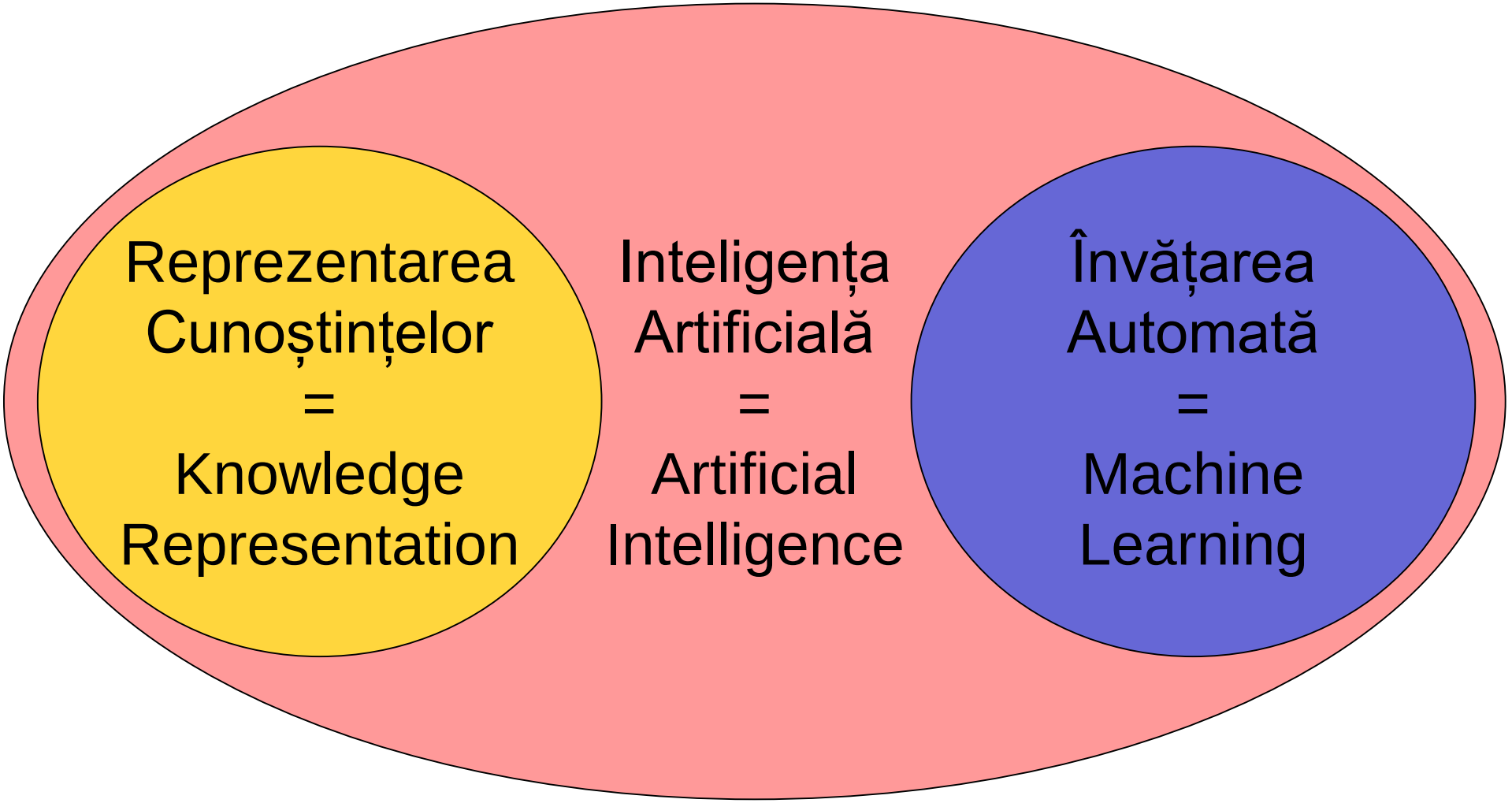
Inteligența artificială. Învățare automată. Concepte de bază.

Prof. Dr. Radu Ionescu
raducu.ionescu@gmail.com

Facultatea de Matematică și Informatică
Universitatea din București

Inteligența artificială și învățarea automată



Profesori - învățarea automată

- Cursuri:
  - Radu Ionescu (raducu.ionescu@gmail.com)

- Laboratoare și seminarii:
  - Alin Croitoru (alincroitoru97@gmail.com)
  - Adriana Costache (adriana16costache@gmail.com)
  - Dana Dăscălescu (danadascalescu00@gmail.com)
  - Diana Grigore (diana.grigore13@s.unibuc.ro)
  - Vlad Hondru (vlad.hondru25@gmail.com)
  - Eduard Poesina (eduard_poe2000@yahoo.com)
  - Mădălina Poșchină (madalinaposchina@gmail.com)

Sistem de notare

- Nota este formată din:

- nota din examen 50%

- nota din laborator 50%

- Vor fi două note la laborator, câte una pentru fiecare materie. Nota finală de la laborator este formată din media notelor de laborator
- Notele finale de la examen și laborator trebuie să fie ambele peste 5 (regula se aplică și la restanță, nu se reportează notele)
- La examen vor fi subiecte din ambele materii
- Pentru laboratorul de ”Învățare automată”:
  - nota se acordă pe baza unui test de laborator (în săptămâna a 14-a / sesiune)

Sistem de notare

- Puncte extra în timpul cursurilor / laboratoarelor

(se acordă doar la prima examinare)

- Curs:
  - Se acordă conform clasamentului din Kahoot
  - top 3 obțin 0.3 puncte pe curs, următorii 3 obțin 0.2 puncte, ș.a.m.d.
- Laborator:
  - Primul care răspunde la o întrebare / rezolvă un exercițiu primește 0.2 puncte
  - Maxim 0.4 puncte pe laborator de persoană (se punctează doar primele două răspunsuri corecte)
- Până la 1.5 puncte în plus la nota din examen
- Până la 2 puncte în plus la nota din laborator

Sistem de notare

- Puncte extra în timpul cursurilor / laboratoarelor

(se acordă doar la prima examinare)

- Seminar:
  - Test scris la final de semestru
  - Exerciții / probleme asemănătoare cu exemplele din clasă
  - Până la 3 puncte bonus la examen pentru testul de seminar
  - Până la 1 punct bonus pentru rezolvarea exercițiilor de seminar (0.2 per exercițiu, 0.4 maxim per student per seminar)

Sistem de notare

- Pentru nota de laborator, puteți primi până la 7 puncte bonus pentru realizarea unui proiect
- Proiectul constă în dezvoltarea unei metode de clasificare și participarea la competiția (TBA) propusă pe platforma Kaggle
- Notele vor fi proporționale cu rata de acuratețe obținută:

- Locurile 1-10 => 7 puncte bonus

- Locurile 11-20 => 6.5 puncte bonus

- Locurile 21-30 => 6 puncte bonus

…

- Locurile 101-110 => 1.5 puncte bonus

- Locurile 111-120 => 1 punct bonus

- Proiectul trebuie prezentat după ultimul laborator (se poate scădea până la 1 punct din bonificație pentru prezentare și documentație)

Sistem de notare

- Proiectele (cod+documentație) se trimit la adresa:

fmi.unibuc.ia@gmail.com

- Trimiteți doar fișiere .py! (nu se acceptă .ipynb)
- Reguli suplimentare vor fi comunicate pe pagina Kaggle a competiției
- Testul de laborator constă în implementarea și antrenarea unui anumit clasificator pe un anumit set de date, o parte semnificativă din nota obținută fiind proporțională cu acuratețea obținută de clasificatorul antrenat.
- Pentru atingerea punctului optim de învățare, punctajul va fi maxim. Vor exista câteva cerințe (cu punctaj separat) care să vă ajute la găsirea modelului și hiperparametrilor optimi.

Sistem de notare

- Codul de la proiecte va fi verificat cu soft-uri anti-plagiat
- NU este permisă preluare codului de pe web (sub nicio formă, nici măcăr din …)
- NU este permisă preluare codului de la colegi
- Bonificația poate fi retrasă pe baza similarităților identificate

Exemple de 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)))
```

Exemple de plagiarism

```
    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'))
```

Exemple de 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())
```

Exemple de 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)
```

Exemple de 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
```

Exemple de plagiarism

```
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)

    _, 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())
```

Exemple acceptabile

```
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
```

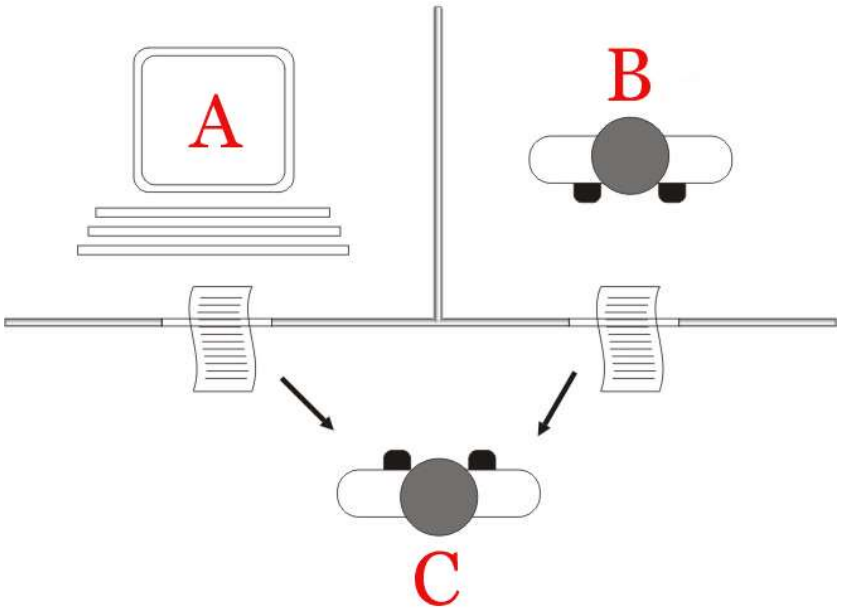
Exemple acceptabile

```
    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)
```

La ce se referă inteligența artificială?

- Scopul suprem al inteligenței artificiale este de a construi sisteme care să atingă nivelul de inteligență al omului
- Testul Turing: un computer prezintă un nivel de inteligență uman dacă un interlocutor uman nu reușește să distingă, în urma unei conversații în limbaj natural, că vorbește cu un om sau cu un calculator



La ce se referă învățarea automată?

- O mare parte din cercetători consideră că acest scop poate fi atins prin imitarea modului în care o oamenii învață
- **Învățarea automată** – domeniu care studiază modul în care calculatoarele pot fi înzestrate cu abilitatea de a învăța, fără ca aceasta să fie programată în mod explicit
- În acest context, **învățarea** se referă la:
  - recunoașterea unor tipare / structuri (patterns) complexe
  - luarea deciziilor inteligente bazate pe observațiile din **date**

Problemă “bine pusă” de învățare automată

- Ce probleme pot fi rezolvate* folosind învățarea automată?
- **Problemă “bine pusă” de învățare automată:**
- Spunem despre un program pe calculator că învață dintr-o experiență E în raport cu o clasă de task-uri T și o măsură de performanță P, dacă performanța sa în rezolvarea task-urilor T, măsurată prin P, se îmbunătățește odată cu experiența E

- **(*) rezolvate cu un anumit grad de acuratețe**

Problemă “bine pusă” de învățare automată

- Arthur Samuel (1959) a scris un program pentru a juca dame (probabil primul program bazat pe conceptul de învățare)
- Programul a jucat împotriva lui însuși 10 mii de jocuri
- Programul a fost conceput să găsească ce poziții ale tablei de joc erau bune sau rele în funcție de probabilitatea de a câștiga sau pierde
- În acest caz:
  - E = 10000 de jocuri
  - T = joacă dame
  - P = dacă câștigă sau nu

Strong AI versus Weak AI

- strong / generic / true AI

(vezi definiția lui Turing)

- weak / narrow AI

(se concentrează pe o anumită problemă)

Când se aplică învățarea automată?

