- Top 81-100 => your grade can be up to 1.4

- Top 101-120 => your grade can be up to 1.2

- Others => your grade can be up to 1

- Ranks can change depending on the final number of participants

Grading System

- Submit projects to: practical.ml.fmi@gmail.com
- Submit only .py files only! (.ipynb not accepted)
- We will set deadlines (during every evaluation session) for:
  - choosing the projects
  - submitting the projects
  - presenting the projects
- If you don’t know the dates, please ask! Don’t wait until the presentation day!
- Demonstrating good knowledge about the studied machine learning methods is mandatory to get a passing grade!
- If you fail the oral exam, projects need to be redone!

Grading System

- Extra points during lectures / labs
  - ➢ awarded only in the first round of evaluation
- Lectures:
  - ➢ awarded based on the ranking of answers on Kahoot
  - ➢ top 3 get up to 0.3 points per lecture, next 3 up to 0.2 points and so on
- Labs:
  - ➢ first to answer solve an exercise gets 0.2 points
  - ➢ maximum 0.4 points per lab for each student
- Up to 1 bonus point during lectures (added to final grade)
- Up to 1 bonus point during labs (added to final grade)

(NO) Collaboration Policy

- Collaboration
  - Each student must write their own code for the project(s)
  - Borrowing code from web sources with copy & paste is not permitted under any circumstances
- No tolerance on plagiarism
  - Neither ethical nor in your best interest
  - Code will be checked automatically and manually!
  - Don’t cheat. We will find out!

We are serious about this!

Examples of unacceptable plagiarism

```
# average test loss
test_loss = test_loss/len(validloader.dataset)
print('Test Loss: {:.6f}\n'.format(test_loss))

for i in range(3):
    if class_total[i] > 0:
        print('Test Accuracy of %5s: %2d%% (%2d/%2d)' % (
            classes[i], 100 * class_correct[i] / class_total[i],
            np.sum(class_correct[i]), np.sum(class_total[i])))
    else:
        print('Test Accuracy of %5s: N/A (no training examples)' % (classes[i]))

print('\nTest Accuracy (Overall): %2d%% (%2d/%2d)' % (
    100. * np.sum(class_correct) / np.sum(class_total),
    np.sum(class_correct), np.sum(class_total)))
```

Examples of unacceptable plagiarism

4

```
print('Epoch: {} \tTraining Loss: {:.6f} \tValidation Loss: {:.6f}'.format(
    epoch, train_loss, valid_loss))

# save model if validation loss has decreased
if valid_loss <= valid_loss_min:
    print('Validation loss decreased ({:.6f} --> {:.6f}).  Saving model ...'.format(
        valid_loss_min,
        valid_loss))
    torch.save(model.state_dict(), 'model_curent.pt')
    valid_loss_min = valid_loss

model.load_state_dict(torch.load('model_curent.pt'))
```

Examples of unacceptable plagiarism

```
batch_size = 64
for data, target in validloader:
    # move tensors to GPU if CUDA is available
    if train_on_gpu:
        data, target = data.cuda(), target.cuda()
    # forward pass: compute predicted outputs by passing inputs to the model
    output = model(data)
    # calculate the batch loss
    loss = criterion(output, target)
    # update test loss
    test_loss += loss.item()*data.size(0)
    # convert output probabilities to predicted class
    _, pred = torch.max(output, 1)
    # compare predictions to true label
    correct_tensor = pred.eq(target.data.view_as(pred))
    correct = np.squeeze(correct_tensor.numpy()) if not train_on_gpu else
np.squeeze(correct_tensor.cpu().numpy())
```

Examples of unacceptable plagiarism

```
########################
# validate the model #
########################
model.eval()
for data, target in validloader:
    # move tensors to GPU if CUDA is available
    if train_on_gpu:
        data, target = data.cuda(), target.cuda()
    # forward pass: compute predicted outputs by passing inputs to the model
    output = model(data)
    # calculate the batch loss
    loss = criterion(output, target)
    # update average validation loss
    valid_loss += loss.item() * data.size(0)
```

Examples of unacceptable plagiarism

```
def forward(self, x):
    x = F.relu(F.max_pool2d(self.conv1(x), 2))
    x = F.relu(F.max_pool2d(self.conv2_drop(self.conv2(x)), 2))
    x = F.relu(F.max_pool2d(self.conv2_drop(self.conv3(x)), 2))
    x = x.view(x.shape[0],-1)
    x = F.relu(self.fc1(x))
    x = F.dropout(x, training=self.training)
    x = self.fc2(x)
    x = F.dropout(x, training=self.training)
    x = self.fc3(x)
    return x
```

Examples of unacceptable plagiarism

```
1
model.eval()
for data, target in validloader:
    # move tensors to GPU if CUDA is available
    if train_on_gpu:
        data, target = data.cuda(), target.cuda()
    # forward pass: compute predicted outputs by passing inputs to the model
    output = model(data)
    # calculate the batch loss
    loss = criterion(output, target)
    # update average validation loss
    valid_loss += loss.item() * data.size(0)

    1
    _, pred = torch.max(output, 1)
    # compare predictions to true label
    correct_tensor = pred.eq(target.data.view_as(pred))
    correct = np.squeeze(correct_tensor.numpy()) if not train_on_gpu else
np.squeeze(correct_tensor.cpu().numpy())
```

Examples of acceptable code

```
from keras.layers import Conv2D, Activation, MaxPooling2D, Flatten, Dense, Dropout
from keras.models import Sequential
from pandas import read_csv
from sklearn.metrics import confusion_matrix
from tqdm import tqdm
from keras.preprocessing import image
from keras.utils.np_utils import to_categorical
import numpy as np
import plot as plt
```

Examples of acceptable code

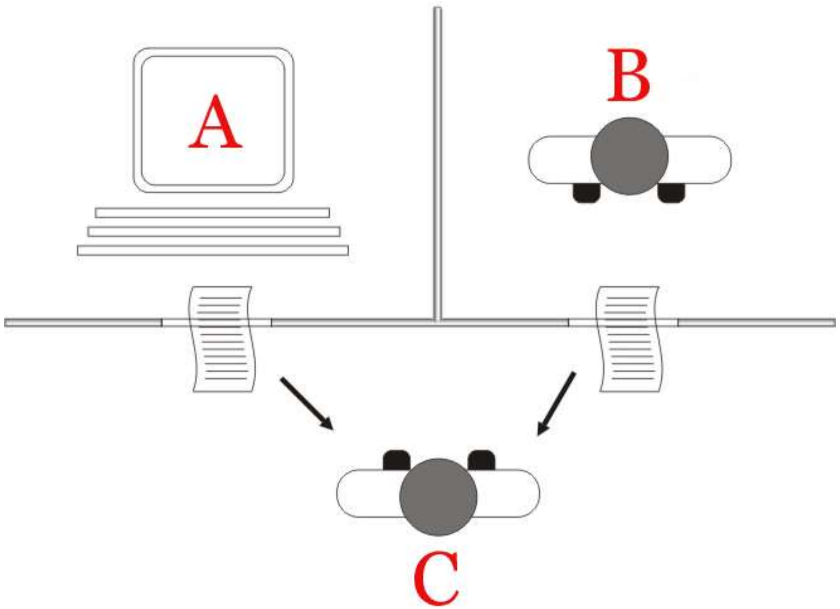
```
    imagini_validare.append(imagine)
imagini_train = np.array(imagini_train)
imagini_validare = np.array(imagini_validare)
train_labels = np.array(train_labels)
validation_labels = np.array(validation_labels)

nume_imagini = []
i = 0
ordine = dict()
for numeImagine in os.listdir(PATH + "/test"):
    imagine = Image.open(PATH + "/test/" + numeImagine)
    #imagine = imagine.convert('RGB')
    imagine = np.array(imagine).astype('d')
    imagini_test.append(imagine)
    ordine[numeImagine] = i
    i += 1
    nume_imagini.append(numeImagine)
imagini_test = np.array(imagini_test)
imagini_train = np.repeat(imagini_train[..., np.newaxis], 3, -1)
imagini_test = np.repeat(imagini_test[..., np.newaxis], 3, -1)
imagini_validare = np.repeat(imagini_validare[..., np.newaxis], 3, -1)
```

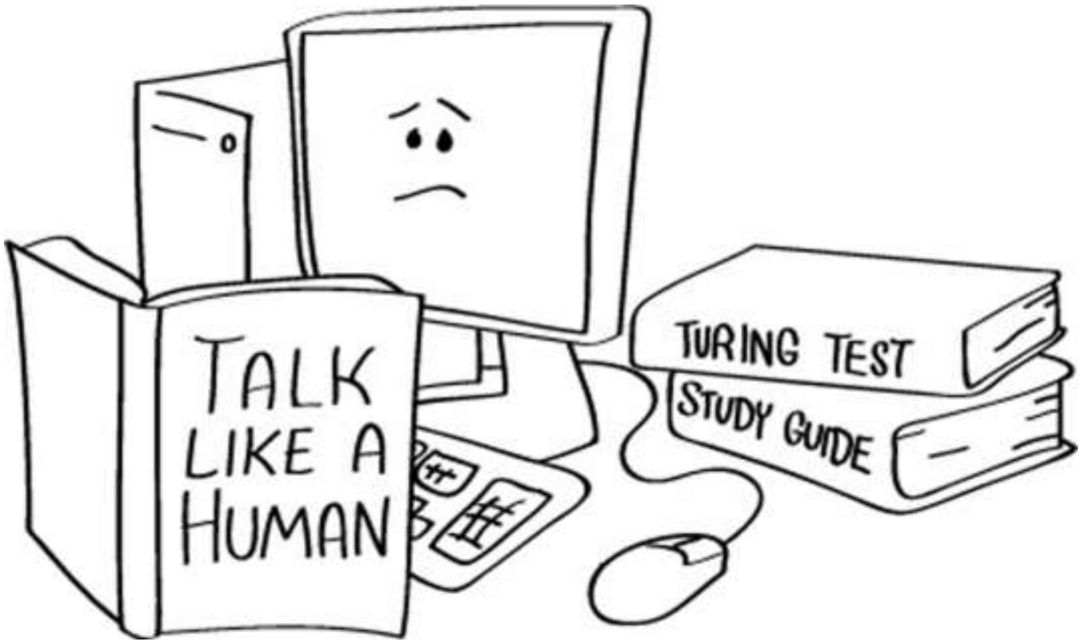
Examples of acceptable code

```
keras_model.trainable = True
keras_model.compile(optimizer=keras.optimizers.Adam(1e-6),
            loss=keras.losses.SparseCategoricalCrossentropy(from_logits=True),  # default
from_logits=False
            metrics=[keras.metrics.SparseCategoricalAccuracy()])
earlystopping = tf.keras.callbacks.EarlyStopping(monitor="val_loss",
                        mode="min", patience=10,
                        restore_best_weights=True)
hist2 = keras_model.fit(imagini_train, train_labels, epochs=1000,
validation_data=(imagini_validare, validation_labels),
             callbacks=[earlystopping, checkpoint])
```

What is artificial intelligence (AI)?

- The ultimate goal of artificial intelligence is to build systems able to reach human intelligence levels
- Turing test: a computer is said to possess human-level intelligence if a remote human interrogator, within a fixed time frame, cannot distinguish between the computer and a human subject based on their replies to various questions posed by the interrogator



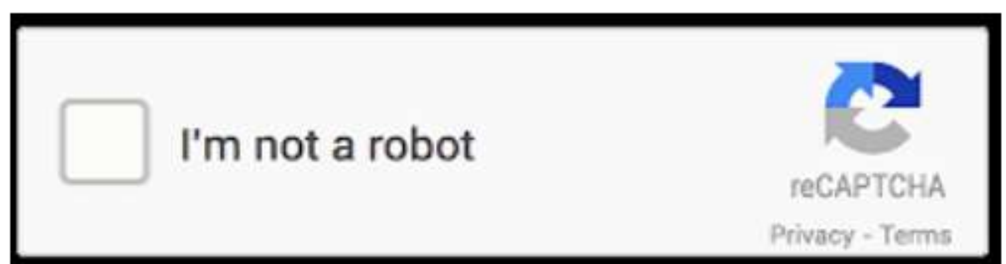


Perhaps we are going in the right direction?

Alan Turing

1950: Can a computer convince a human that it is not a computer but a real person.

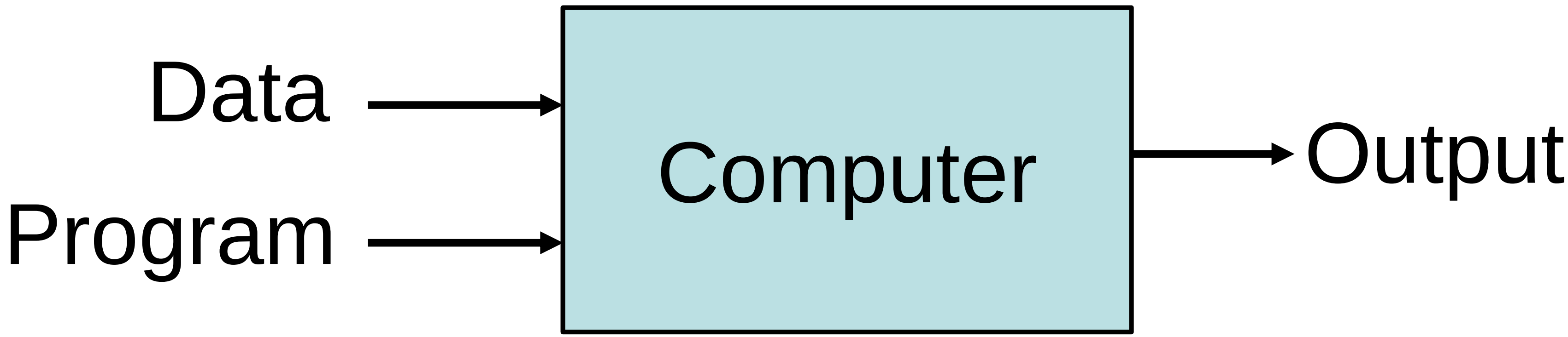
Now: Can a human convince a computer that he is a real person, not a computer



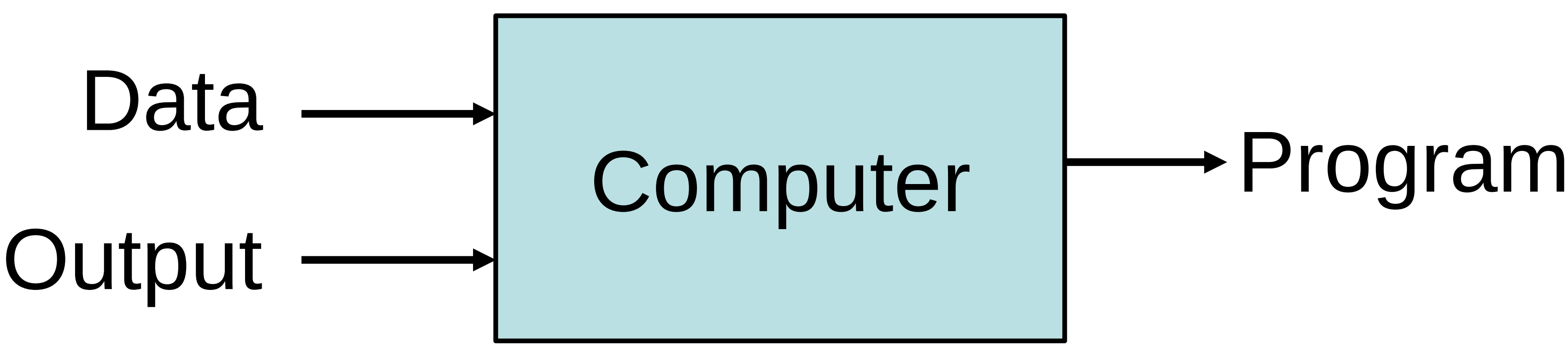
What is machine learning (ML)?

- Many AI researchers consider the ultimate goal of AI can be achieved by imitating the way humans learn
- **Machine Learning** – is the scientific study of algorithms and statistical models that computer systems use to learn from observations, without being explicitly programmed
- In this context, **learning** refers to:
  - recognizing complex patterns in data
  - making intelligent decisions based on data observations

Classic Programming



Machine Learning



A well-posed machine learning problem

- What problems can be solved* with machine learning?
- **Well-posed machine learning problem:**

"A computer program is said to learn from experience **E** with respect to some class of tasks **T** and performance measure **P**, if its performance at tasks in **T**, as measured by **P**, improves with experience **E**." – Tom Mitchell

(*) implies a certain degree of accuracy

A well-posed machine learning problem

- Arthur Samuel (1959) wrote a program for playing checkers (perhaps the first program based on the concept of learning, as defined by Tom Mitchell)
- The program played 10K games against itself
- The program was designed to find the good and bad positions on the board from the current state, based on the probability of winning or losing
- In this example:
  - ➢ E = 10000 games
  - ➢ T = play checkers
  - ➢ P = win or lose

Strong AI versus Weak AI

- Strong / generic / true AI / AGI

(see the Turing test and its extensions)

- Weak / narrow AI

(focuses on a specific well-posed problem)

When do we use machine learning?

- We use ML when it is hard (impossible) to define a set of rules by hand / to write a program based on explicit rules

- Examples of tasks that be solved through machine learning:
  - ➢ face detection
  - ➢ speech recognition
  - ➢ stock price prediction
  - ➢ object recognition

The essence of machine learning

- A pattern exists
- We cannot express it programmatically
- We have data on it


What is machine learning?

[Arthur Samuel, 1959] field of study that:

- gives computers the ability to learn without being explicitly programmed

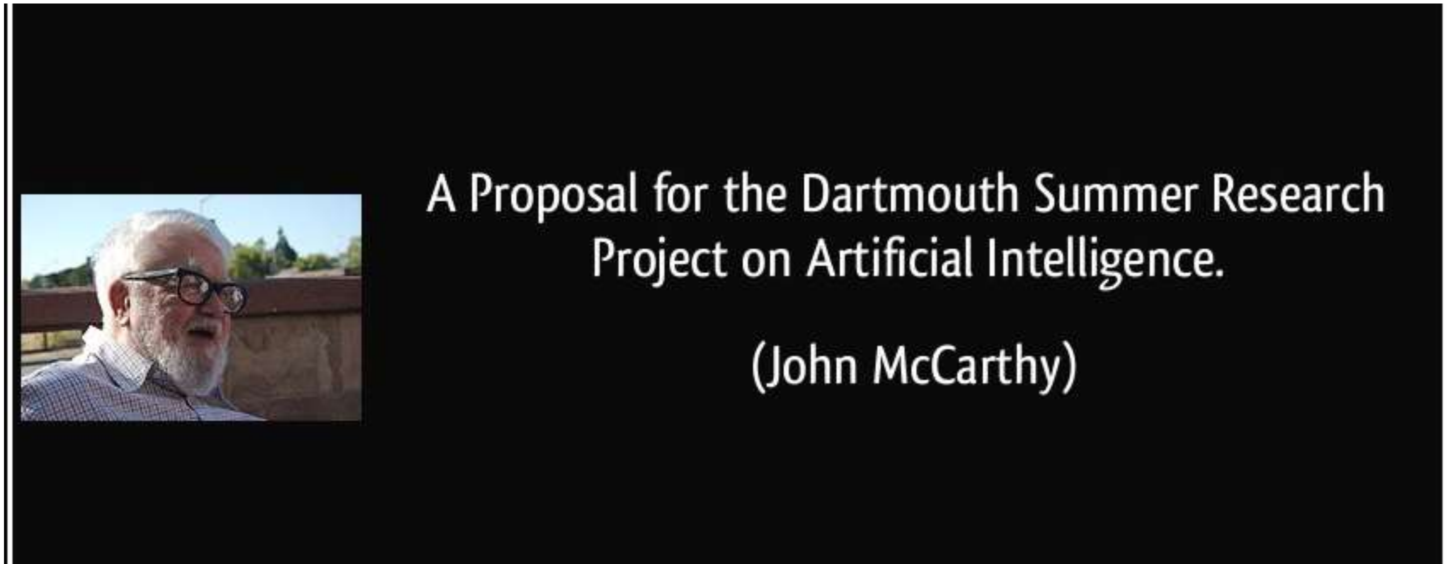
[Kevin Murphy] algorithms that:

- automatically detect patterns in data
- use the uncovered patterns to predict future data or other outcomes of interest

[Tom Mitchell] algorithms that:

- improve their performance (P)
- at some task (T)
- with experience (E)

Brief history of AI



Brief history of AI

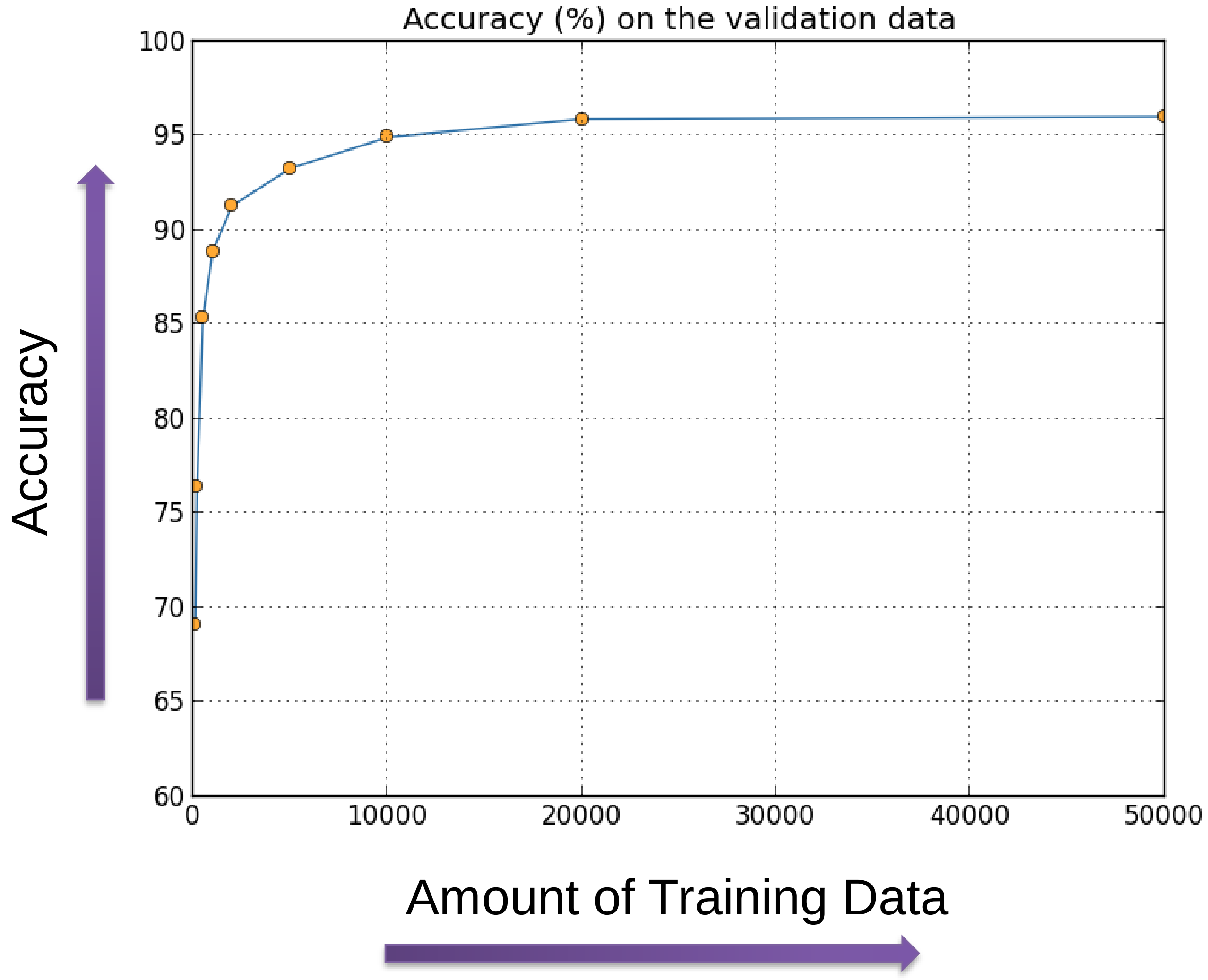
- “We propose that a 2 month, 10 man study of artificial intelligence be carried out during the summer of 1956 at Dartmouth College in Hanover, New Hampshire.”
- The study is to proceed on the basis of the conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it.
- An attempt will be made to find how to make machines use language, form abstractions and concepts, solve kinds of problems now reserved for humans, and improve themselves.
- We think that a significant advance can be made in one or more of these problems if a carefully selected group of scientists work on it together for a summer.”

Brief history of AI

- 1960-1980s: ”AI Winter”
- 1990s: Neural networks dominate, essentially because of the discovery of the backpropagation for training neural networks with two or more layers
- 2000s: Kernel methods dominate, essentially because of the instability of training neural networks
- 2010s: The comeback of neural networks, essentially because of the discovery of deep learning

Why are things working today?

- More compute power
- More data
- Better algorithms / models



ML in a nutshell

- Tens of thousands of machine learning algorithms
  - ➢ Researchers publish hundreds new every year

- Decades of ML research oversimplified:
  - ➢ Learn a mapping f from the input X to the output Y, i.e.:

$$f : X \to Y$$

  - ➢ Example: X are emails, Y: {spam, not-spam}

ML in a nutshell

- Input: X (images, texts, emails…)
- Output: Y (spam or not-spam…)
- (Unknown) Target Function:

$f: X \to Y$ (the “true” mapping / reality)

- Data

$(x_1, y_1), (x_2, y_2), \ldots (x_N, y_N)$

- Model / Hypothesis Class

$g: X \to Y$

$y = g(x) = sign(w^T x)$

ML in a nutshell

- Every machine learning algorithm has three components:
  - Representation / Model Class
  - Evaluation / Objective Function
  - Optimization

Where does ML fit in?