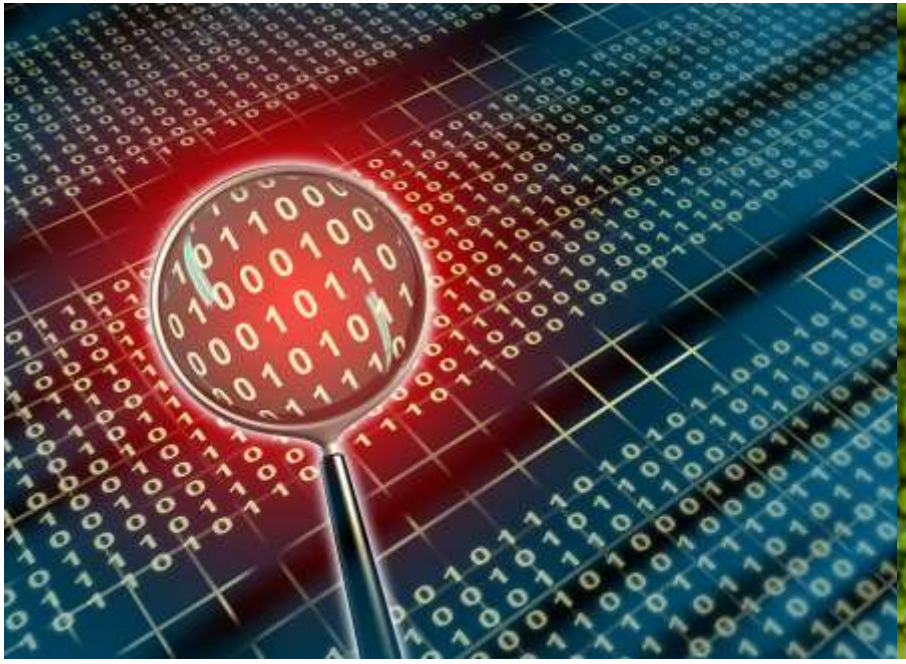
- Se aplică în situații în care este foarte greu (imposibil) să definim un set de reguli de mână / să scriem un program

- Exemple de probleme unde putem aplica învățarea automată:

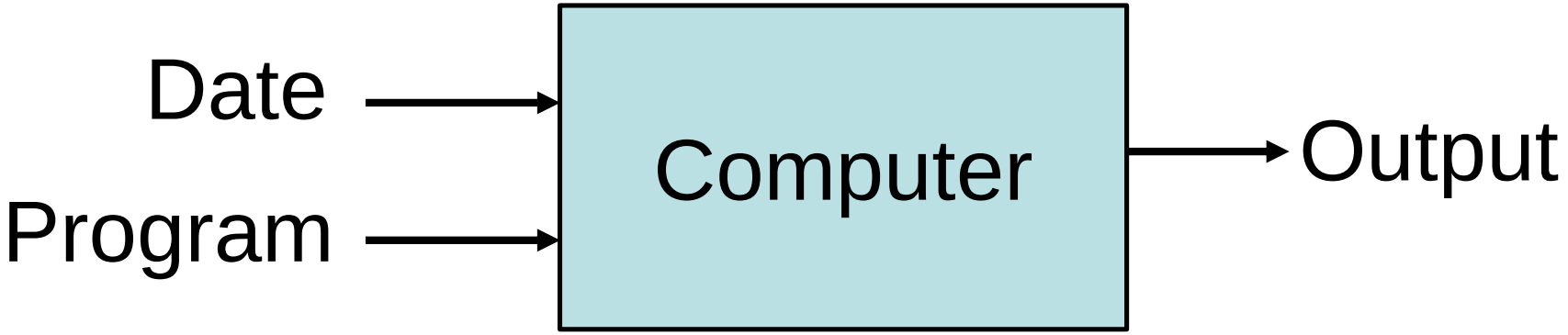
  - Detectarea facială
  - Înțelegerea vorbirii
  - Prezicerea prețului acțiunilor
  - Recunoașterea obiectelor

Esența învățării automate

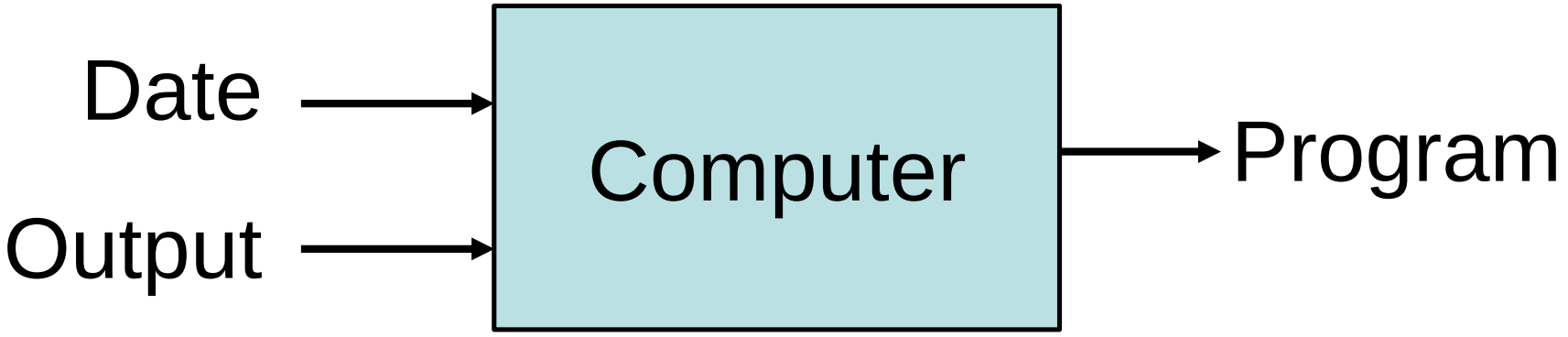
- Există un tipar
- Dar nu îl putem exprima programatic / matematic
- Avem date / exemple în care regăsim acest tipar



Programare tradițională



Învățare automată



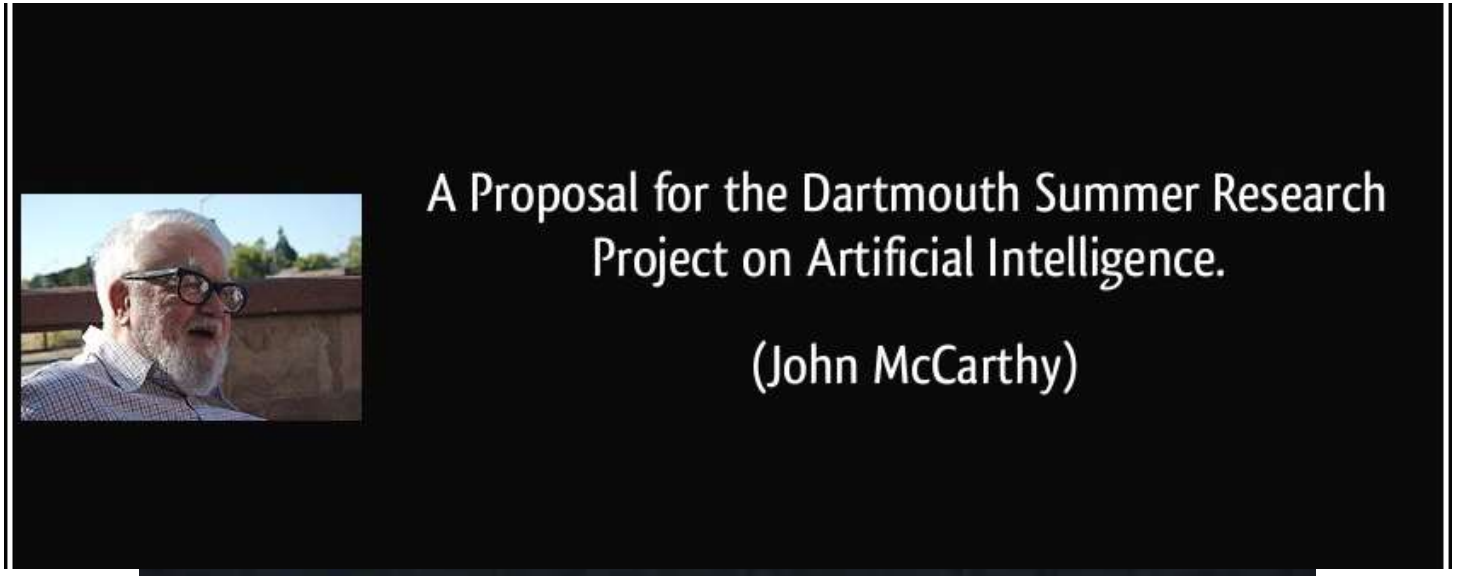
Ce este învățarea automată?

[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)

Scurt istoric al inteligenței artificiale



Scurt istoric al inteligenței artificiale

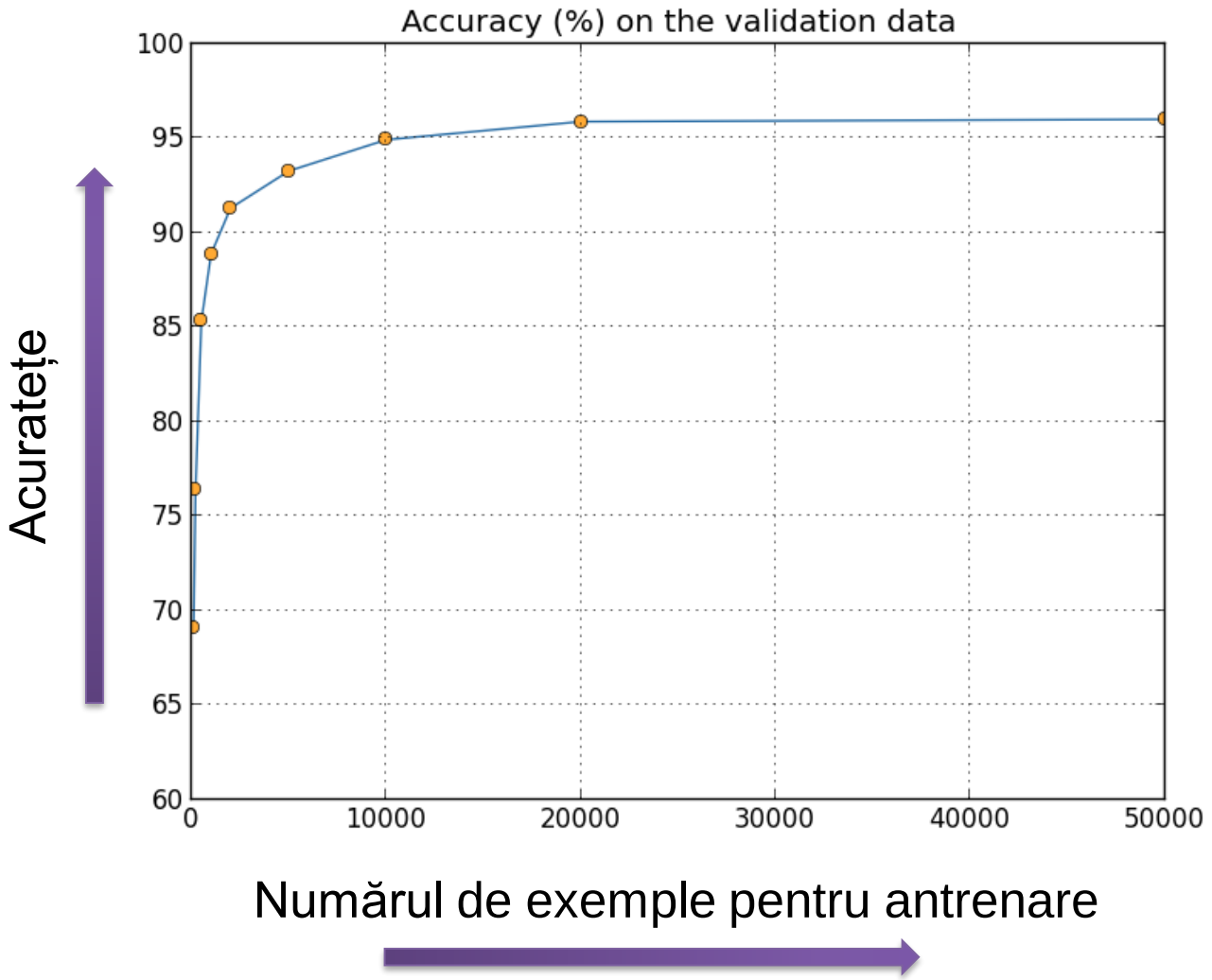
- “We propose that a 2 month, 10 man study of artificial intelligence be carried out during the summer of 1956 at Dartmouth College in Hanover, New Hampshire.”
- The study is to proceed on the basis of the conjecture that every aspect of learning or any other feature of intelligence can in principle 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.”

Scurt istoric al inteligenței artificiale

- Anii 1960-1980: ”AI Winter”
- Anii 1990: Rețelele neuronale domină, în principal datorită descoperirii algoritmului de propagare a erorii înapoi pentru rețele cu mai multe straturi
- Anii 2000: Metodele kernel domină, în principal din cauza instabilității rețelelor neuronale
- Anii 2010: Revenirea la rețele neuronale, în principal datorită conceptului de învățare profundă (deep learning)

De ce funcționează în prezent?

- Mai multă putere de calcul
- Mai multe date
- Modele mai bune



Esența învățării automate

- Mii de algoritmi de învățare automata existenți
- ➢ Cercetătorii publică sute de noi algoritmi în fiecare an

- Simplificând decenii de cercetare în domeniu, putem reduce învățarea automată la:
- ➢ Învățarea unei funcții f care să mapeze un input X către un output Y, anume $f: X \rightarrow Y$
- ➢ Exemplu: X: email-uri, Y: {spam, non-spam}

Esența învățării automate

- Input: X (imagini, texte, email-uri…)
- Output: Y (spam sau non-spam…)
- Funcție Target (necunoscută)

$f: X \to Y$ (realitatea / ”adevărata” mapare)

- Date

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

- Model

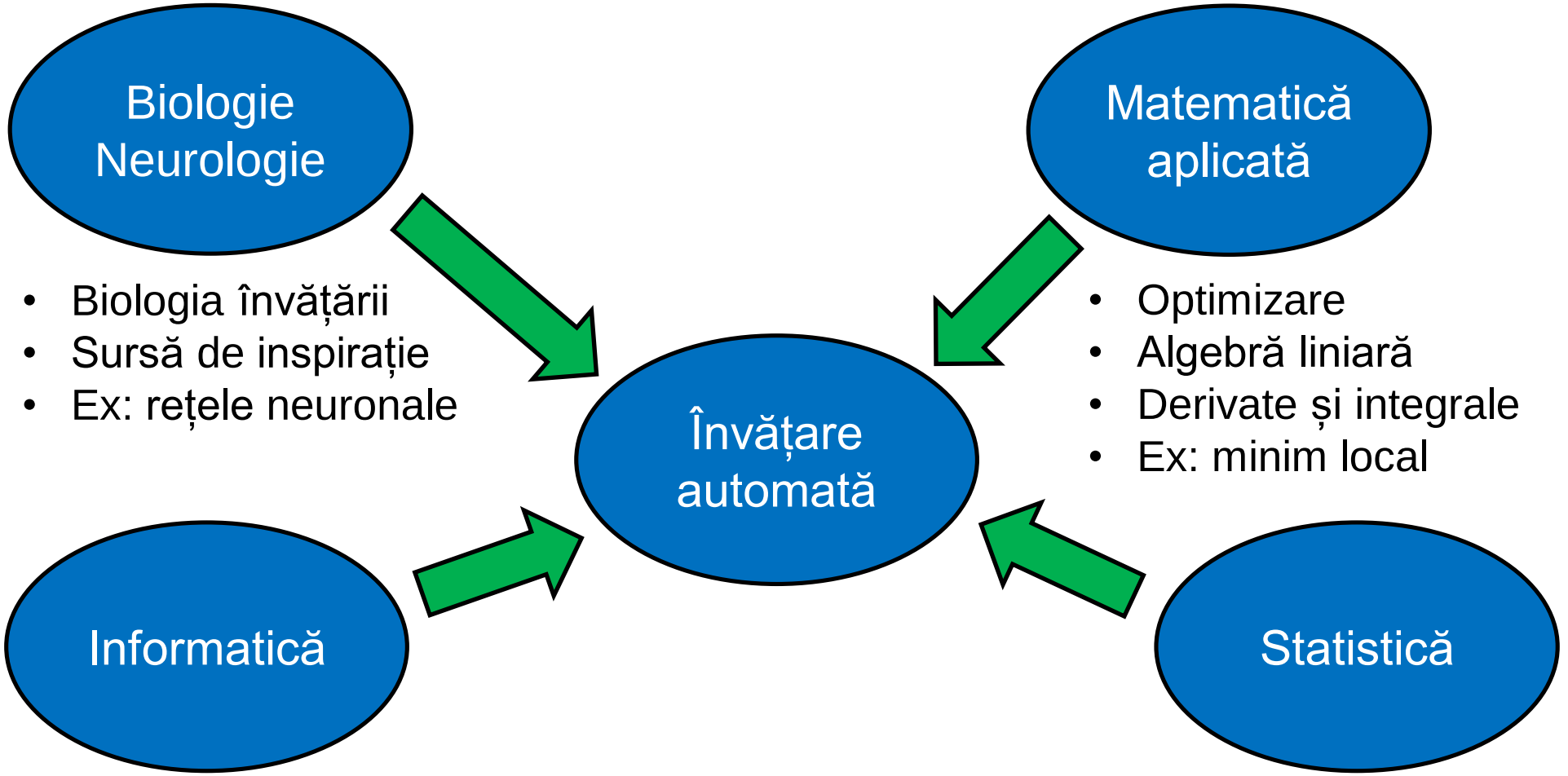
$g: X \to Y$

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

Esența învățării automate

- Orice algoritm de învățare automată are 3 componente:
  - Reprezentare / Modelare
  - Evaluare / Funcție obiectiv
  - Optimizare

Ce cunoștințe sunt necesare?