Machine Learning

Biology Neuroscience

- Biology of learning
- Inspiring paradigms
- E.g.: neural networks

Applied Maths

- Optimization
- Linear algebra
- Derivatives
- E.g.: local minimum

Computer Science

- Algorithms
- Data structures
- Complexity analysis
- E.g.: k-d trees

Statistics

- Estimation techniques
- Theoretical frameworks
- Optimality, efficiency
- E.g.: Bayes rule

Learning paradigms

- Standard learning paradigms:
  - ➢ Supervised learning
  - ➢ Unsupervised learning
  - ➢ Semi-supervised learning
  - ➢ Reinforcement learning

- Non-standard paradigms:
  - ➢ Active learning
  - ➢ Transfer learning
  - ➢ Transductive learning

Supervised learning

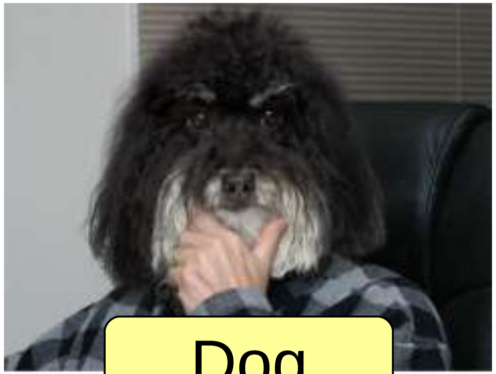
- We have a set of labeled training samples
- Example 1: object recognition in images annotated with corresponding class labels

Car

Car

Person

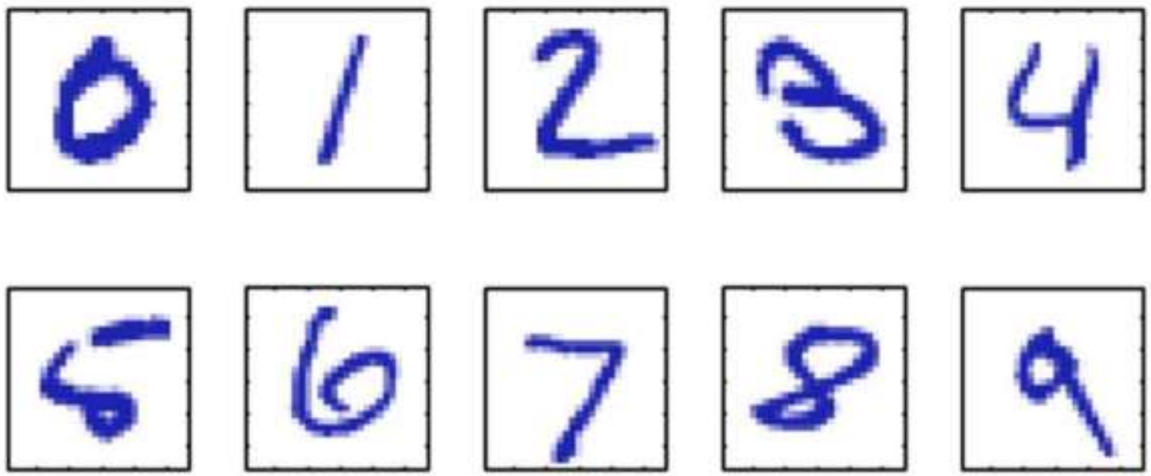
Person



Dog

Supervised learning

- Example 2: handwritten digit recognition (on the MNIST data set)

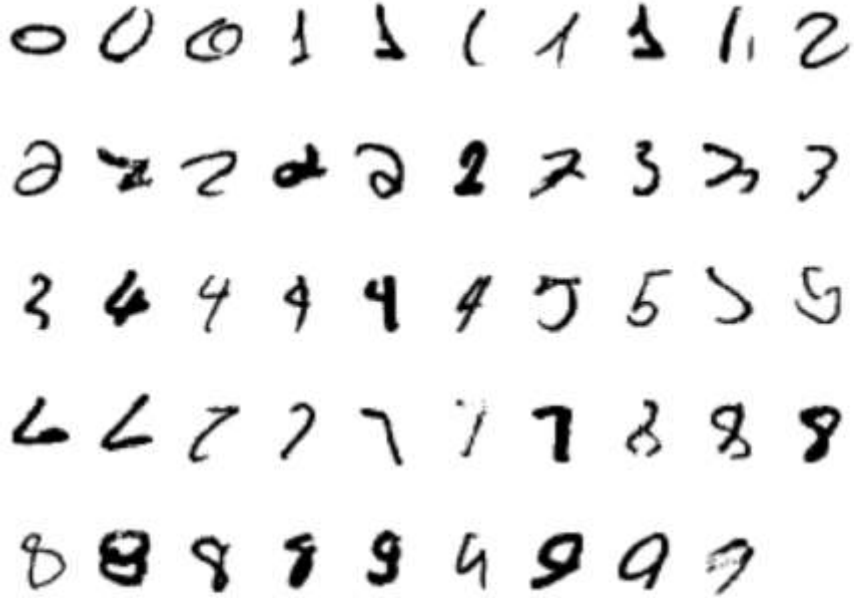


- Images of 28 x 28 pixels
- We can represent each image as a vector x of 784 components
- We train a classifier $f(x)$ such that:

$$f : x \rightarrow \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

Supervised learning

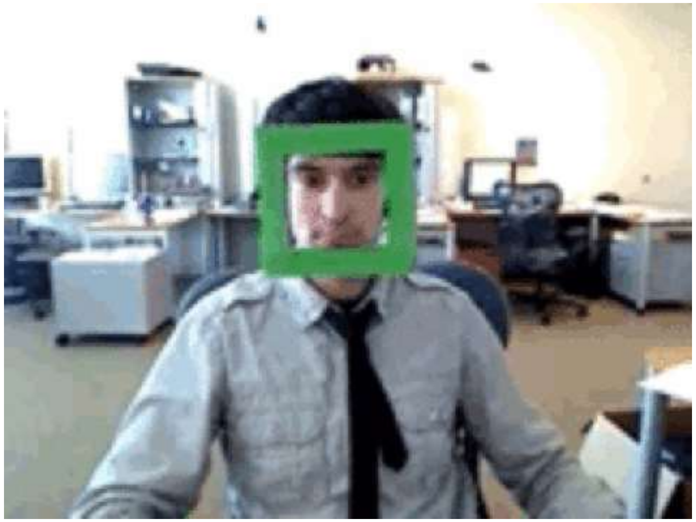
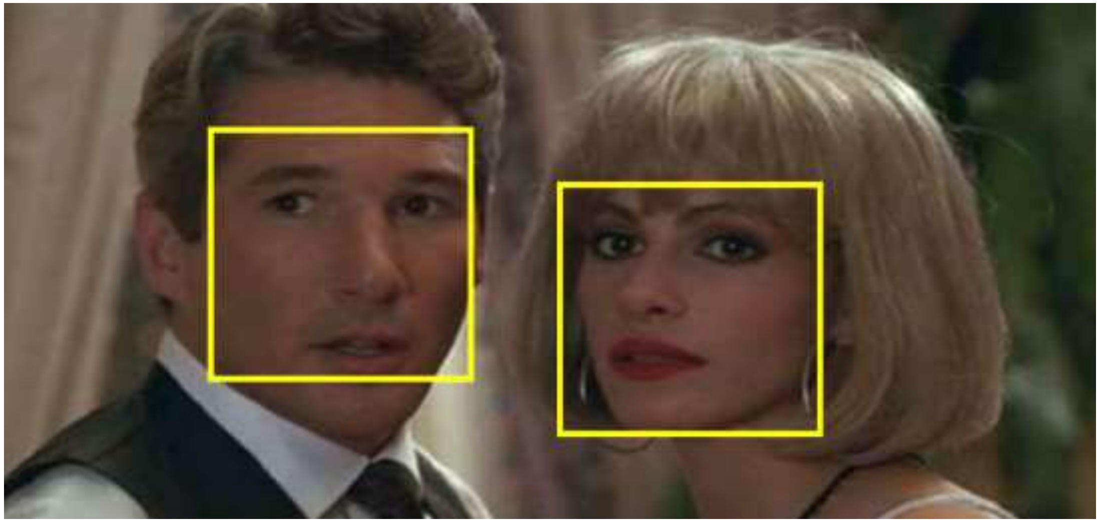
- Example 2 (continued): handwritten digit recognition (on the MNIST data set)



- Starting with a training set of about 60K images (about 6000 images per class)
- … the error rate can go down to 0.23% (using convolutional neural networks)
- Among the first (learning-based) systems used in a large-scale commercial setting for postal code and bank cheque processing

Supervised learning

- Example 3: face detection

- One approach consists of sliding a window over the image
- The goal is to classify each window into one of the two possible classes: face or not-face
- The original problem is transformed into a classification problem

Supervised learning

- Example 3: face detection

- We start with a set of face images with different variations such as age, gender, illumination, pose, but no translations
- … and a larger set of images that do not contain full faces

Supervised learning

- Example 4: spam detection

> **rama rama ramaumar002@hotmail.com** via yahoo.com
> to
>
> From: Mrs. Rama Umar
> Groupe Bank of Africa (Annexe) Burkina Faso
> Foreign Department Operation.
>
> My name is Mrs.Rama Umar. I am working with Bank of Africa here in Burkina Fa
> late foreign customer.
>
> When I discovered that there had been neither deposits nor withdrawals from this ac
> none of the family member or relations of the late person are aware of this account,
> (Five Million USA Dollars).

- The task is to classify an email into spam or not-spam
- The occurrence of the word “Dollars” is a good indicator of spam
- A possible representation is a vector of word frequencies

We count the words…

obtaining X

rama rama ramaumar002@hotmail.com via yahoo.com
to

From: Mrs. Rama Umar
Groupe Bank of Africa (Annexe) Burkina Faso
Foreign Department Operation.

My name is Mrs.Rama Umar. I am working with Bank of Africa here in Burkina Fa
late foreign customer.

When I discovered that there had been neither deposits nor withdrawals from this ac
none of the family member or relations of the late person are aware of this account,
(Five Million USA Dollars).

$$\begin{pmatrix} \text{free} & 100 \\ \text{money} & 2 \\ \vdots & \vdots \\ \text{account} & 2 \\ \vdots & \vdots \end{pmatrix}$$

Yoshua Bengio <yoshua.bengio@gmail.com>
to Dong-Hyun, Ian, Dumitru, Pierre, Aaron, Mehdi, Ben, Will, Charlie,

Nice slides!

See you next week,

—Yoshua

$$\begin{pmatrix} \text{free} & 1 \\ \text{money} & 1 \\ \vdots & \vdots \\ \text{account} & 2 \\ \vdots & \vdots \end{pmatrix}$$

The spam detection algorithm

$$\begin{pmatrix} \text{free} & 100 \\ \text{money} & 2 \\ \vdots & \vdots \\ \text{account} & 2 \\ \vdots & \vdots \end{pmatrix}$$

Why these words?

$$\begin{pmatrix} 100 \times 0.2 \\ 2 \times 0.3 \\ \vdots \\ 2 \times 0.3 \\ \vdots \end{pmatrix} = 3.2$$

$$\begin{pmatrix} 100 \times 0.01 \\ 2 \times 0.02 \\ \vdots \\ 2 \times 0.01 \\ \vdots \end{pmatrix} = 1.03$$

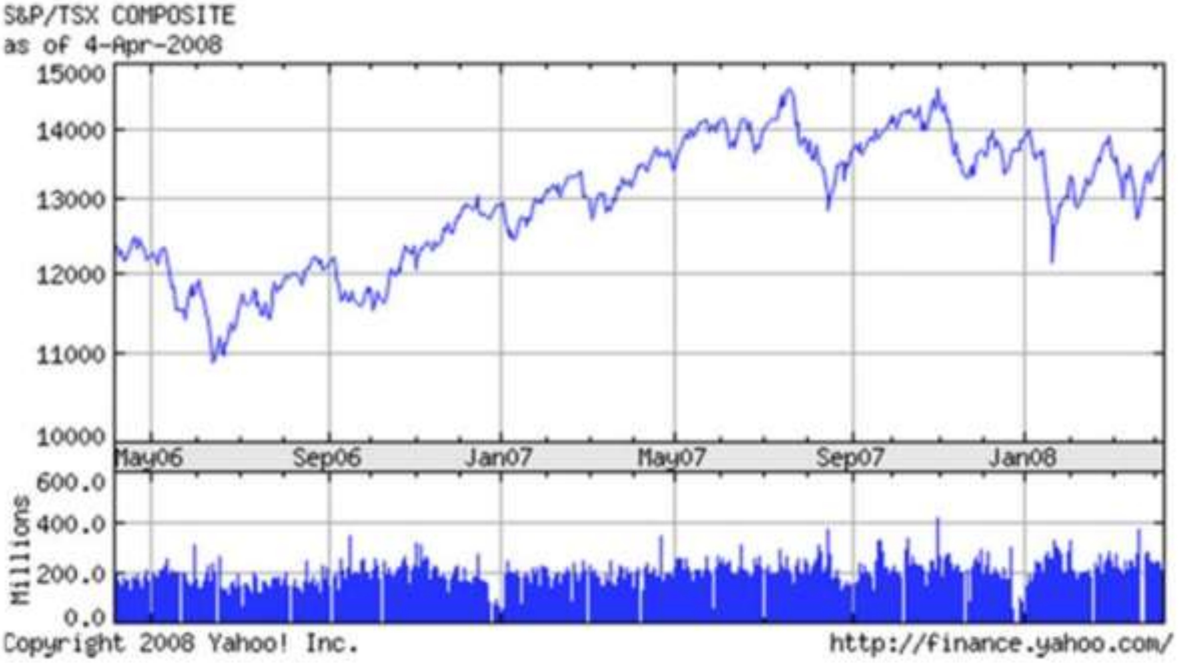
Why linear combination?

Where do the weights come from?

Confidence / performance guarantee?

Supervised learning

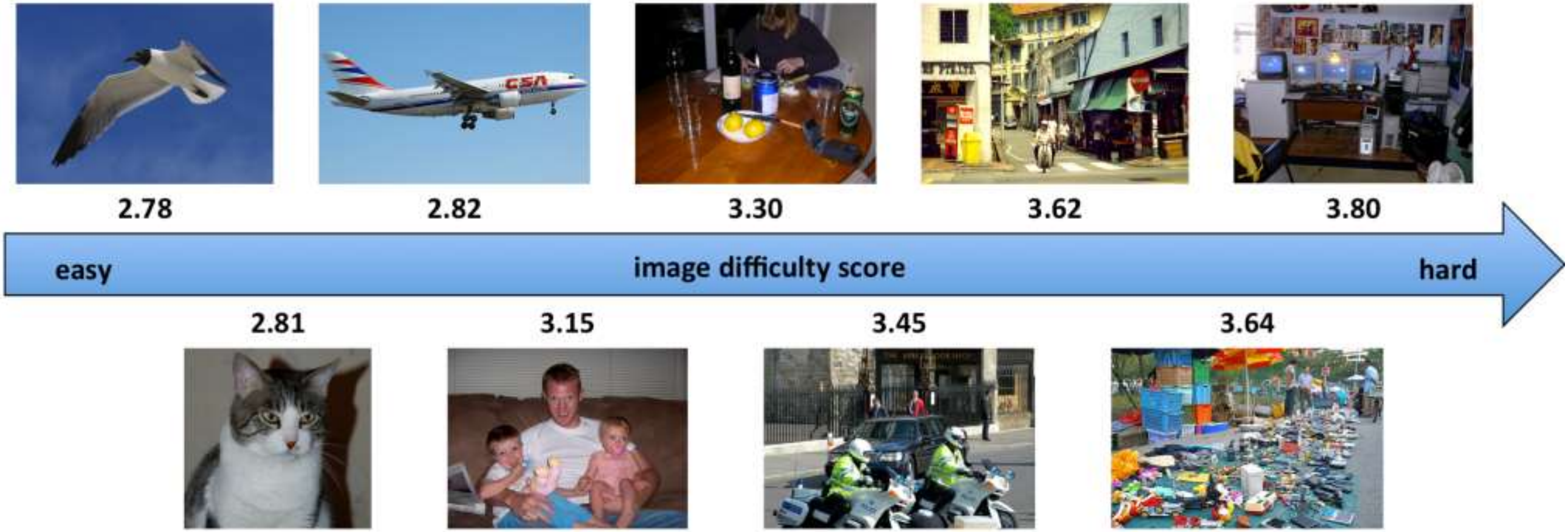
- Example 5: predicting stock prices on the market



- The goal is to predict the price at a future date, for example in a few days
- This is a regression task, since the output is continuous

Supervised learning

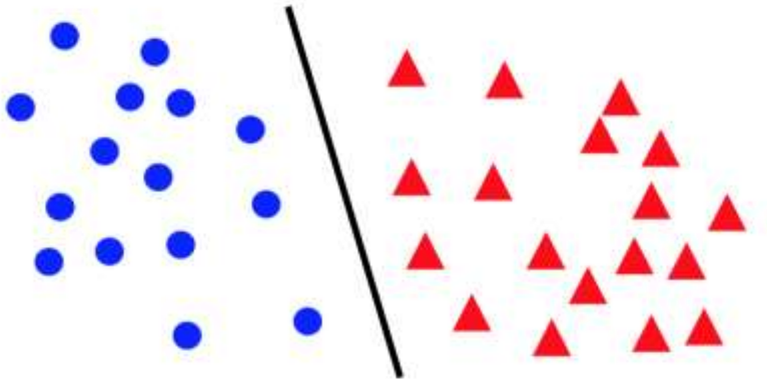
- Example 6: image difficulty prediction [Ionescu et al. CVPR2016]



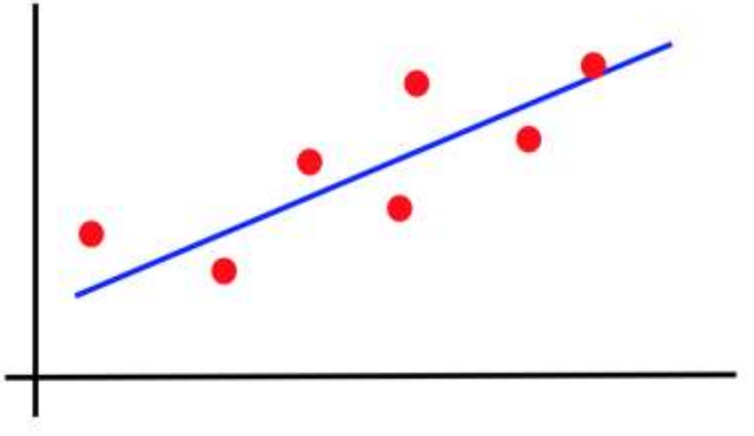
- The goal is to predict the time necessary for a human to solve a visual search task (data set available for project 2!)
- This is a regression task, since the output is continuous

Canonical forms of supervised learning problems

- Classification

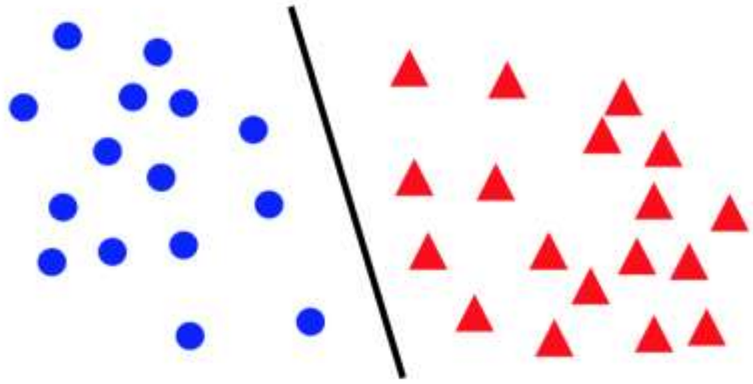


- Regression



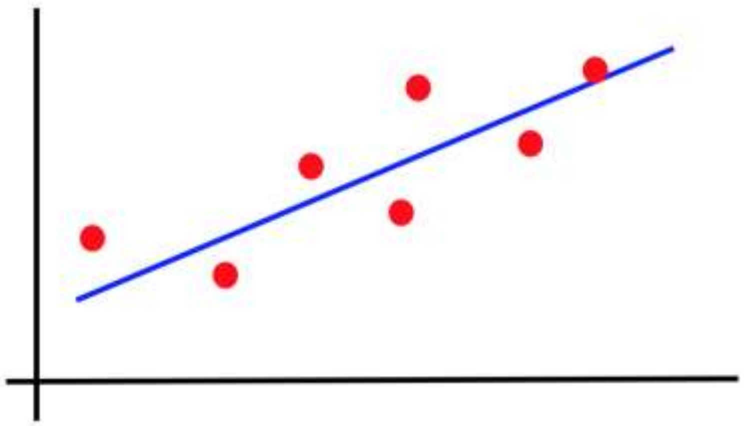
Age estimation in images

- Classification?



What age?

- Regression?



The supervised learning paradigm

Functions $\mathcal{F}$

$f : \mathcal{X} \to \mathcal{Y}$

Training data

$\{(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}\}$

LEARNING

find $\hat{f} \in \mathcal{F}$
s.t. $y_i \approx \hat{f}(x_i)$

Learning machine

New data

PREDICTION

$y = \hat{f}(x)$ $\Leftarrow$ x

Supervised learning models

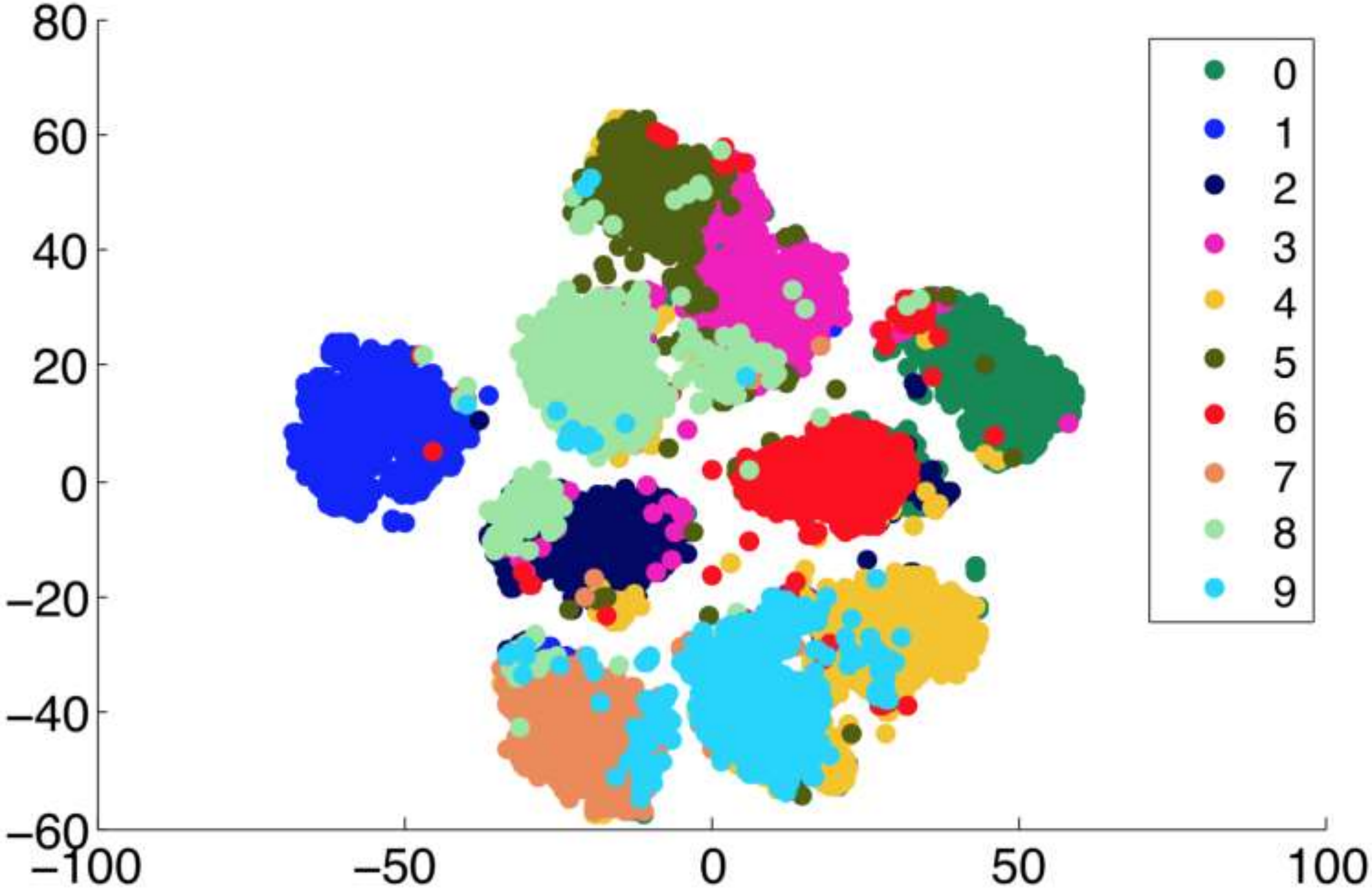
- Naive Bayes (lecture 2)
- k-Nearest Neighbors (lecture 3)
- Decision trees and random forests (lecture 4)
- Support Vector Machines (lecture 5, 6)
- Kernel methods (lecture 5)
- Kernel Ridge Regression (lecture 5)
- Neural networks (lectures 7, 8, 9)
- Many others…

Unsupervised learning

- We have an unlabeled training set of samples
- Example 1: clustering images based on similarity

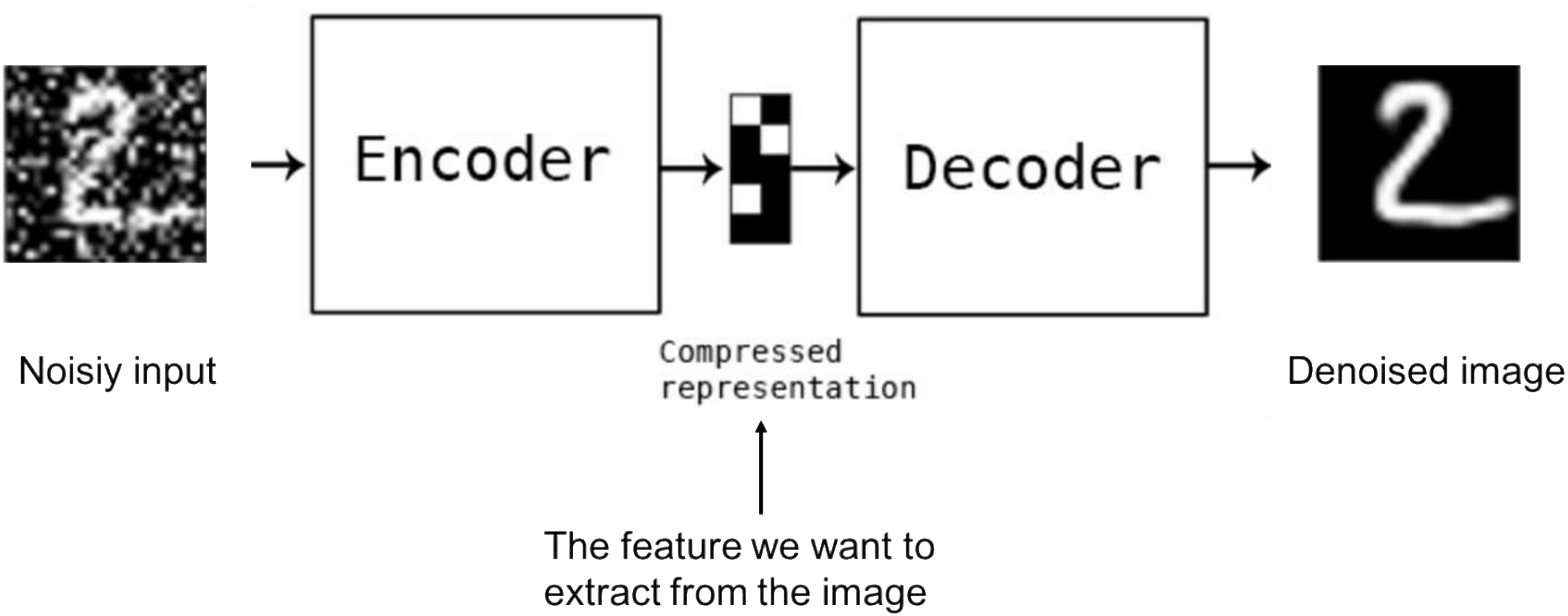
Unsupervised learning

- Example 1: clustering MNIST images based on similarity [Georgescu et al. ICIP2019]