- Biologia învățării
- Sursă de inspirație
- Ex: rețele neuronale

- Optimizare
- Algebră liniară
- Derivate și integrale
- Ex: minim local

- Algoritmi
- Structuri de date
- Analiza complexității
- Ex: arbori k-d

- Tehnici de estimare
- Paradigme teoretice
- Optimalitate, eficiență
- Ex: regula Bayes

Paradigme ale învățării

- Învățare supervizată (supervised learning)
- Învățare nesupervizată (unsupervised learning)
- Învățare semi-supervizată (semi-supervised learning)
- Învățare ranforsată (reinforcement learning)

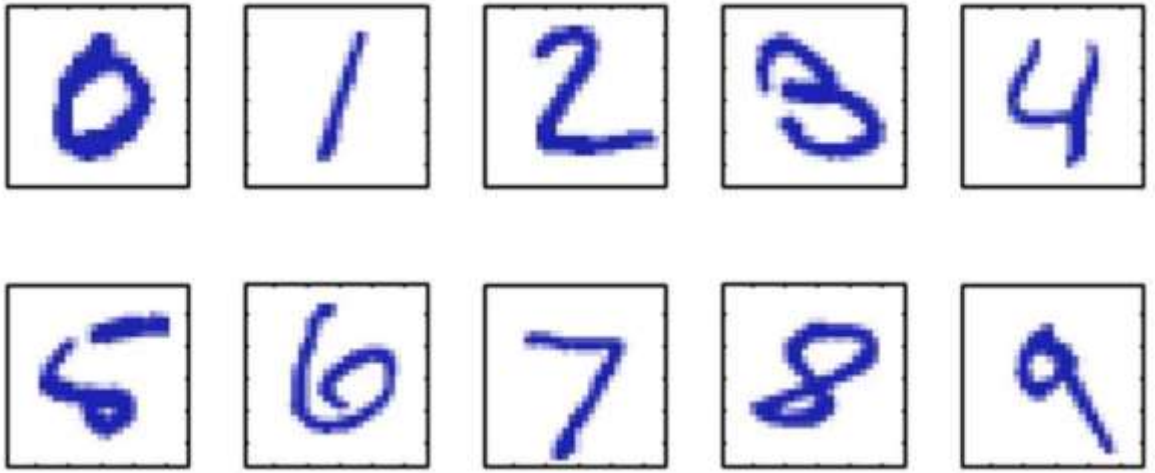
- Paradigme non-standard:
- Învățarea activă (active learning)
- Învățare prin transfer (transfer learning)

Învățare supervizată

- Avem la dispoziție exemple de obiecte etichetate
- Exemplu 1: recunoașterea obiectelor din imagini cu eticheta obiectelor conținute

Învățare supervizată

- Exemplu 2: recunoașterea caracterelor scrise de mână (setul de date MNIST)



- Imagini de 28 x 28 de pixeli
- Reprezentăm o imagine ca un vector x cu 784 de componente
- Antrenăm un clasificator f(x) astfel încât:
- $f : x \rightarrow \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

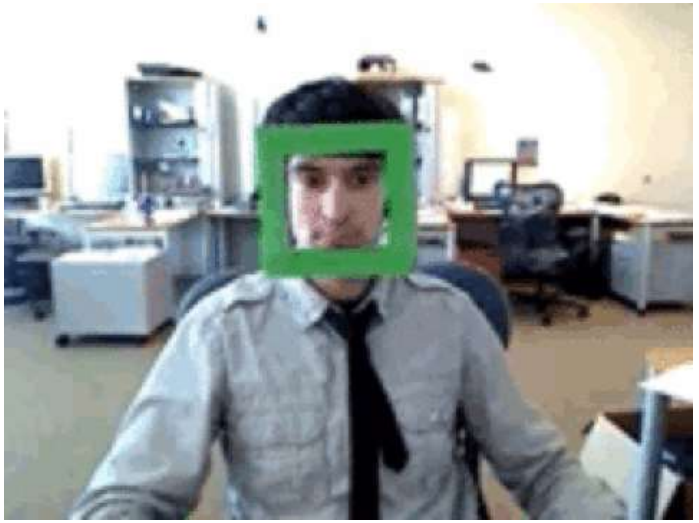
Învățare supervizată

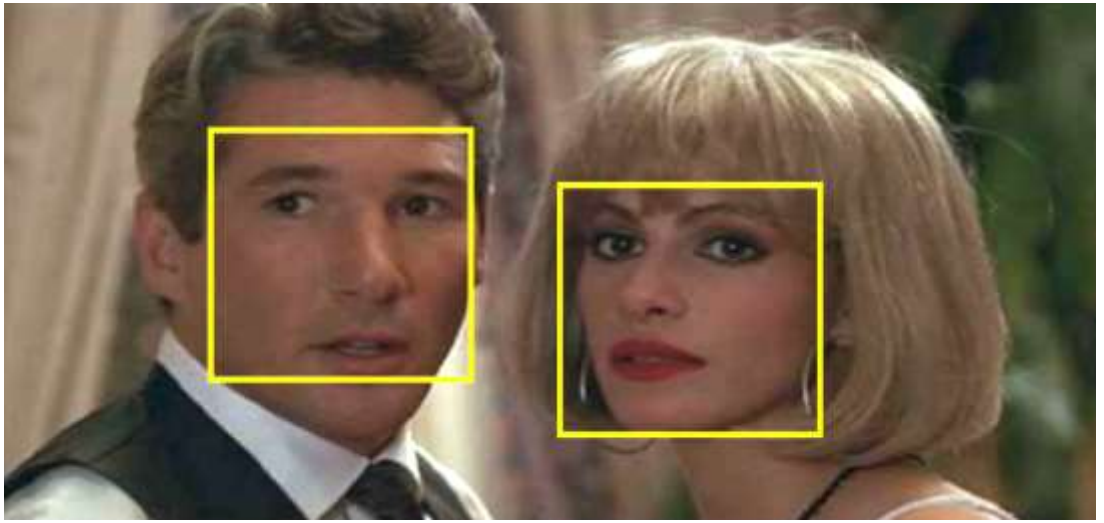
- Exemplu 2: recunoașterea caracterelor scrise de mână (setul de date MNIST)

- Pornind de la un set de antrenare, de exemplu 6000 de imagini per clasă
- Rata de eroare poate ajunge la 0.23% (cu rețele neuronale convoluționale)
- Printre primele sisteme (bazate pe învățare) comerciale utilizate pe scară largă pentru procesare de coduri poștale și cecuri bancare

Învățare supervizată

- Exemplu 3: detectare facială





- O abordare constă în plimbarea unei ferestre peste imagine
- Scopul este să clasificăm fereastra într-una din cele două clase posibile: față sau non-față (transformarea problemei într-una de clasificare)

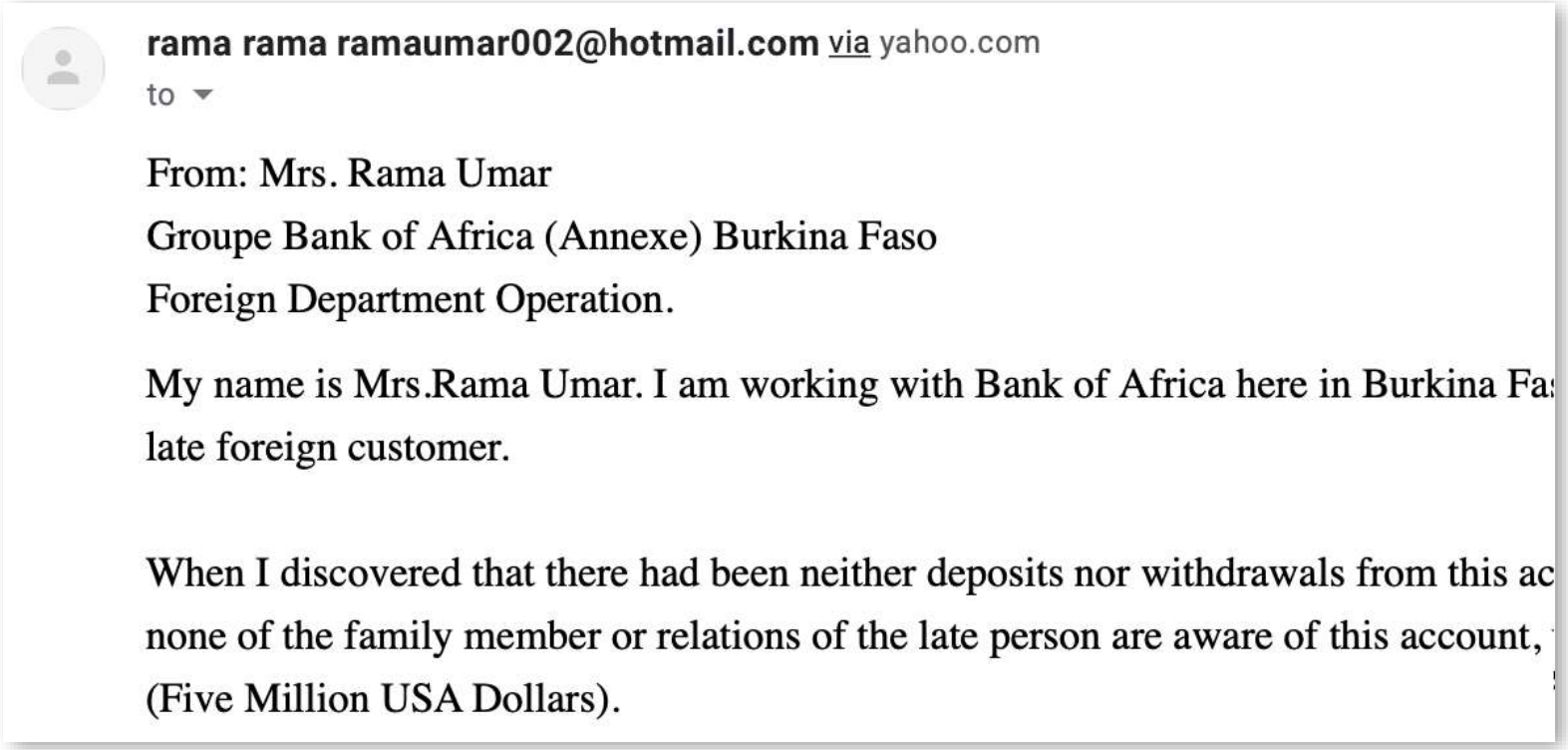
Învățare supervizată

- Exemplu 3: detectare facială

- Pornim de la un set cu imagini cu fețe cu diverse variații de vârstă, gen, condiții de iluminare, dar nu translație.
- Și un set mult mai mare cu imagini care nu conțin fețe

Învățare supervizată

- Exemplu 4: detectare de spam



rama rama ramaumar002@hotmail.com via yahoo.com

to

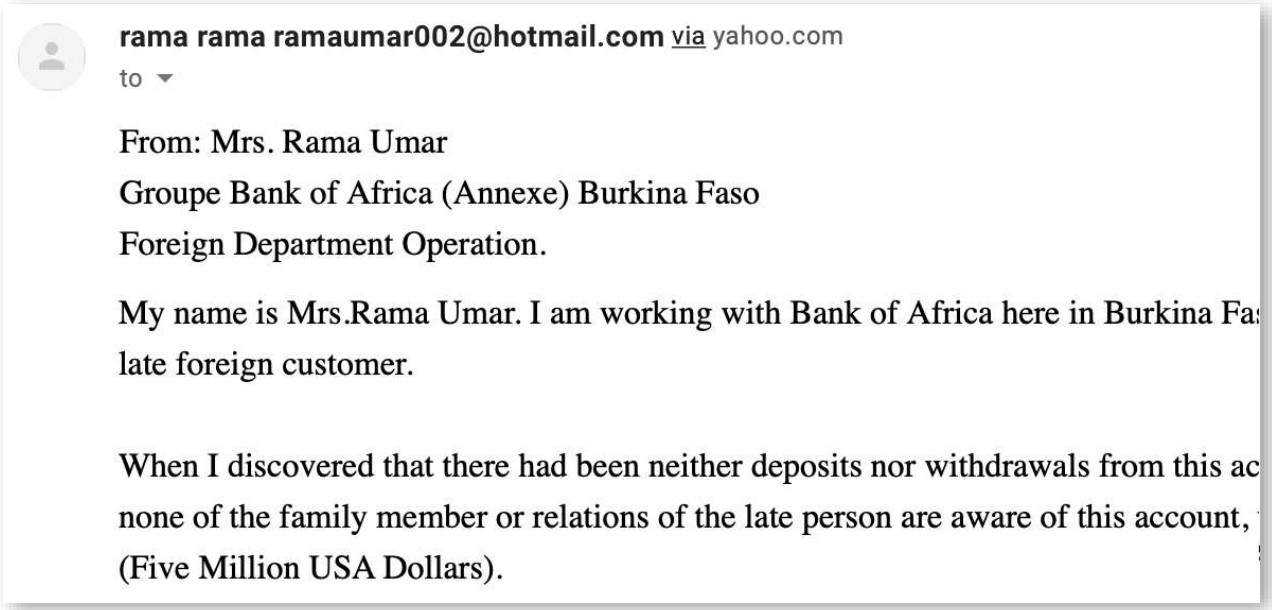
From: Mrs. Rama Umar
Groupe Bank of Africa (Annexe) Burkina Faso
Foreign Department Operation.