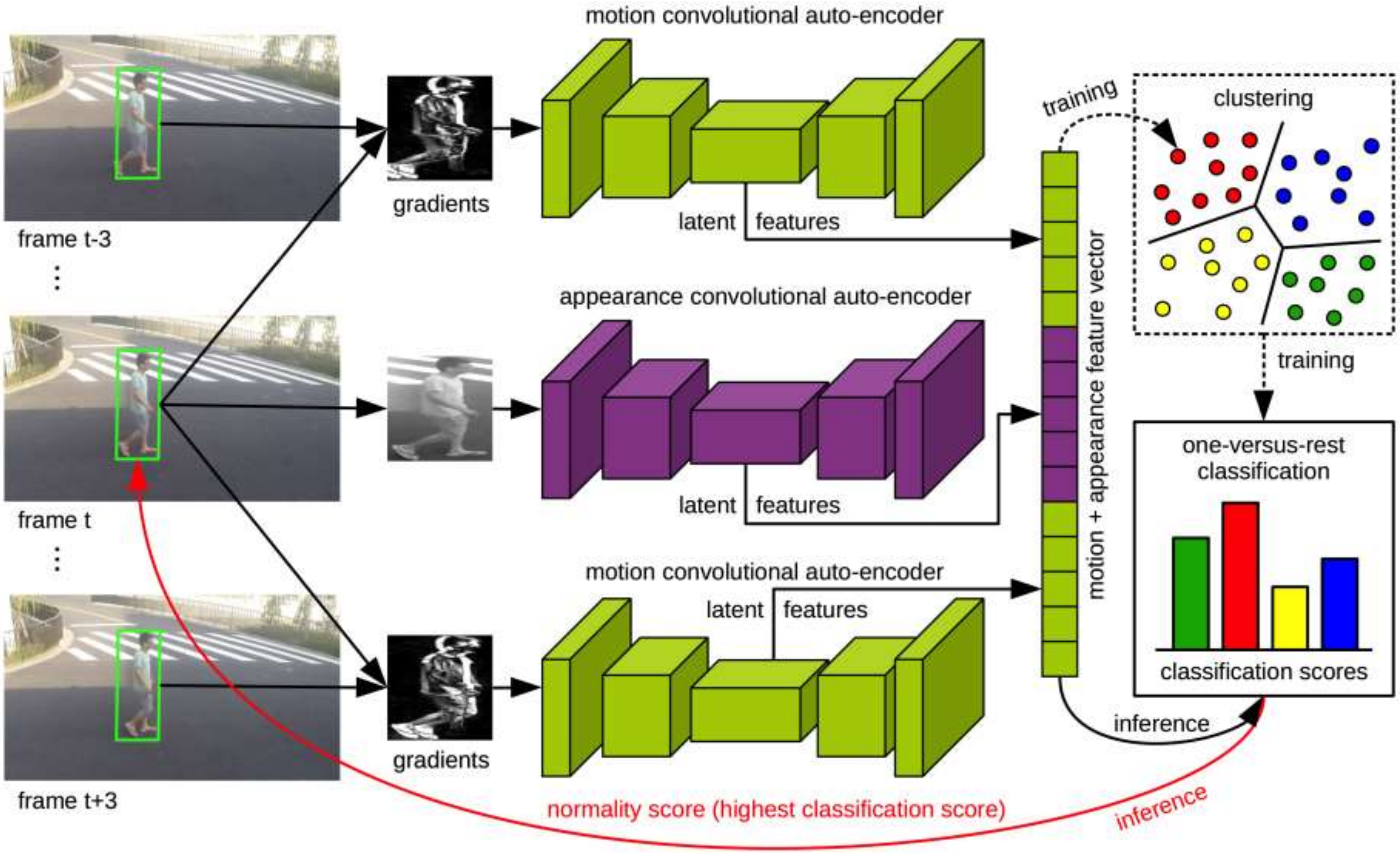
Unsupervised learning

- Example 2: unsupervised features learning



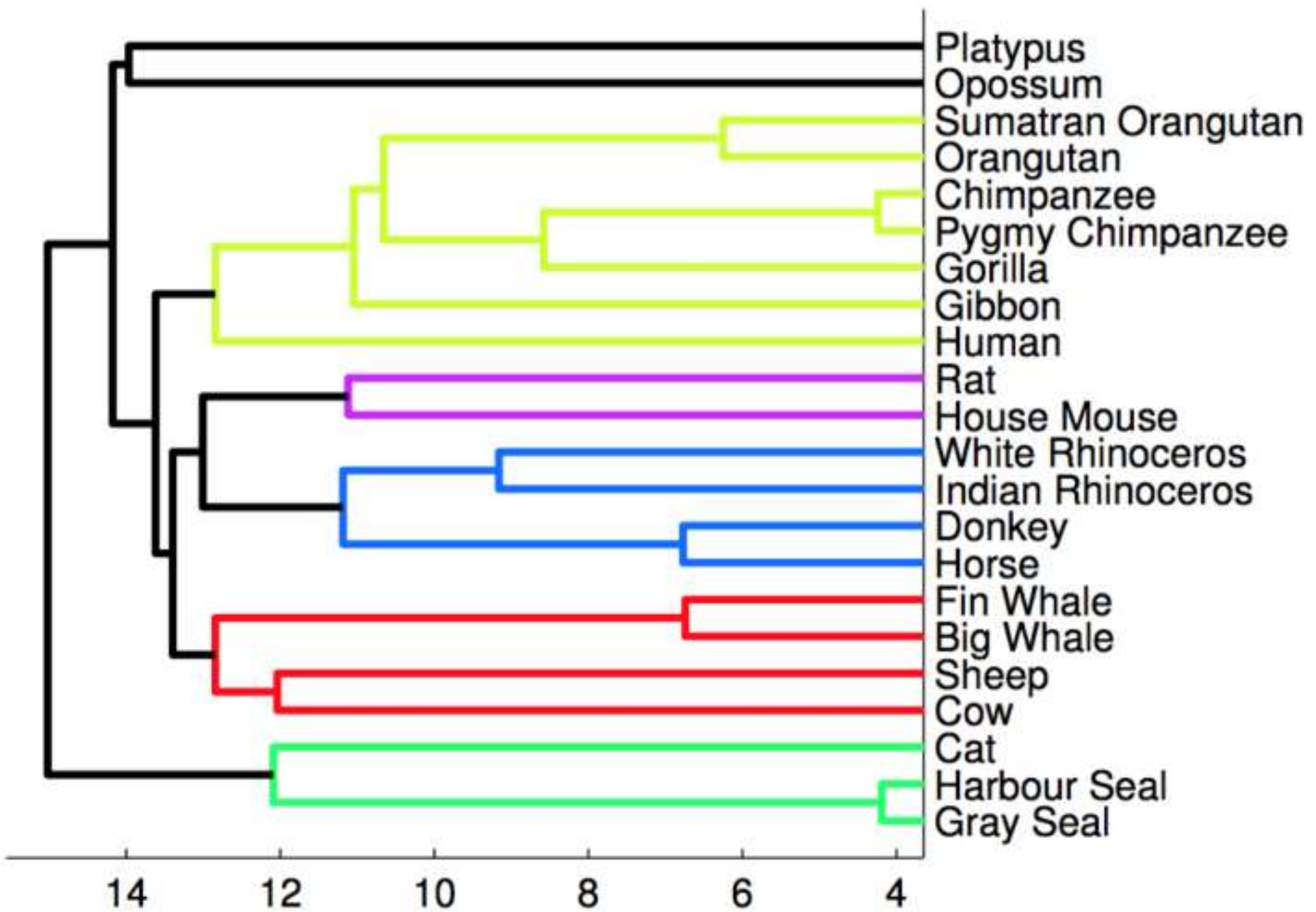
Unsupervised learning

- Example 2: unsupervised features learning for abnormal event detection [Ionescu et al. CVPR2019]



Unsupervised learning

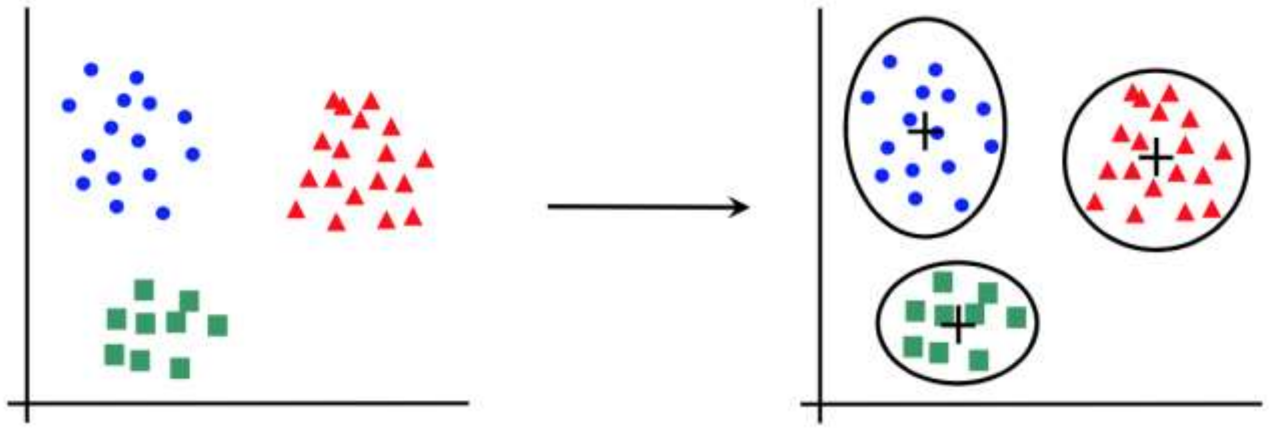
- Example 3: clustering mammals by family, species, etc.



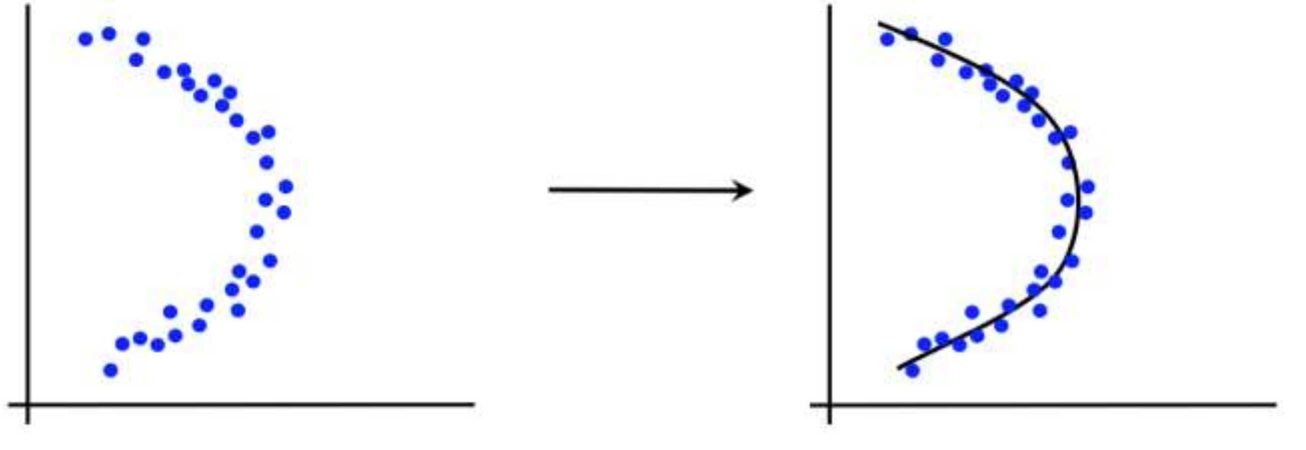
- The task is to generate the phylogenetic tree based on DNA

Canonical forms of unsupervised learning problems

- Clustering



- Dimensionality Reduction



Unsupervised learning models

- K-means clustering (lecture 10, 11)
- DBScan (lecture 12)
- Hierarchical clustering (lecture 12)
- Principal Component Analysis (lecture 13)
- t-Distributed Stochastic Neighbor Embedding (lecture 13)
- Hidden Markov Models
- Many others…

Semi-supervised learning

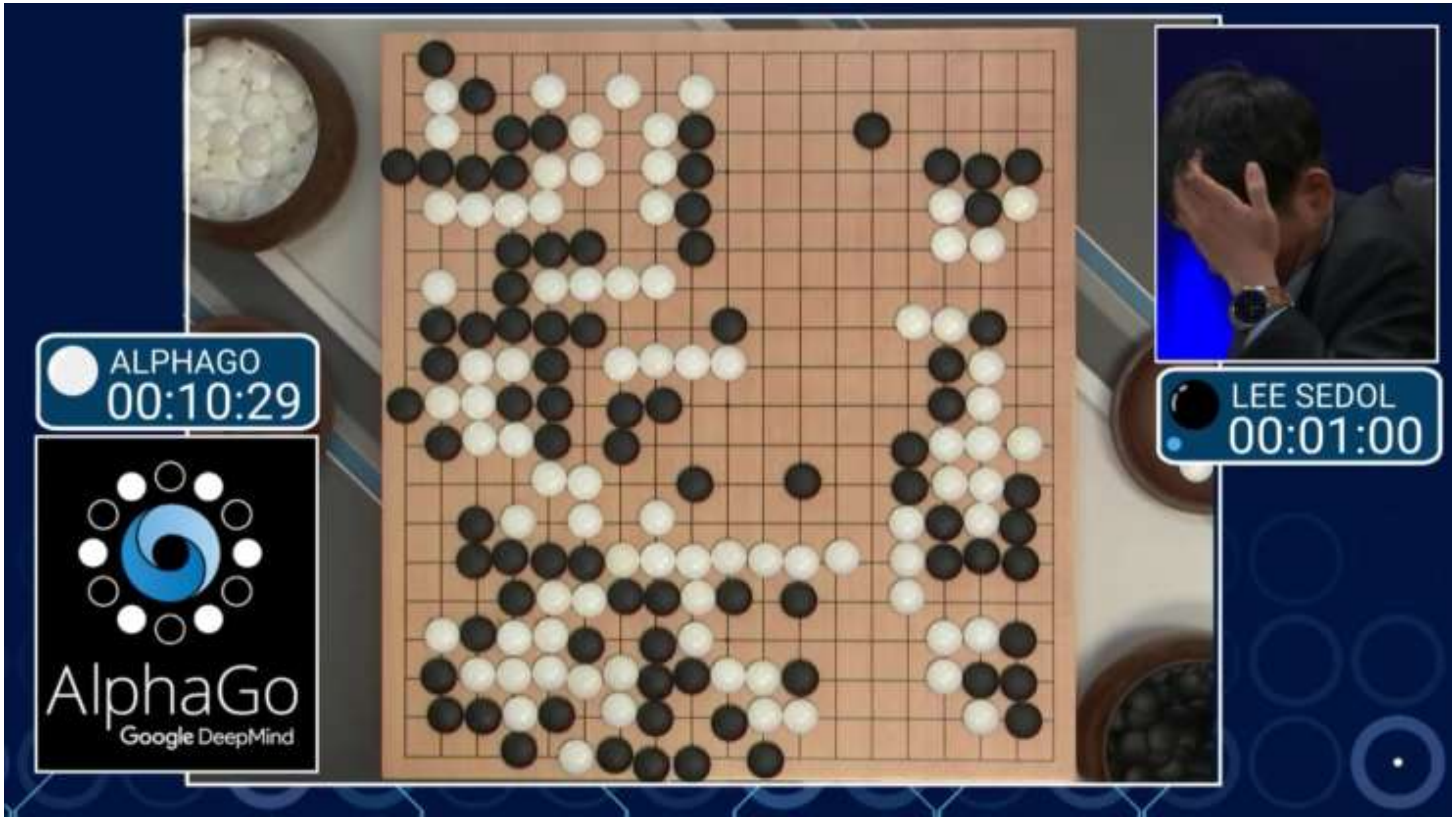
- We have a training set of samples that are partially annotated with class labels
- Example 1: object recognition in images, some of which are annotated with corresponding class labels

Reinforcement learning

- How does it work?
- The system learns intelligent behavior using a reinforcement signal (reward)
- The reward is given after several actions are taken (it does not come after every action)
- Time matters (data is sequential, not i.i.d.)
- The actions of the system can influence the data

Reinforcement learning

- Example 1: learning to play Go
- +/- reward for winning / losing the game

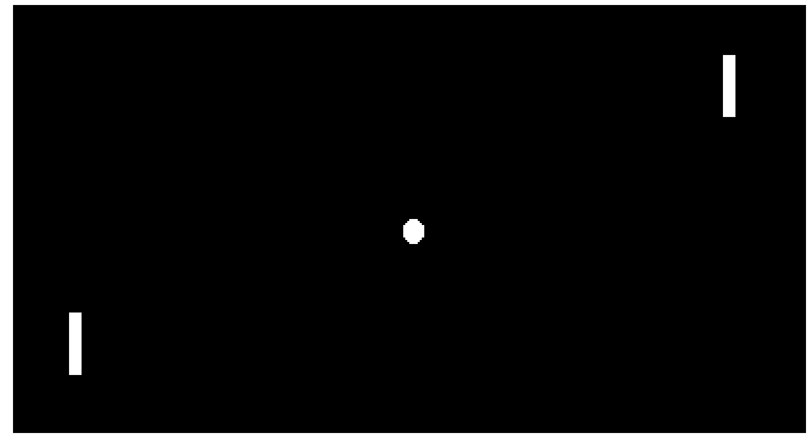


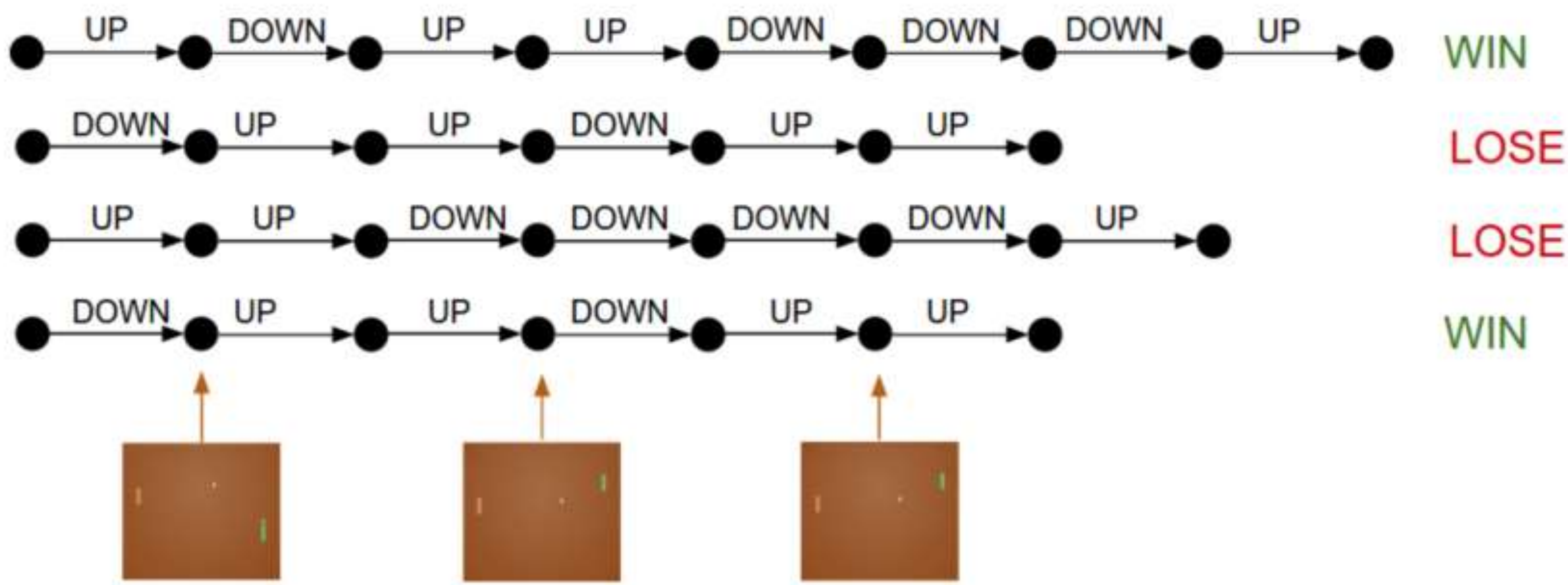
Reinforcement learning

- Example 2: teaching a robot to ride a bike
- +/- reward for moving forward / falling

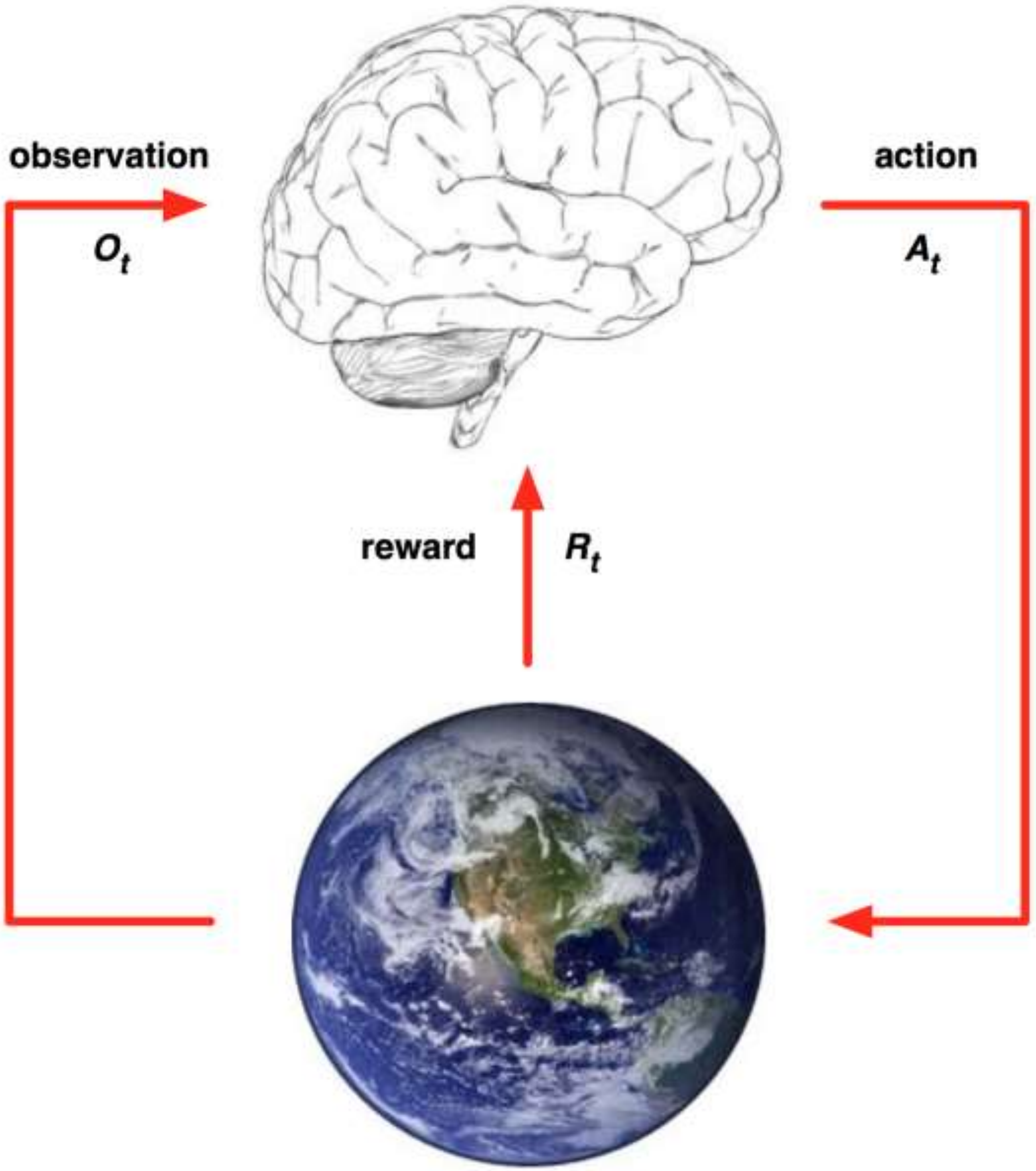
Reinforcement learning

- Example 3: learning to play Pong from image pixels
- +/- reward for increasing
- personal / adversary score