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).

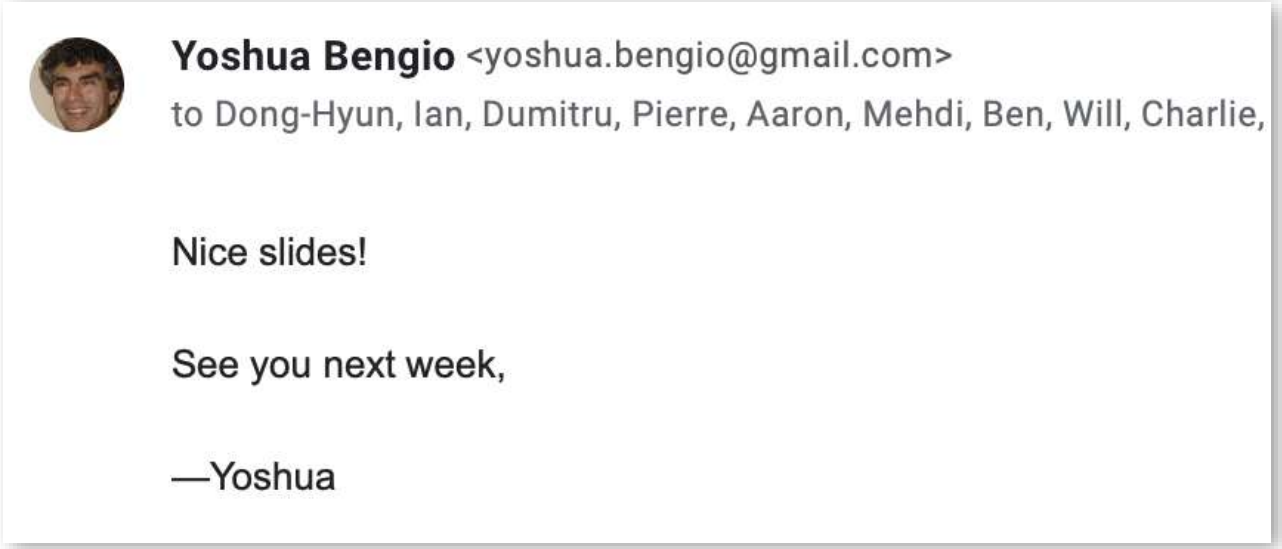
- Problema este de a clasifica un e-mail în spam și non-spam
- Apariția cuvântului “Dollars” este un indicator de spam
- Un exemplu de reprezentare este un vector cu frecvența cuvintelor

Numărăm cuvintele



Obținem X

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



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

Algoritm de detectare a spam-ului

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

De ce aceste cuvinte?

$$\begin{pmatrix} 100 \times 0.2 \\ 2 \times 0.3 \\ \vdots \\ 2 \times 0.3 \\ \vdots \end{pmatrix} \qquad \begin{pmatrix} 100 \times 0.01 \\ 2 \times 0.02 \\ \vdots \\ 2 \times 0.01 \\ \vdots \end{pmatrix}$$

De ce combinație liniară?

De unde vin aceste ponderi?

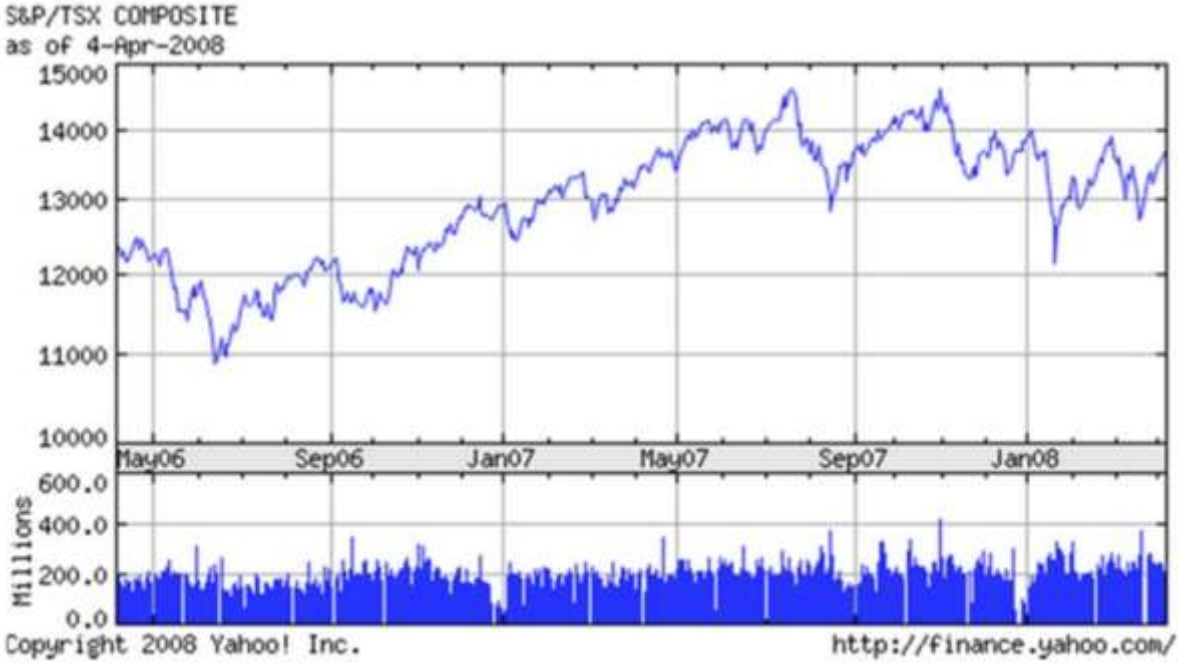
$= 3.2$

$= 1.03$

Confidență / garanția performanței?

Învățare supervizată

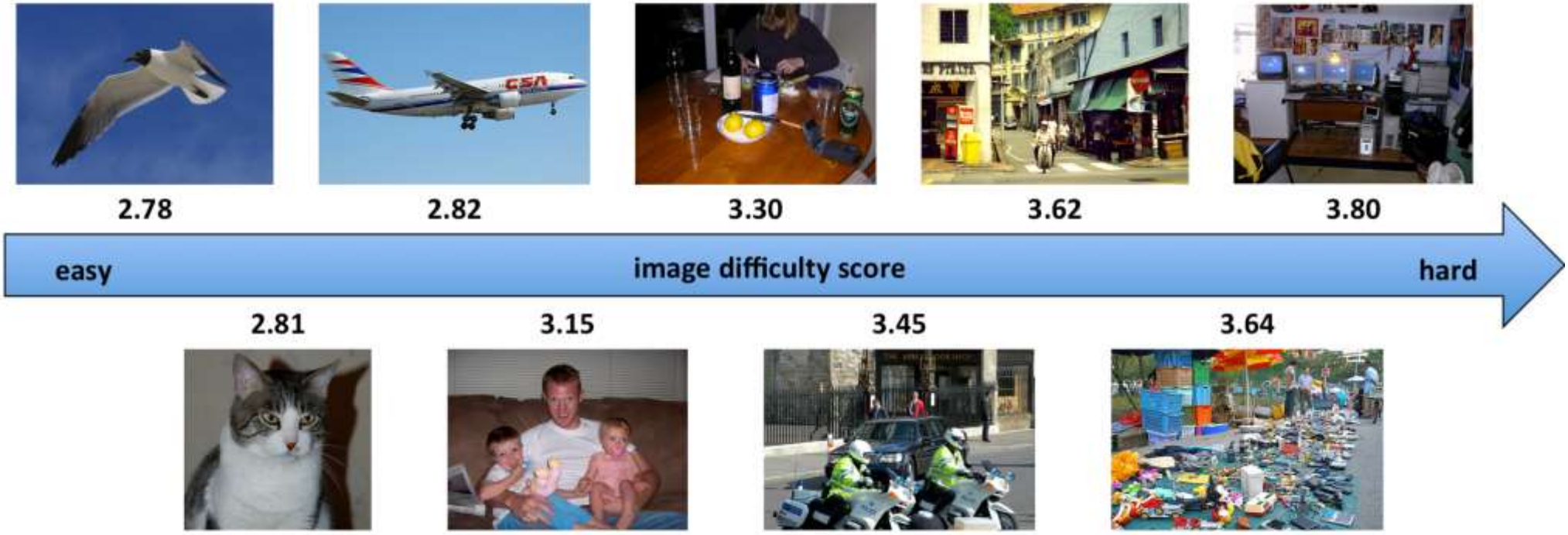
- Exemplu 5: prezicerea prețului acțiunilor la bursă



- Scopul este de a prezice prețul la o dată din viitor, de exemplu peste câteva zile
- Acesta este un task de regresie, deorece output-ul este unul continuu

Învățare supervizată

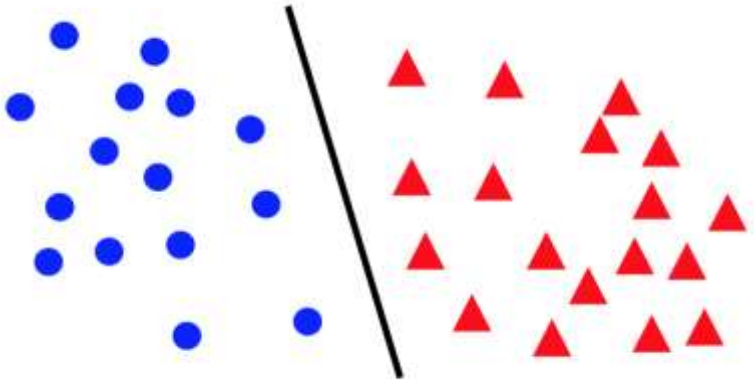
- Exemplu 6: prezicerea dificultății unei imagini



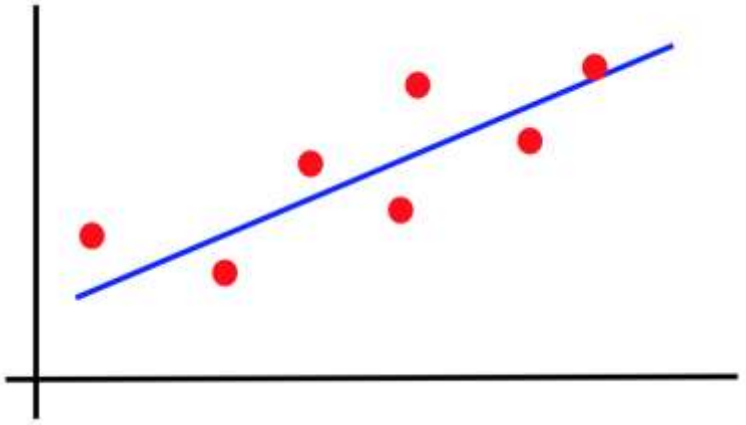
- Scopul este de a prezice cât de dificil ar fi pentru un om să recunoască obiectele din imagine
- Acesta este un task de regresie, deorece output-ul este unul continuu

Formele canonice ale problemelor de învățare supervizată

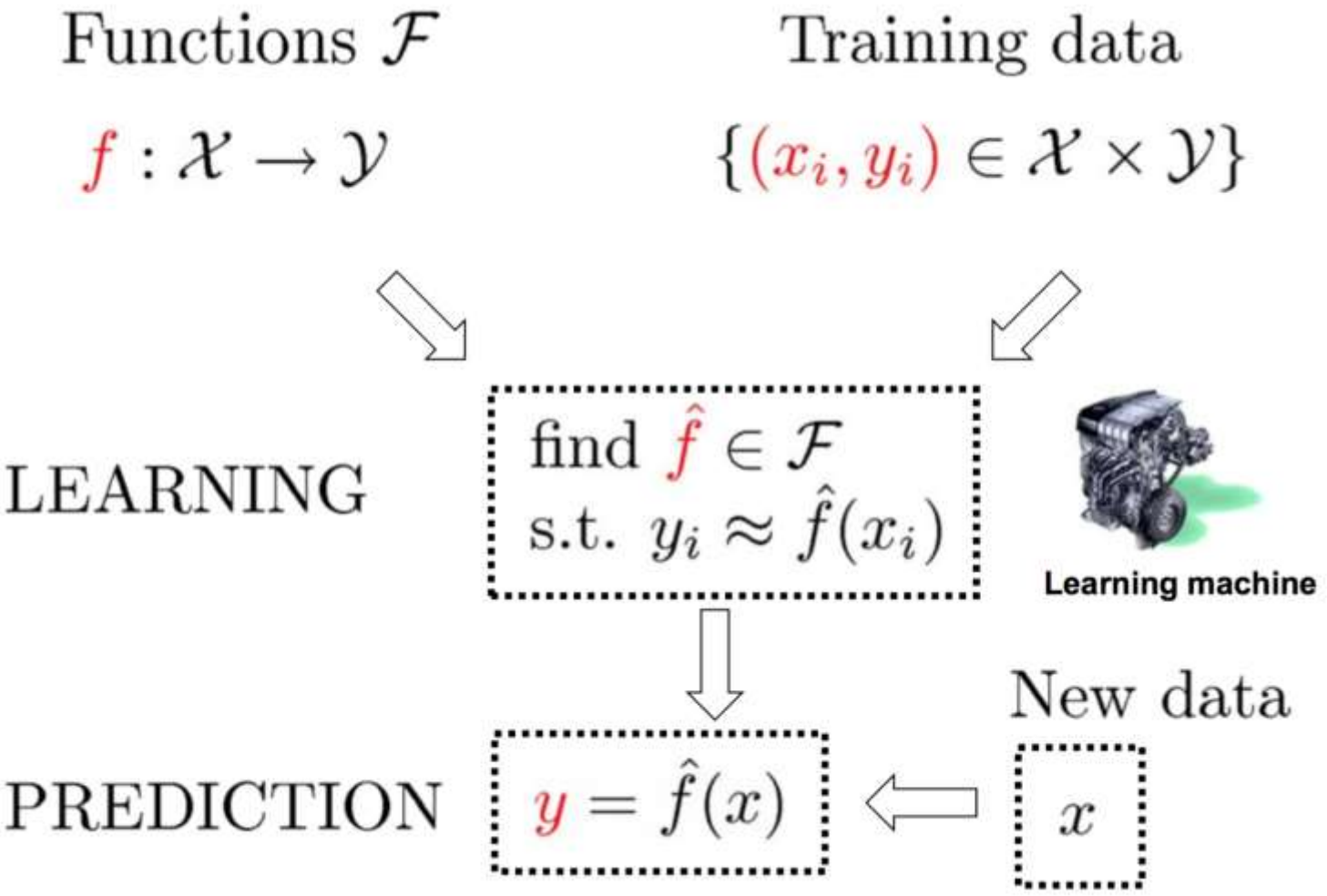
- Clasificare



- Regresie



Paradigma de învățare supervizată



Modele de învățare supervizată

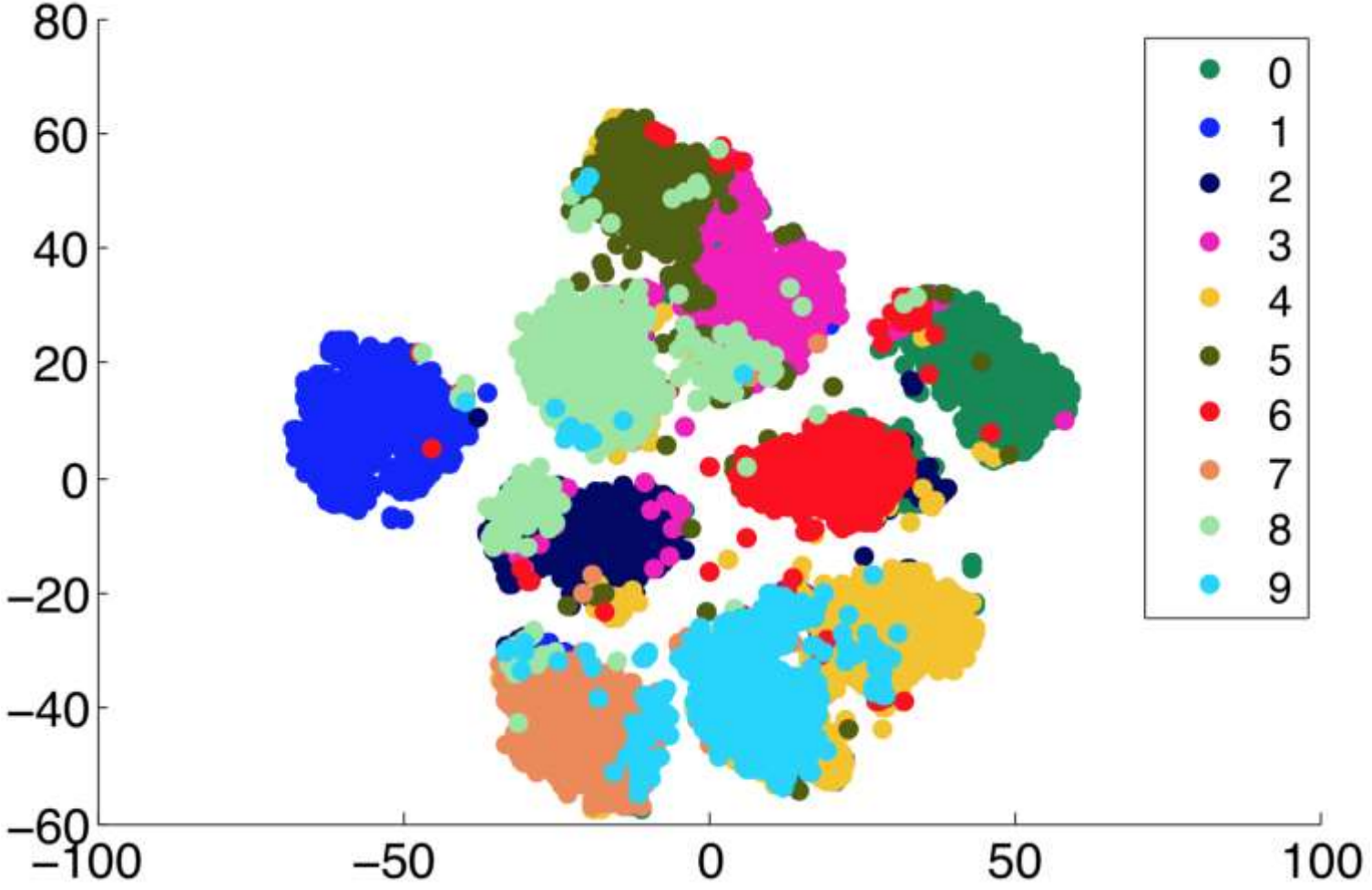
- Clasificatorul Bayes naiv (cursul 2)
- Metoda celor mai apropiați vecini (cursul 3)
- Clasificatorul cu vectori suport (cursul 4)
- Metode kernel (cursul 4)
- Rețele neuronale și învățare “deep” (cursurile 5, 6, 7)
- Arbori de decizie și random forests (la master)
- Altele

Învățare nesupervizată

- Avem la dispoziție exemple de obiecte fără etichete
- Exemplu 1: gruparea imaginilor după similaritate

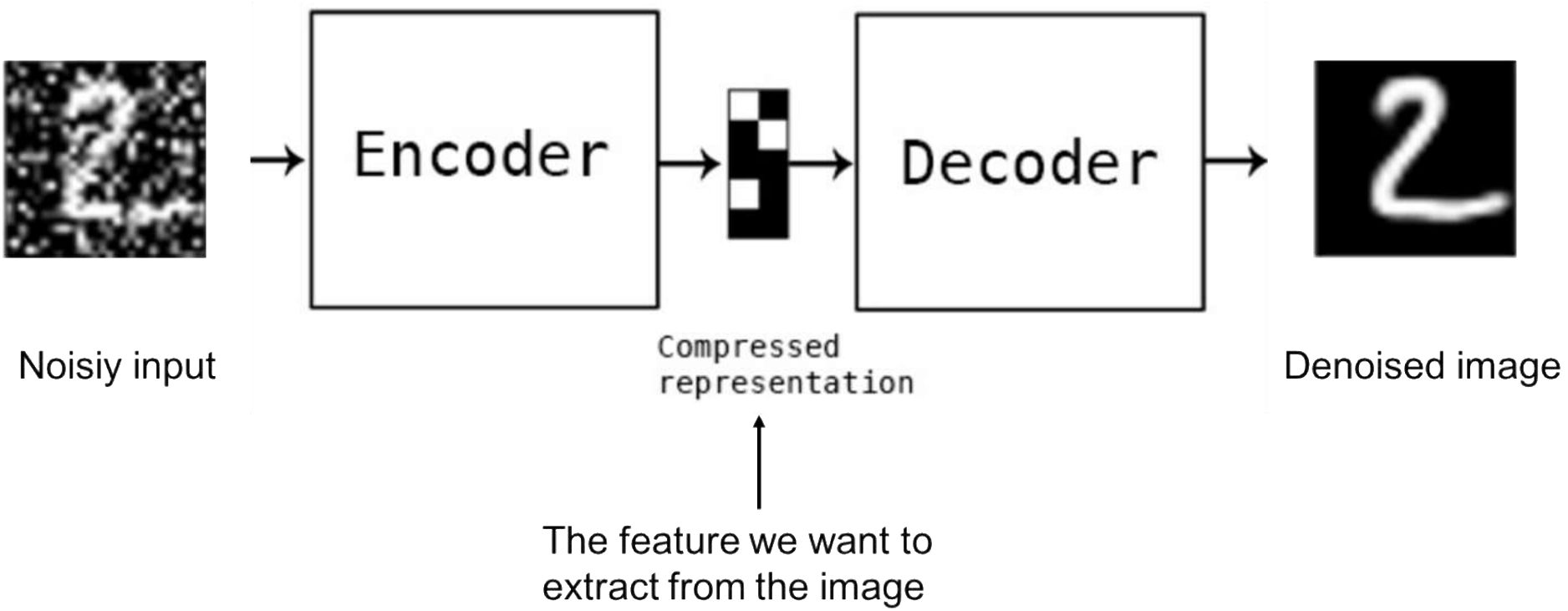
Învățare nesupervizată

- Exemplu 2: clusterizarea aglomerativă a imaginilor MNIST [Georgescu et al. ICIP2019]



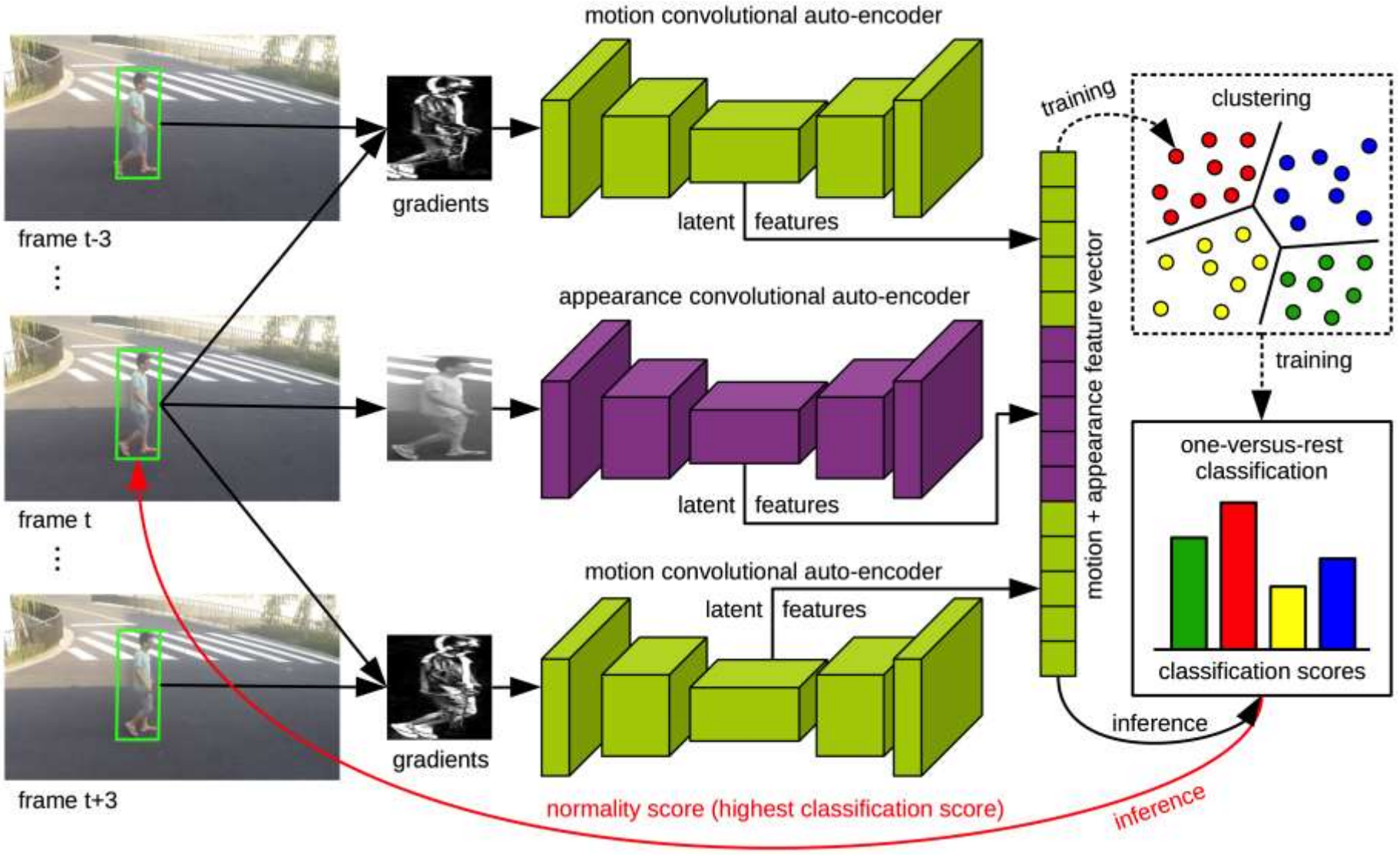
Învățare nesupervizată

- Exemplu 3: învățarea de trăsături folosind principiul “bottleneck”