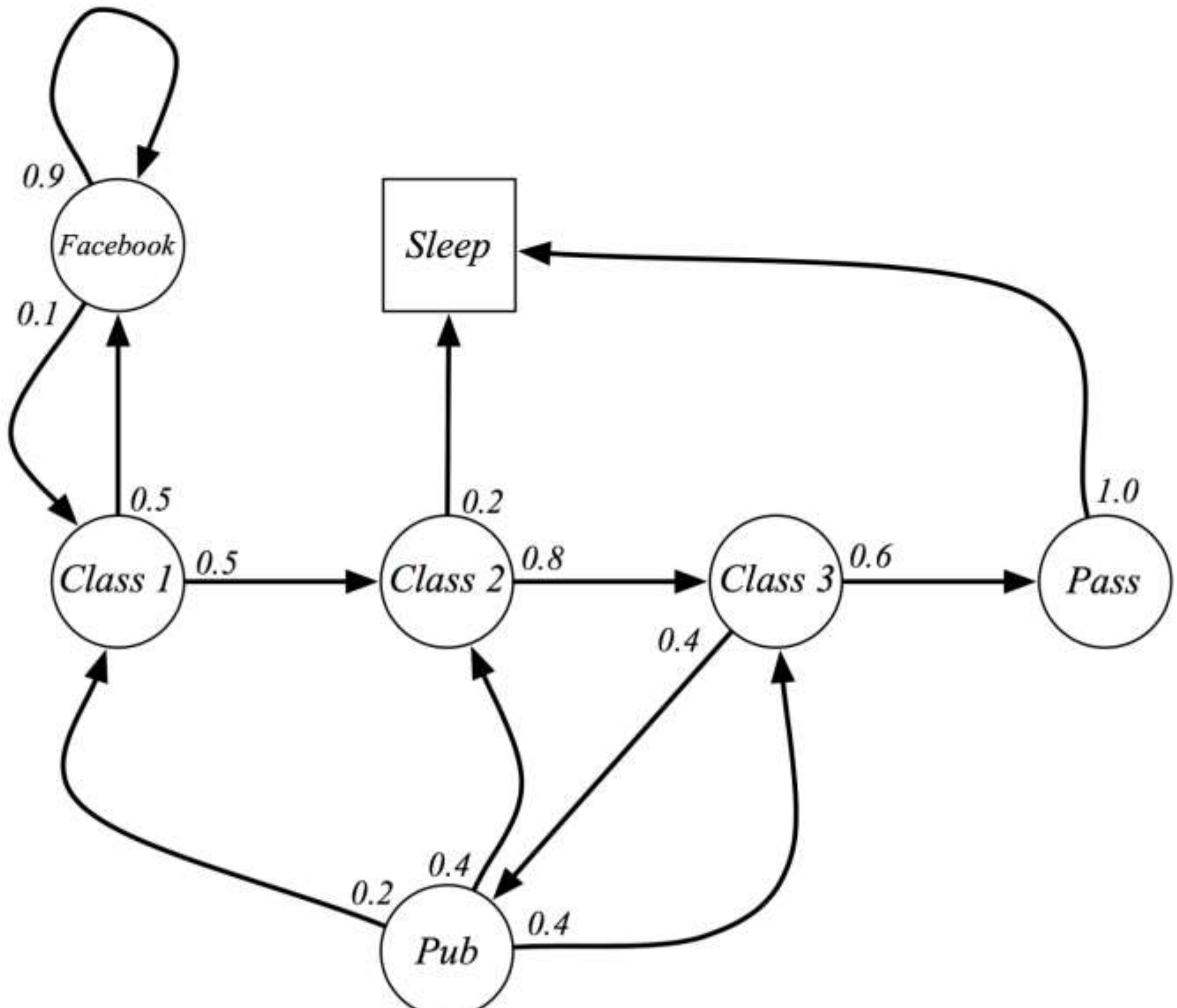




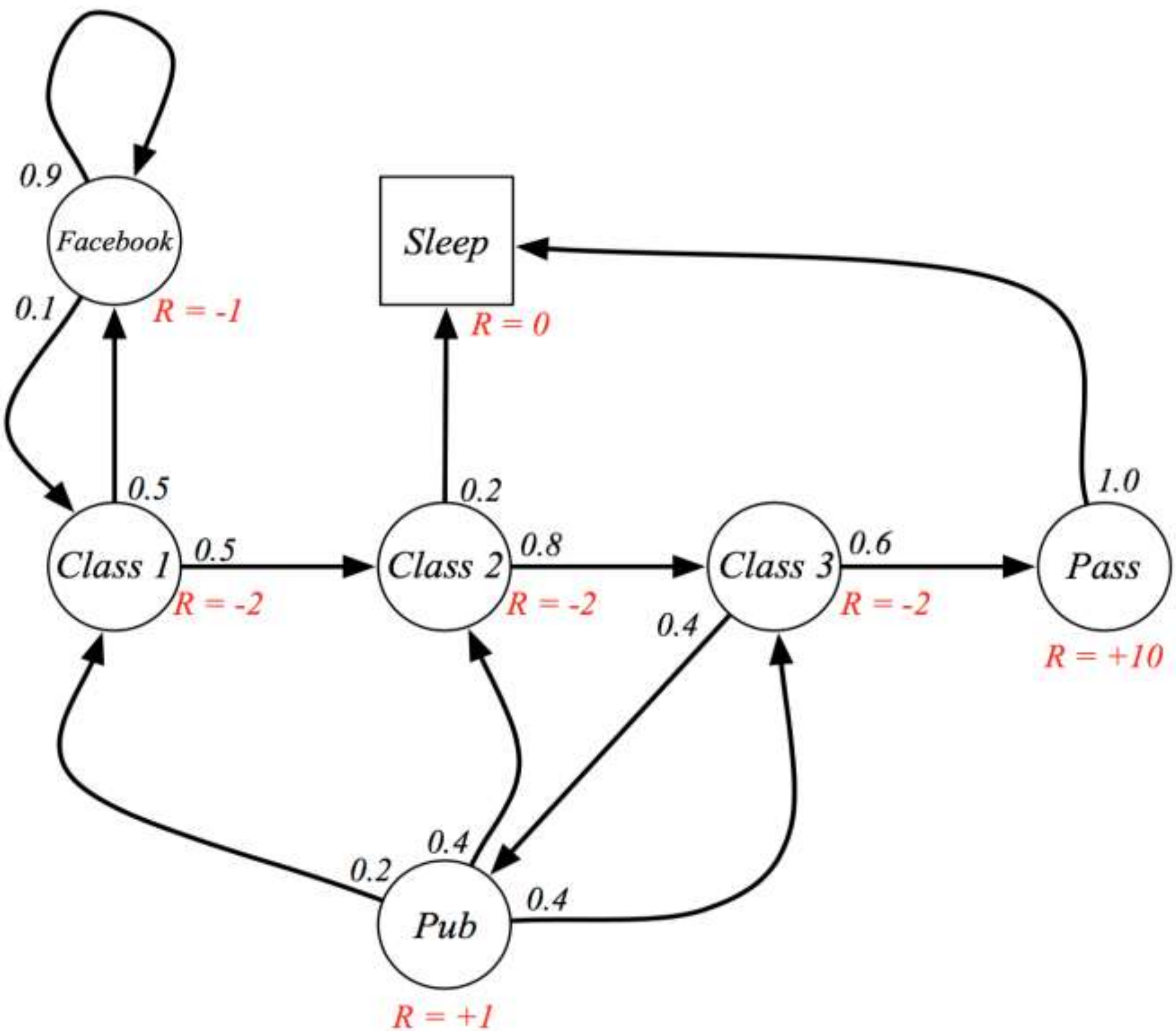
Reinforcement learning paradigm



Formalizing as Markov Decision Process



Formalizing as Markov Decision Process



Formalizing as Markov Decision Process

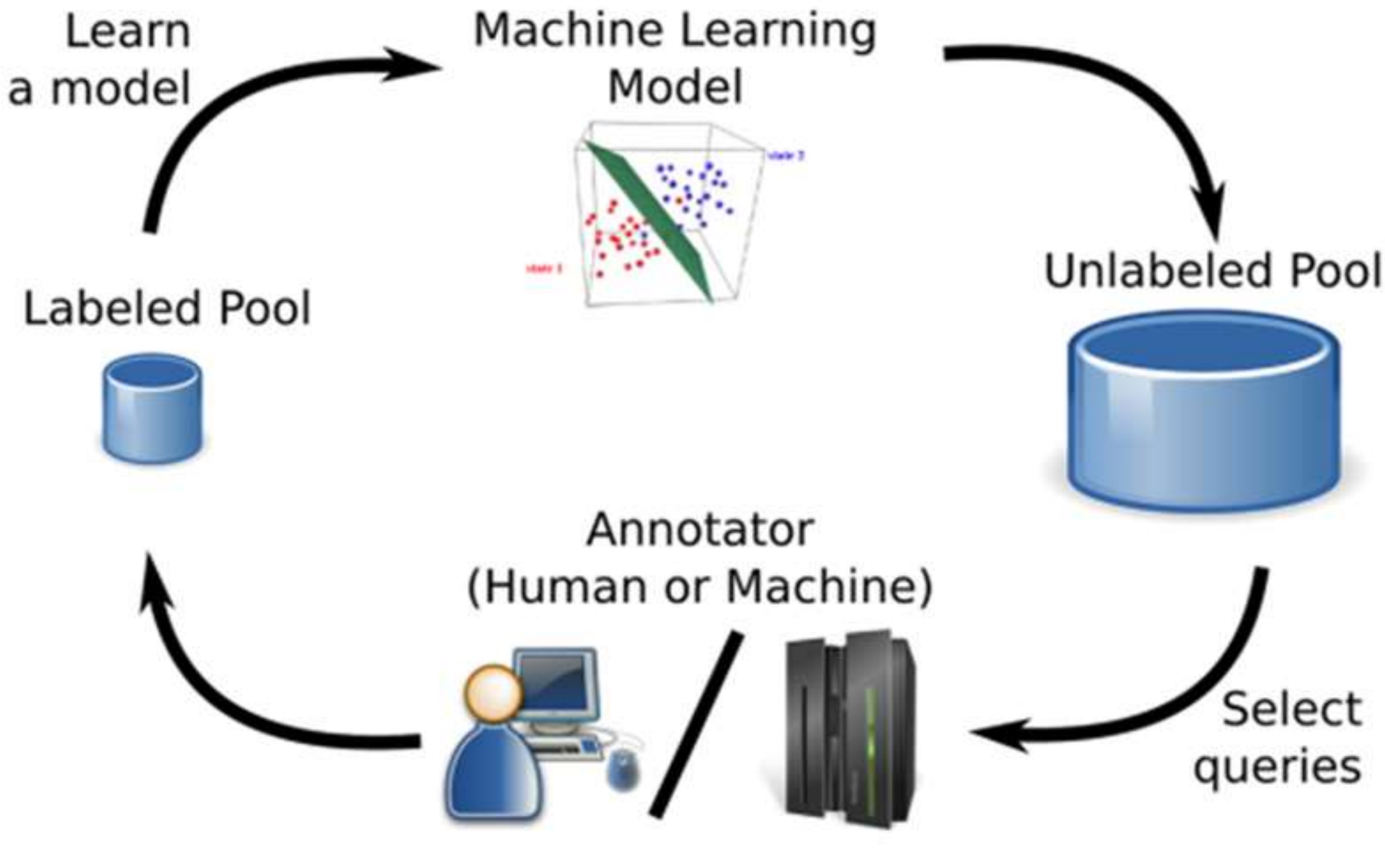
- Solution based on dynamic programming (small graphs) or approximation (large graphs)
- Goal: select the actions that maximize the total final reward
- The actions can have long-term consequences
- Sacrificing the immediate reward can lead to higher rewards on the long term

Formalizing as Markov Decision Process

- AlphaGo example:
  - Narrator 1: “That’s a very strange move”
  - Narrator 2: “I thought it was a mistake”
  - But actually, “the move turned the course of the game. AlphaGo went on to win Game Two, and at the post-game press conference, Lee Sedol was in shock.”
  - https://www.wired.com/2016/03/two-moves-alphago-lee-sedol-redefined-future/

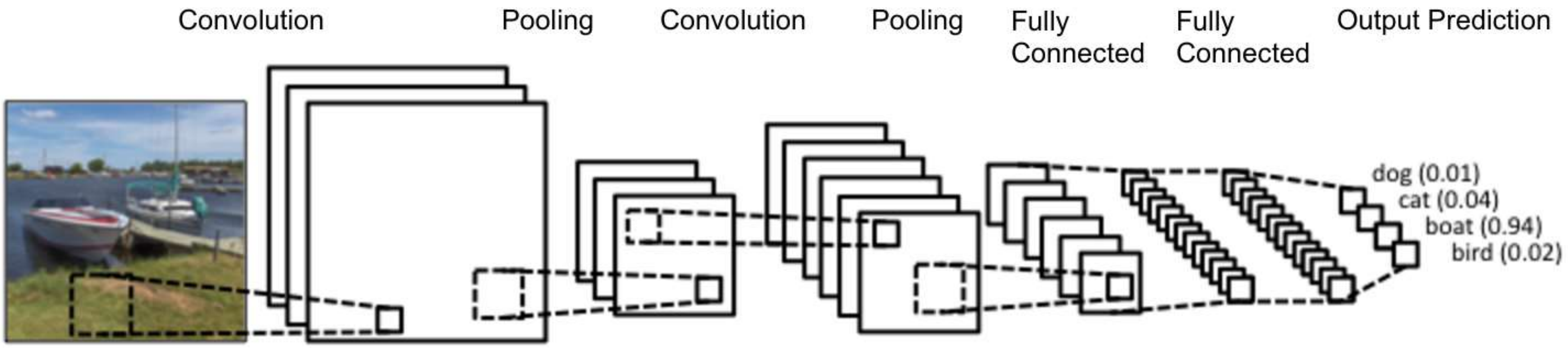
Active learning

- Given a large set of unlabeled samples, we have to choose a small subset for annotation in order to obtain a good classification model



Transfer learning

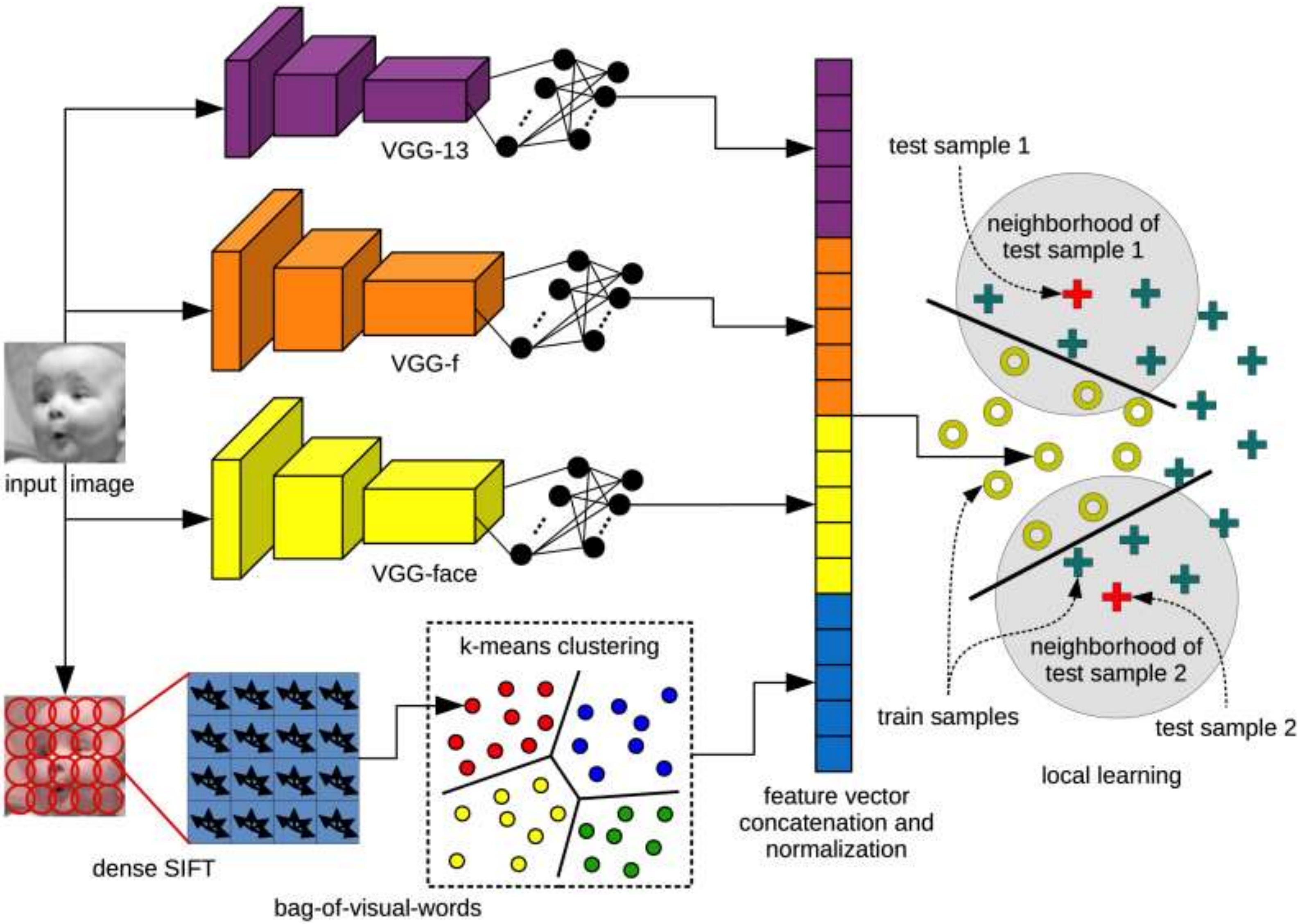
- Starting with a model trained for a certain task / domain, use the model for a different task / domain



More specific object classes,
face recognition,
texture classification, etc.