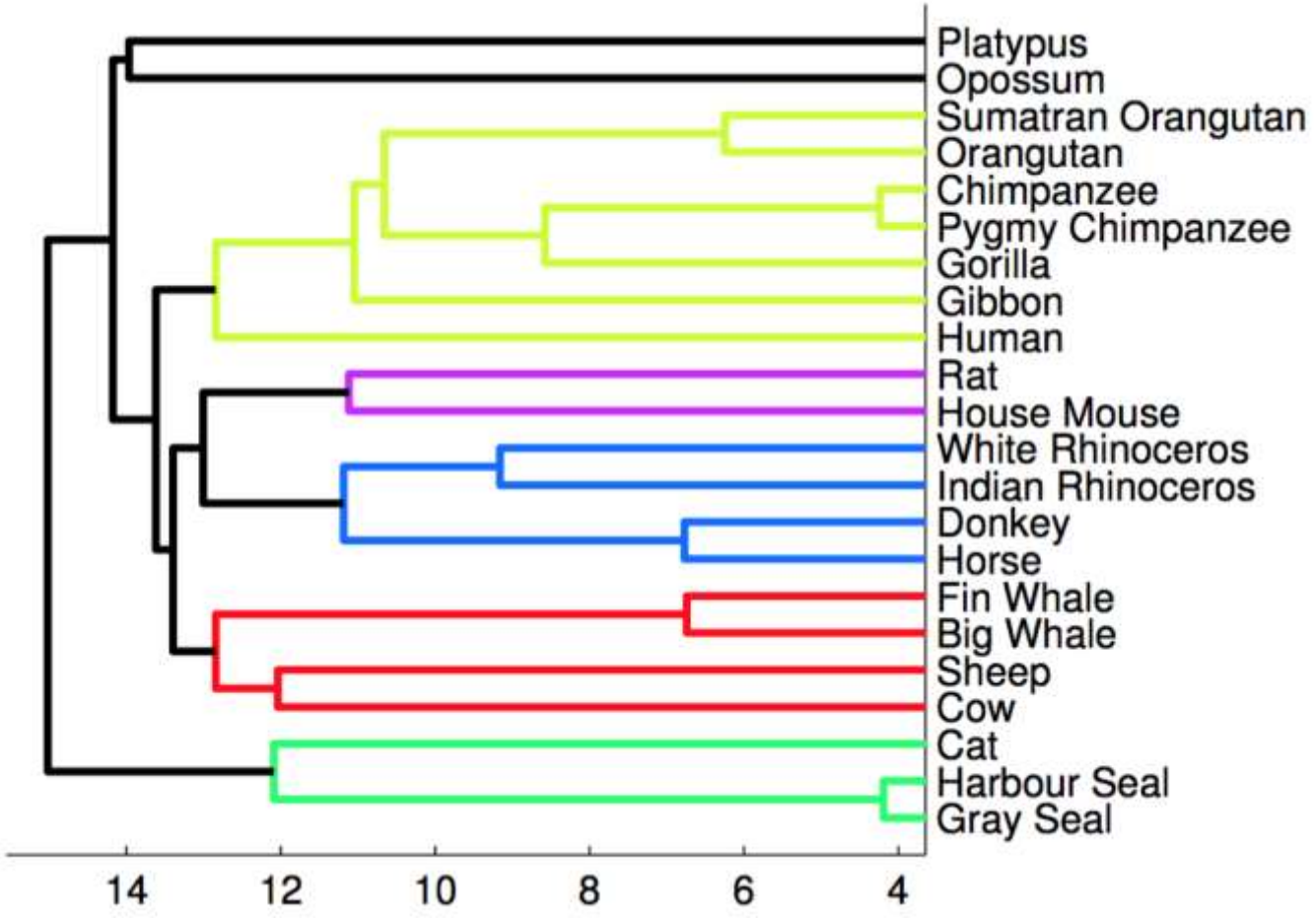
Învățare nesupervizată

- Exemplu 3: învățarea de trăsături pentru detectarea evenimentelor anormale [Ionescu et al. CVPR2019]



Învățare nesupervizată

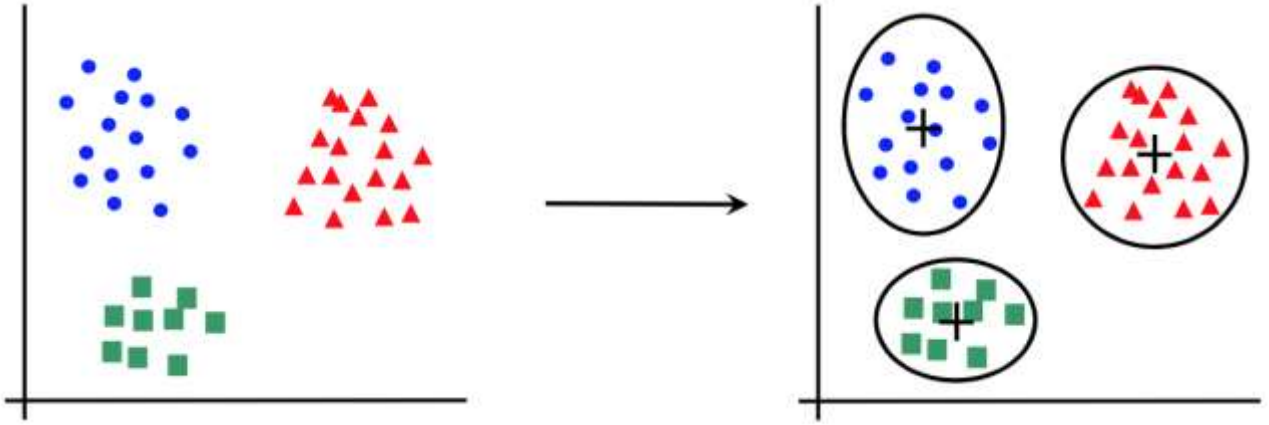
- Exemplu 2: gruparea mamiferelor pe familii, specii, etc.



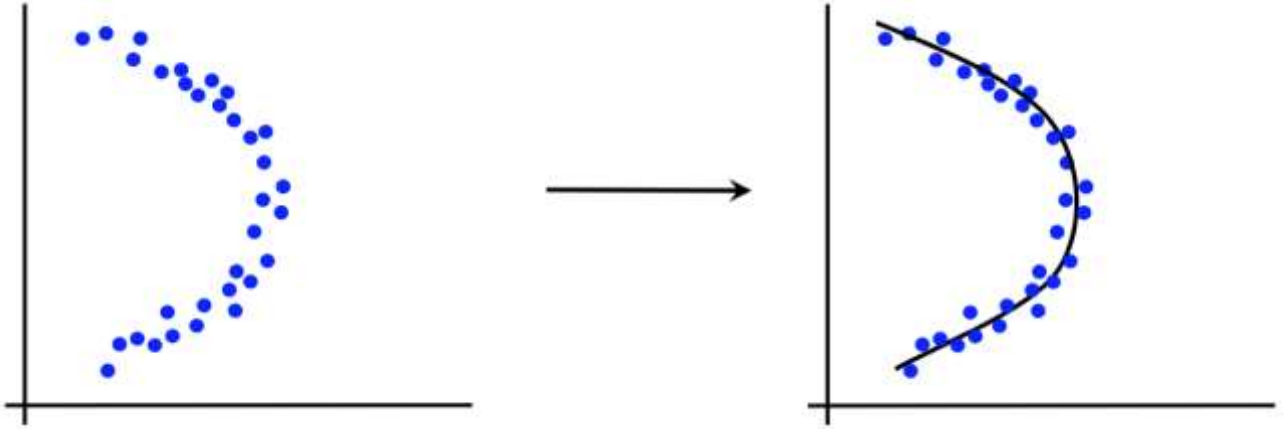
- Generarea arborelui filogenetic pe baza secvențelor ADN

Formele canonice ale problemelor de învățare nesupervizată

- Grupare (clustering)



- Reducerea dimensiunii



Modele de învățare nesupervizată

- K-means clustering (la master)
- Clustering ierarhic (la master)
- Analiza în componente principale (la master)
- Modele de tip auto-encoder (la master)
- Altele

Învățare semi-supervizată

- Avem la dispoziție exemple de obiecte etichetate și exemple de obiecte netichetate
- Exemplu 1: recunoașterea obiectelor din imagini, unele cu eticheta obiectelor conținute

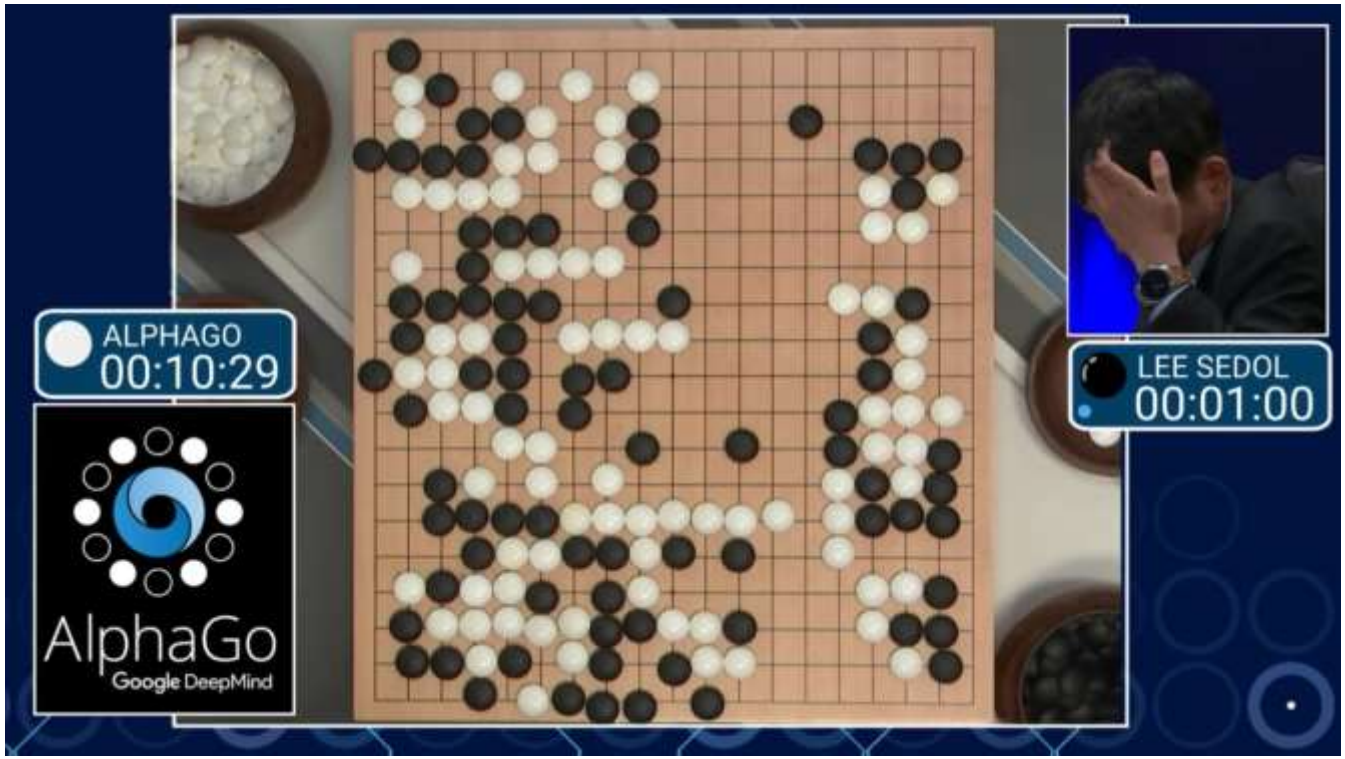
Învățare ranforsată

- Cu ce diferă această paradigmă de învățare?

- Sistemul învață comportamentul inteligent pe baza unei recompense (reinforcement signal)
- Recompensa este primită după mai multe acțiuni (nu vine instant)
- Timpul contează (datele sunt secvențiale, nu i.i.d.)
- Acțiunea sistemului influențează datele

Învățare ranforsată

- Exemplu 1: învățarea jocului Go
- recompensă +/- pentru câștigarea/pierderea unui joc



Învățare ranforsată

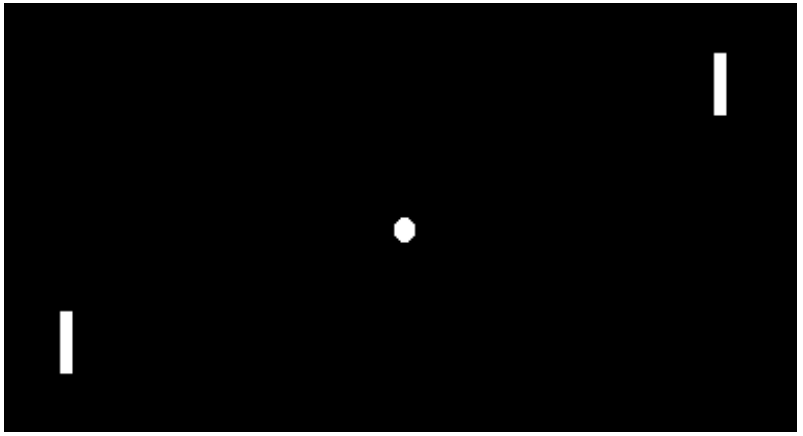
- Exemplu 2: învățarea unui robot să meargă pe bicicletă
- recompensă +/- pentru mișcare înainte/cădere

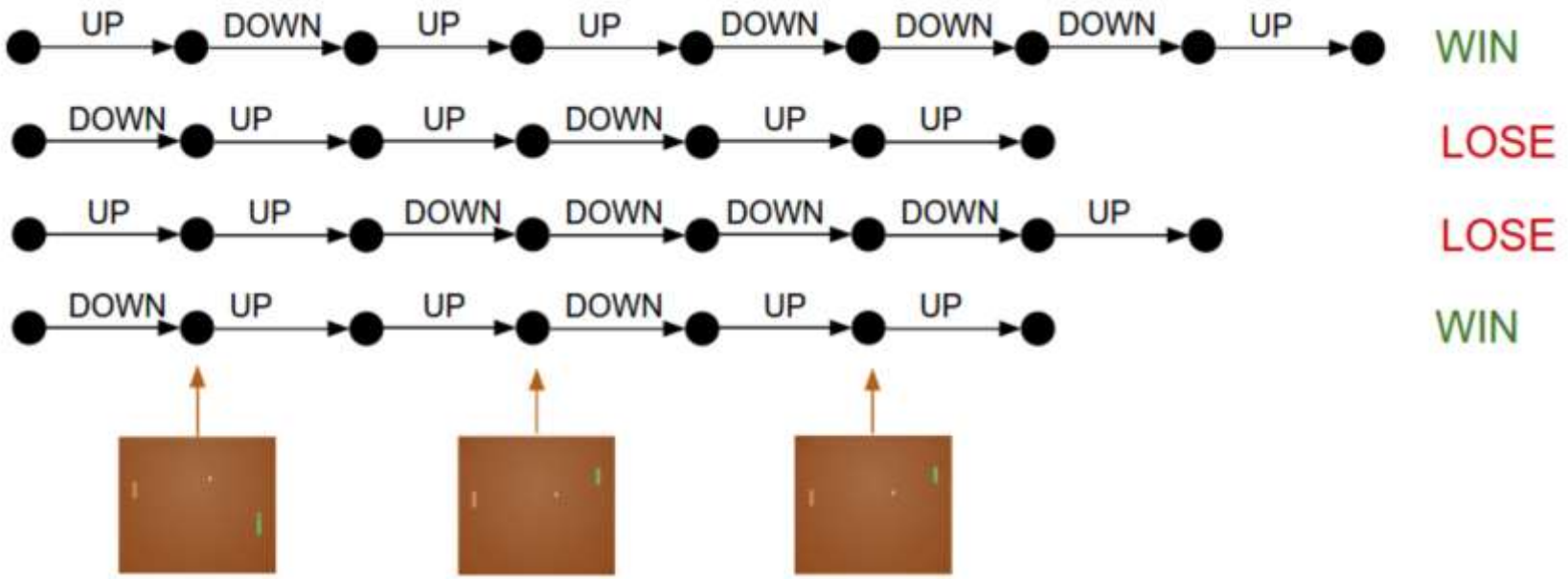
Învățare ranforsată

- Exemplu 3: învățarea jocului Pong din pixeli
- recompensă +/- pentru

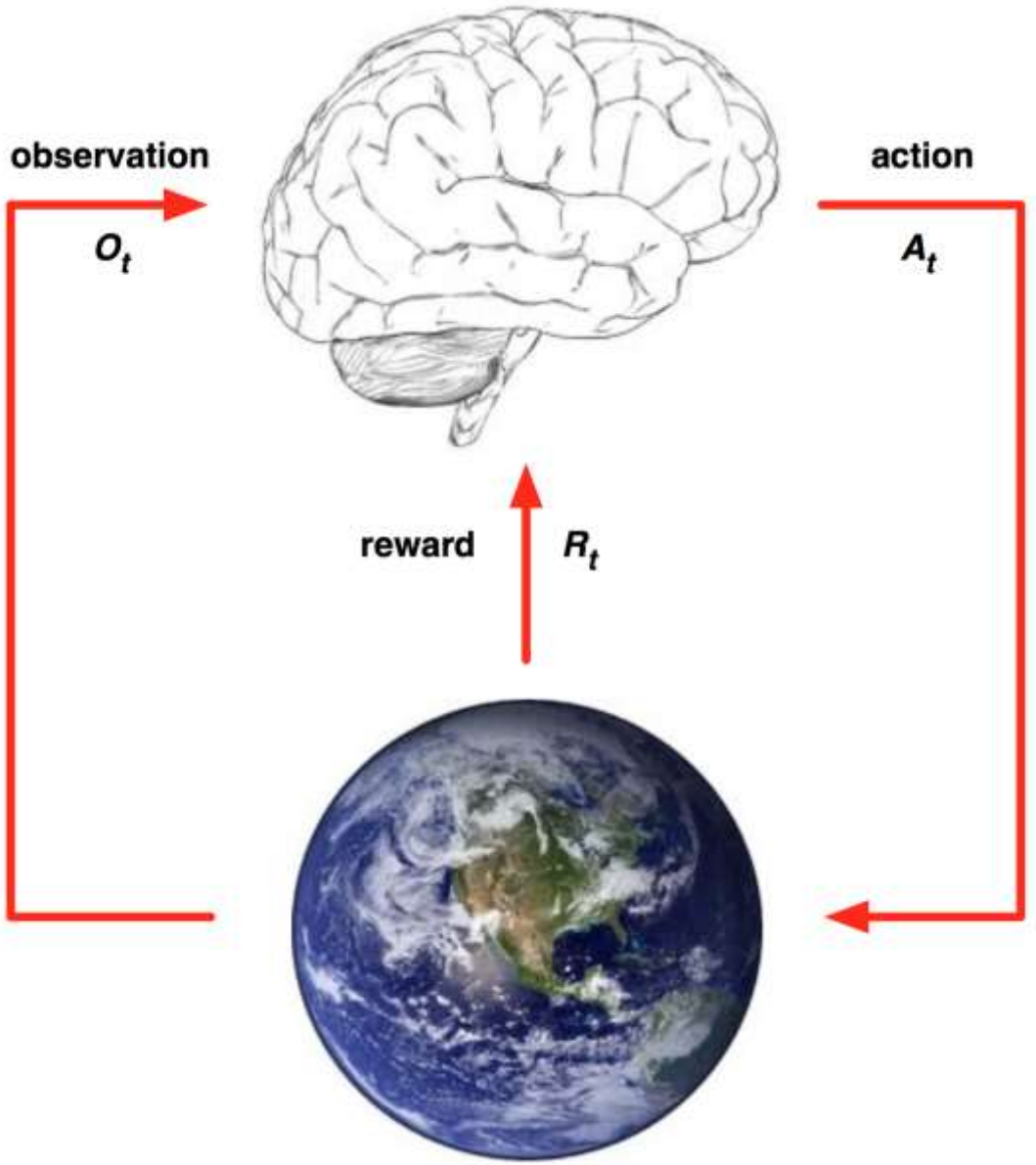
creșterea scorului

personal/al adversarului

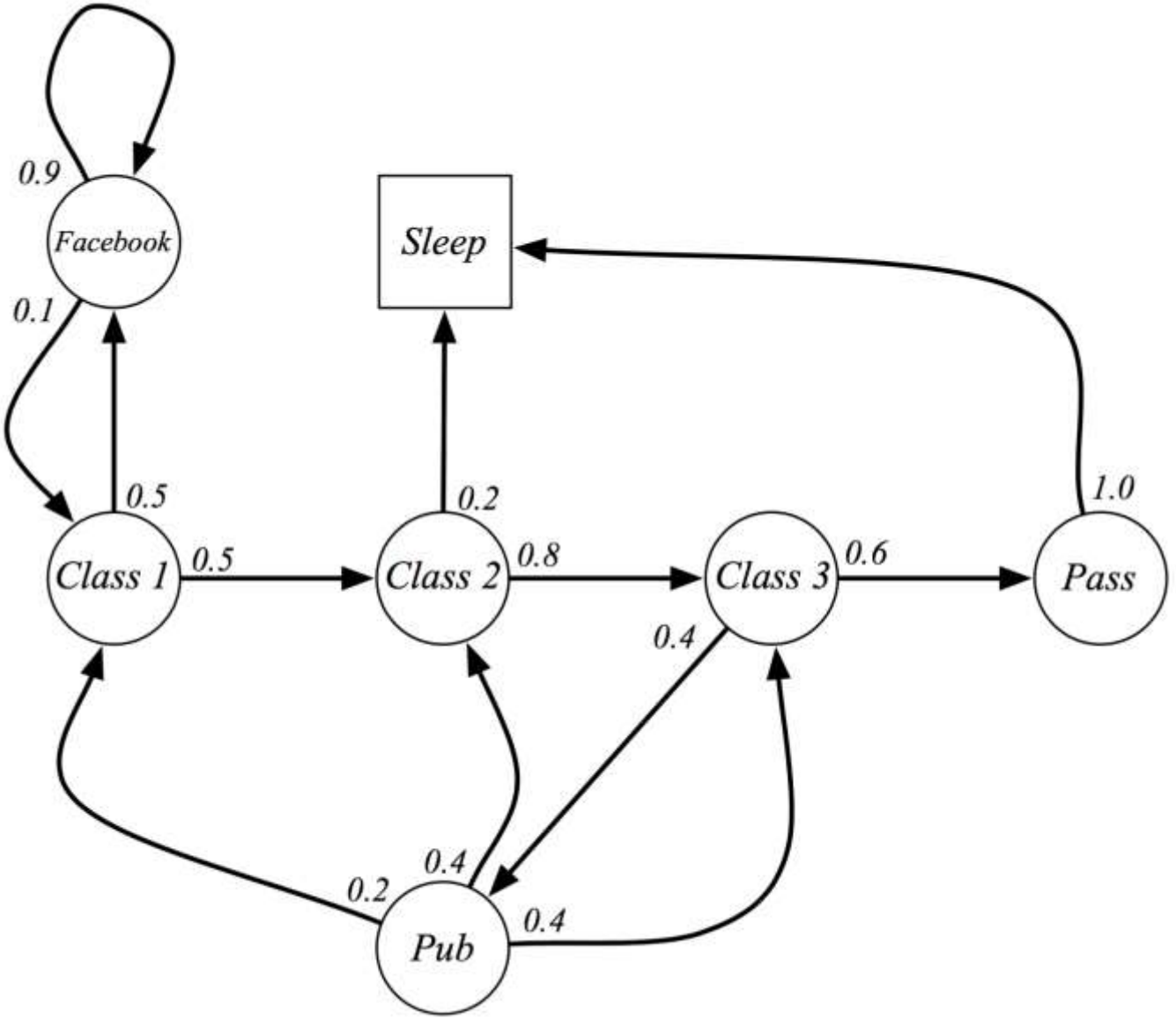




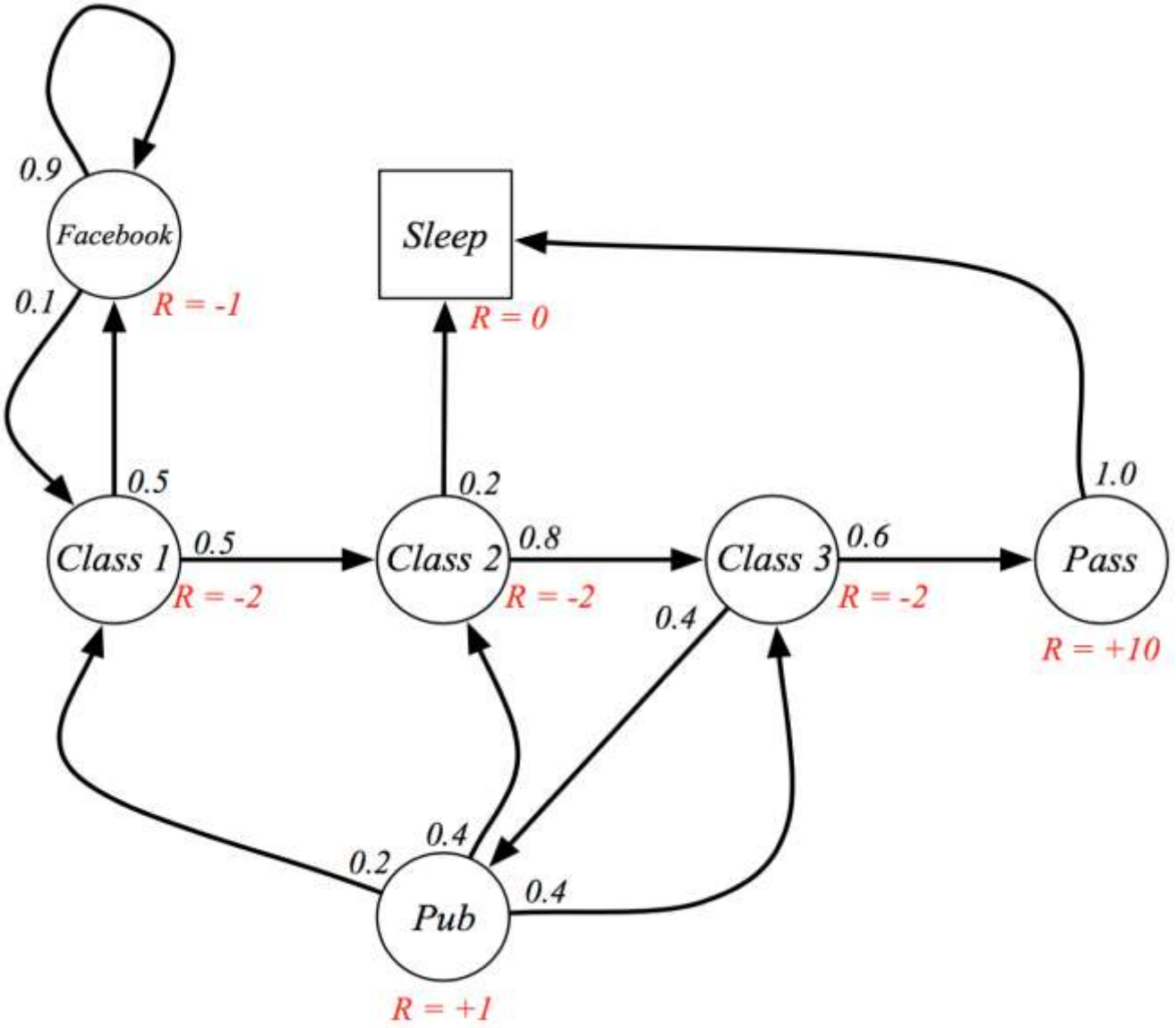
Paradigma de învățare ranforsată



Formalizarea cu Procese de Decizie Markov



Formalizarea cu Procese de Decizie Markov



Formalizarea cu Procese de Decizie Markov

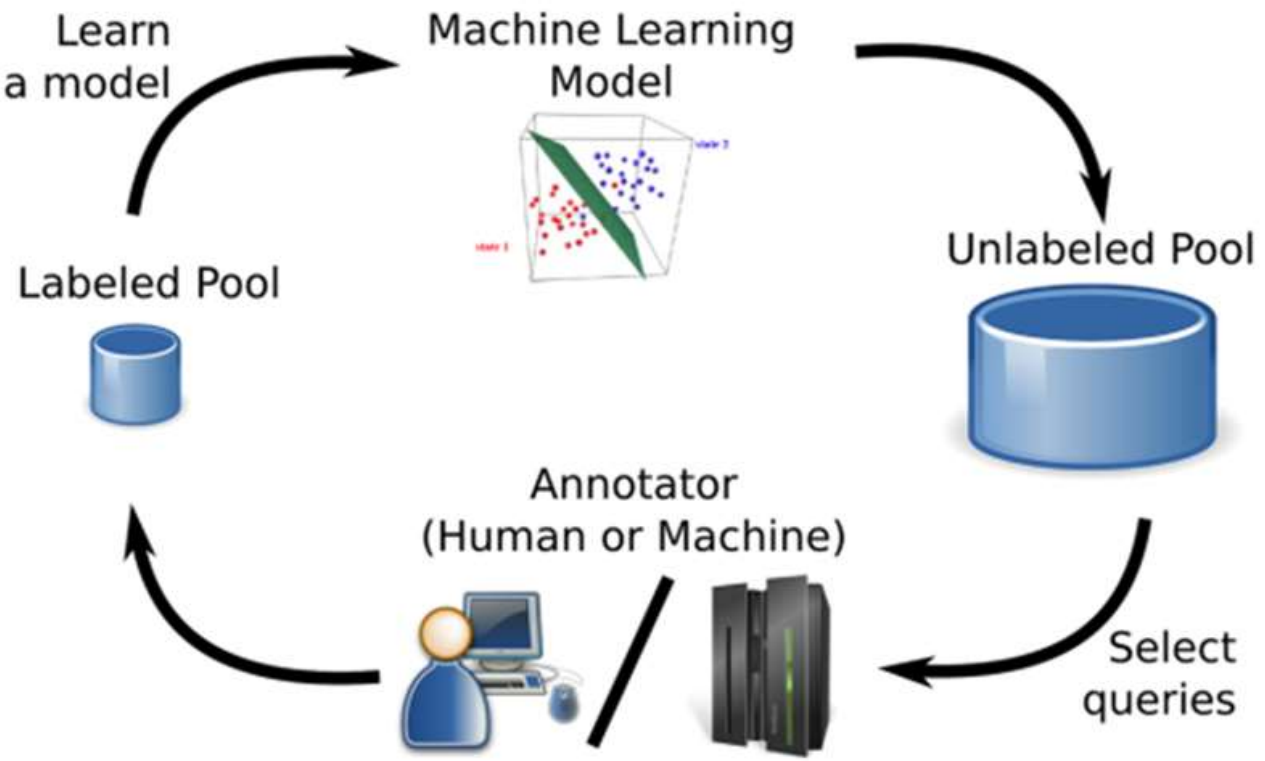
- Soluția bazată pe programare dinamică (grafuri mici) sau aproximare (grafuri mari)
- Scop: selectarea acțiunilor pentru a maximiza recompensa totală finală
- Acțiunile pot avea consecințe pe termen lung
- Sacrificarea unei recompense imediate poate conduce la câștiguri mai mari pe termen lung

Conduce la strategii noi de joc

- Exemplu AlphaGo:
- Comentator 1: “That’s a very strange move”