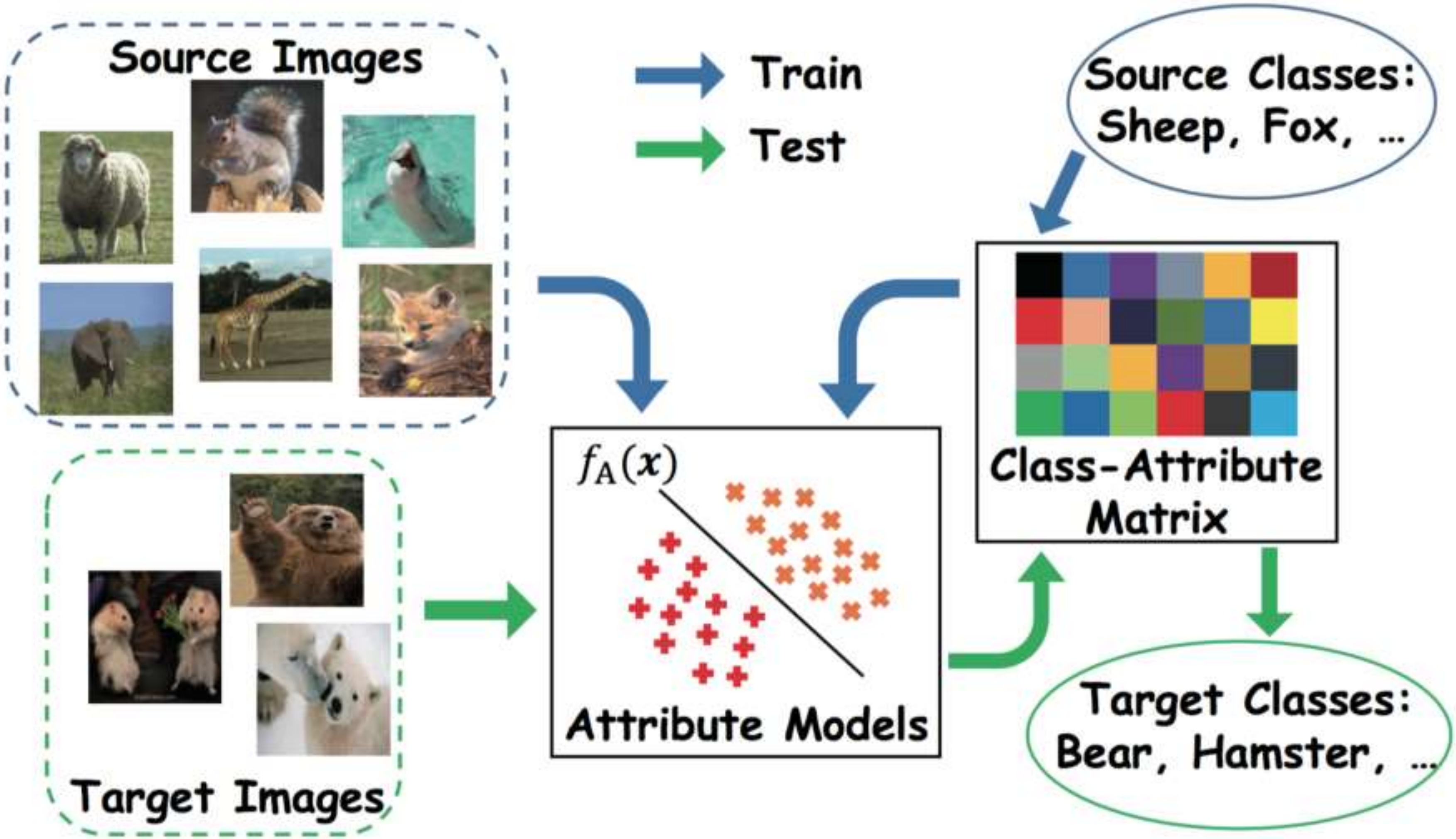
Transfer learning

- Adapt the model to specific test samples
- Example 1: facial expression recognition [Georgescu et al. Access2019]



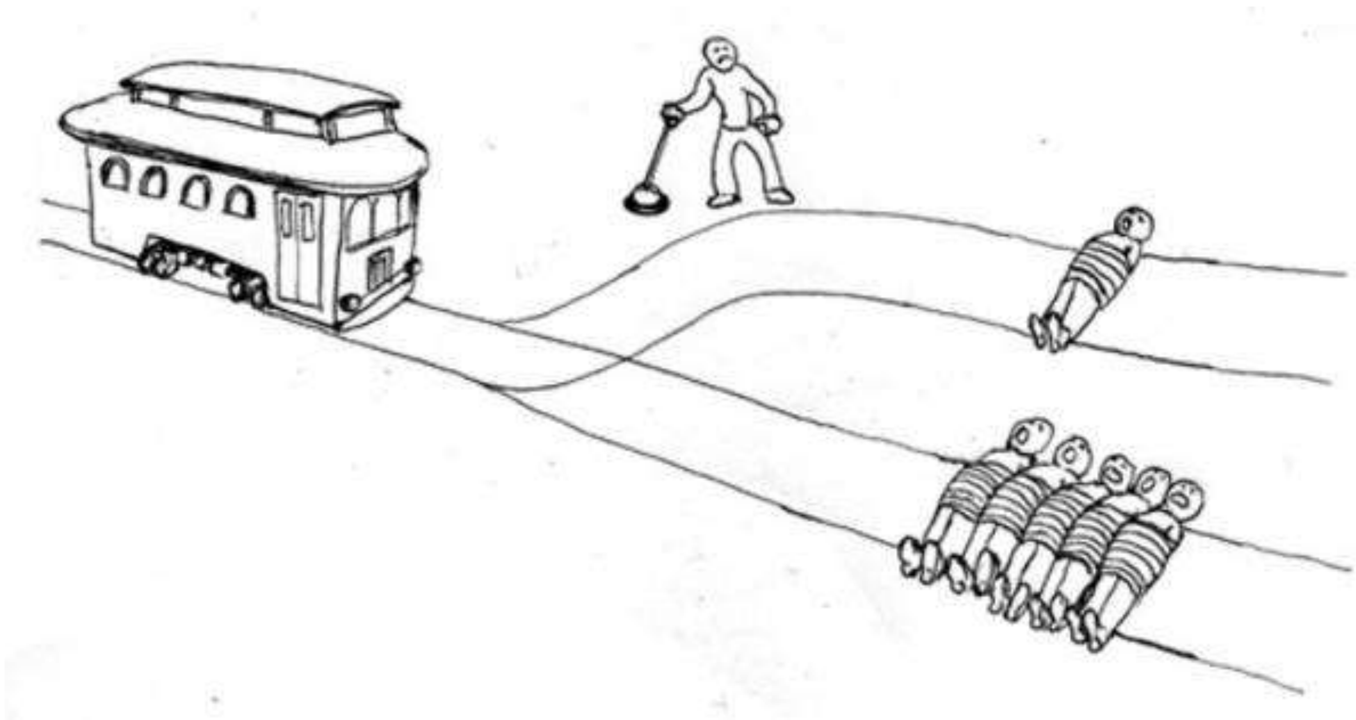
Transfer learning

- Example 2: zero-shot learning



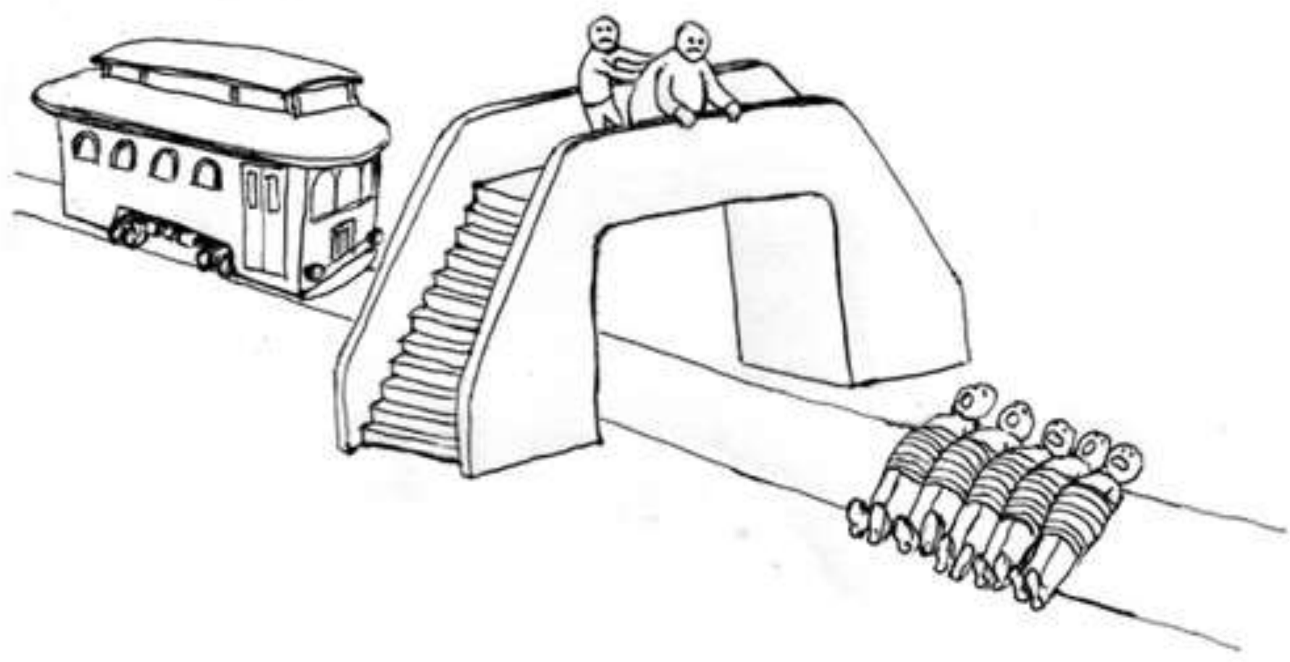
Many interesting applications, but…

- What is ethical and what is not?
- Trolley paradox



Many interesting applications, but…

- What is ethical and what is not?
- Trolley paradox



Many interesting applications, but…

- What is ethical and what is not?
- Trolley paradox
- http://moralmachine.mit.edu

Bibliography

Springer Series in Statistics

Trevor Hastie
Robert Tibshirani
Jerome Friedman

The Elements of Statistical Learning

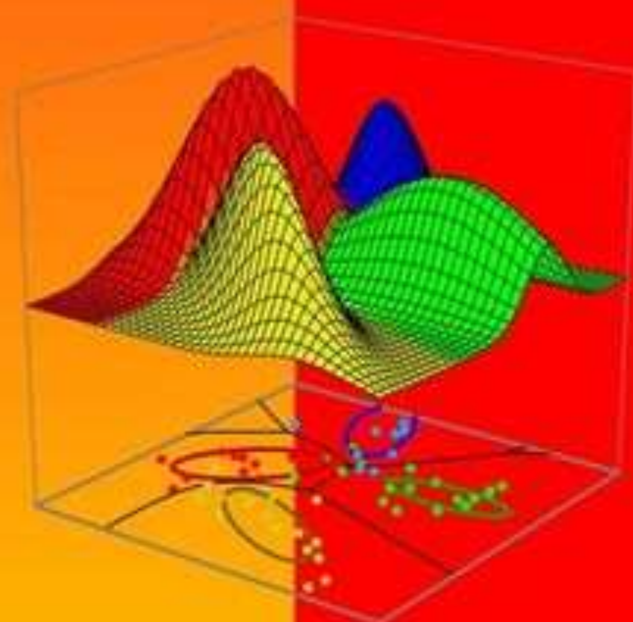
Data Mining, Inference, and Prediction

Second Edition

Springer

Richard O. Duda
Peter E. Hart
David G. Stork

Pattern Classification



Second Edition

O'REILLY

Hands-On Machine Learning with Scikit-Learn & TensorFlow

CONCEPTS, TOOLS, AND TECHNIQUES TO BUILD INTELLIGENT SYSTEMS

powered by jupyter

Aurélien Géron

Advances in Computer Vision and Pattern Recognition

ACVPR

Radu Tudor Ionescu
Marius Popescu

Knowledge Transfer between Computer Vision and Text Mining

Similarity-based Learning Approaches

Springer