- Comentator 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/

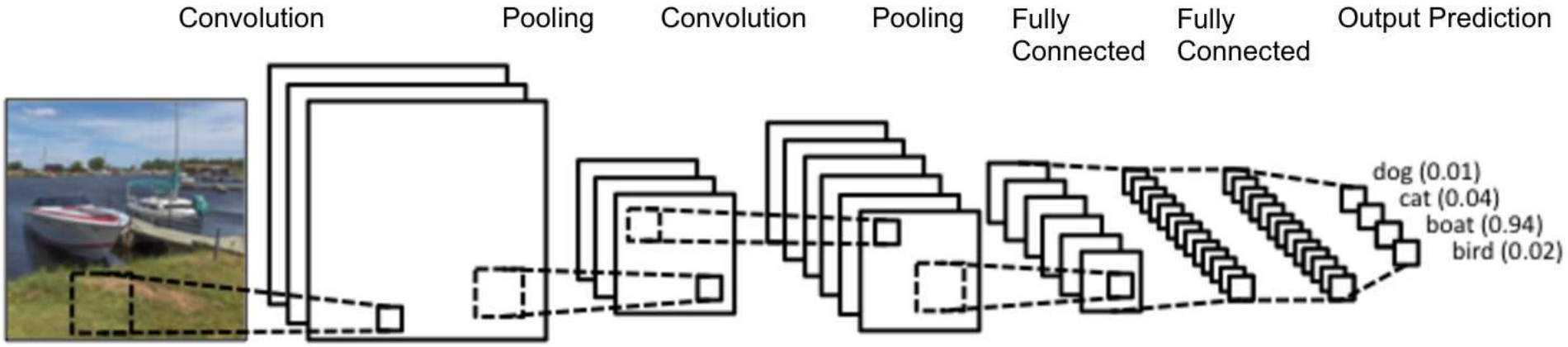
Învățarea activă

- Având un set mare de exemple netichetate, trebuie să alegem un subset mult mai mic pe care să îl etichetăm pentru a obține un clasificator cât mai bun



Învățarea prin transfer

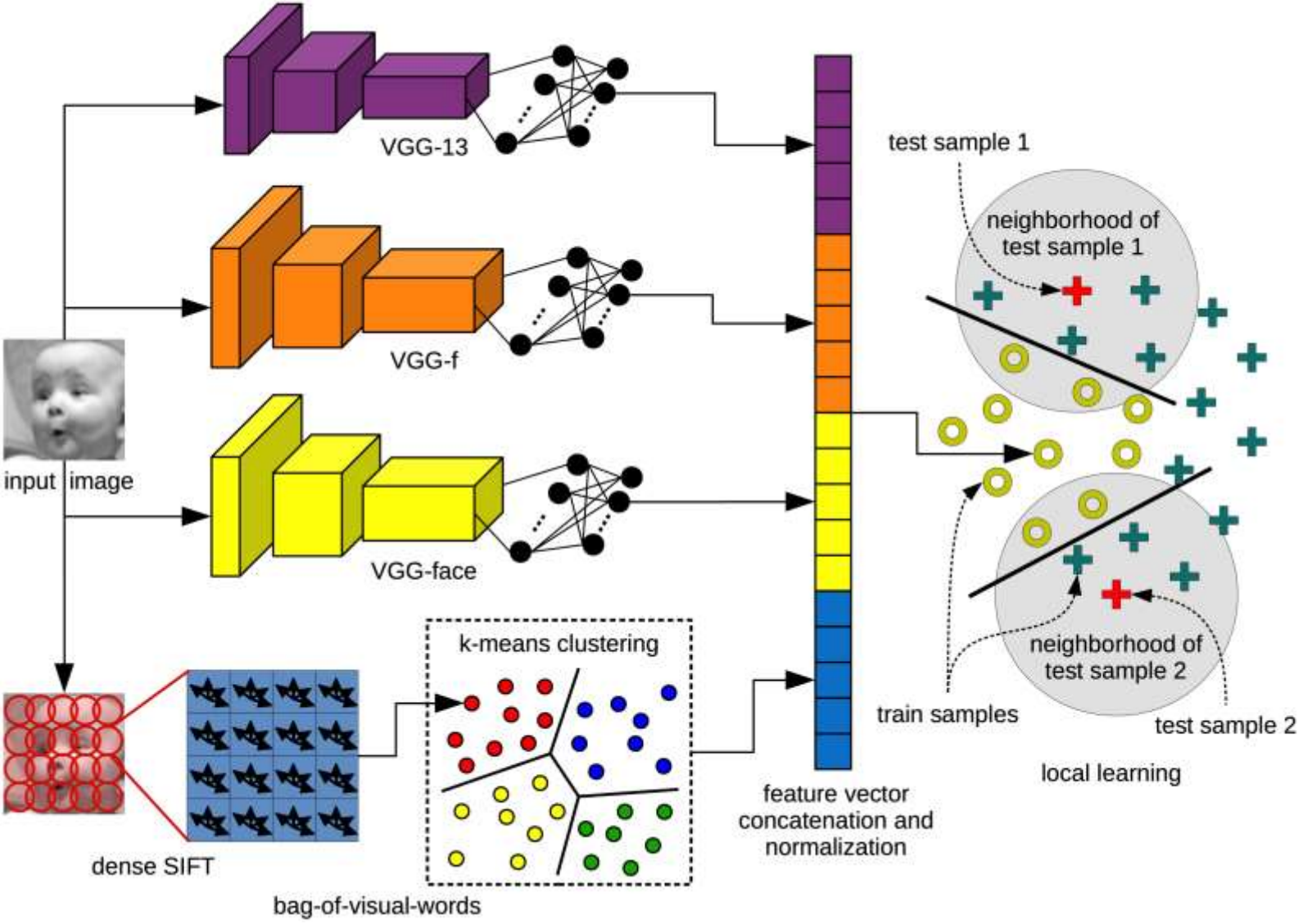
- Pornind la un model antrenat pe un domeniu / o problemă anume, doresc să îl folosesc pentru o altă problemă / alt domeniu
- Exemplu 1: rețele neuronale convoluționale



Alte clase de obiecte (mai specifice), recunoaștere facială, clasificare de texturi, etc.

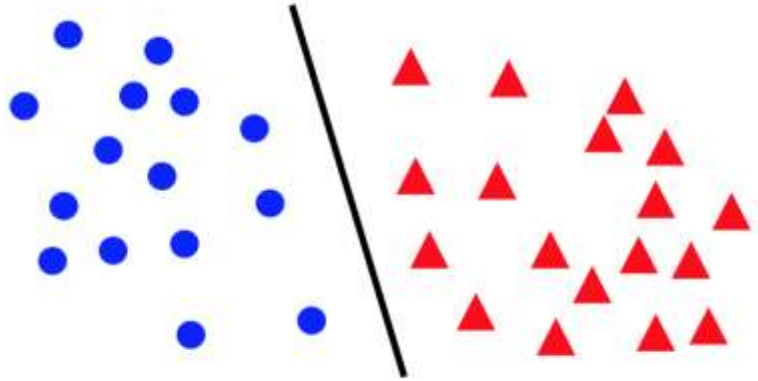
Învățarea prin transfer

- Exemplu 1: recunoașterea expresiilor faciale [Georgescu et al. Access2019]



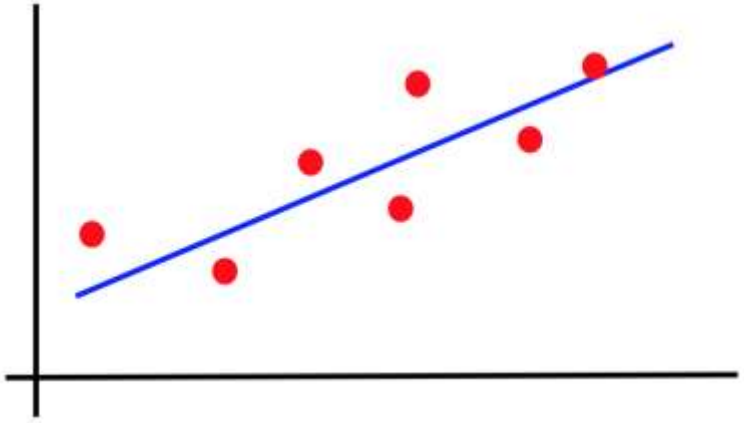
Estimarea vârstei unei persoane din imagini

- Clasificare?



Ce vârstă?

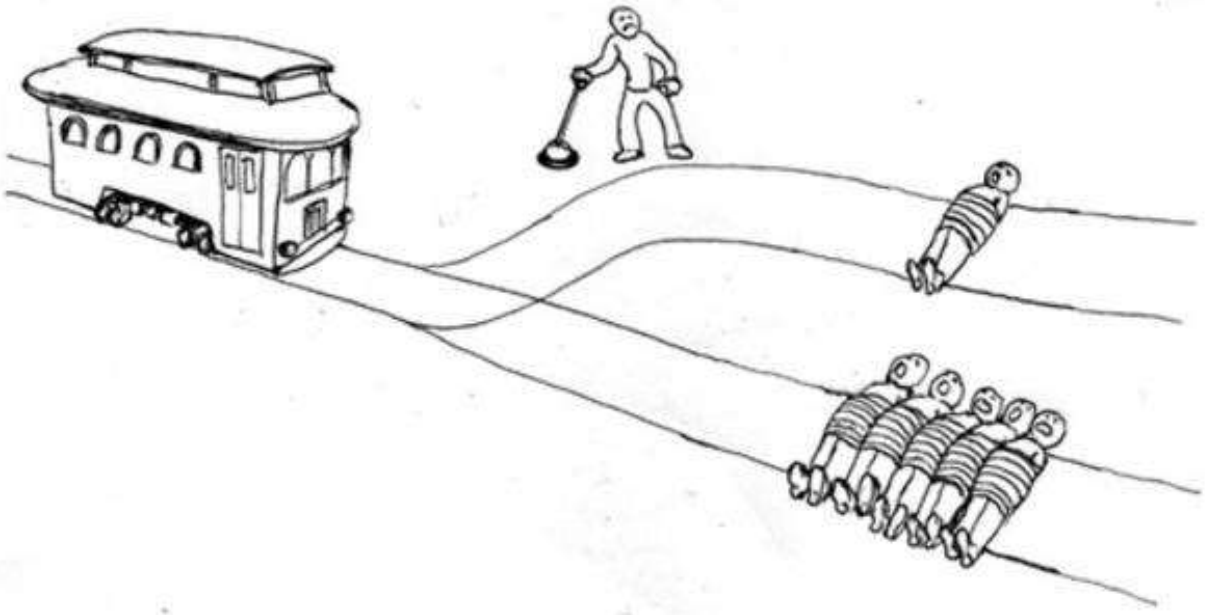
- Regresie?



- Alt tip de învățare?

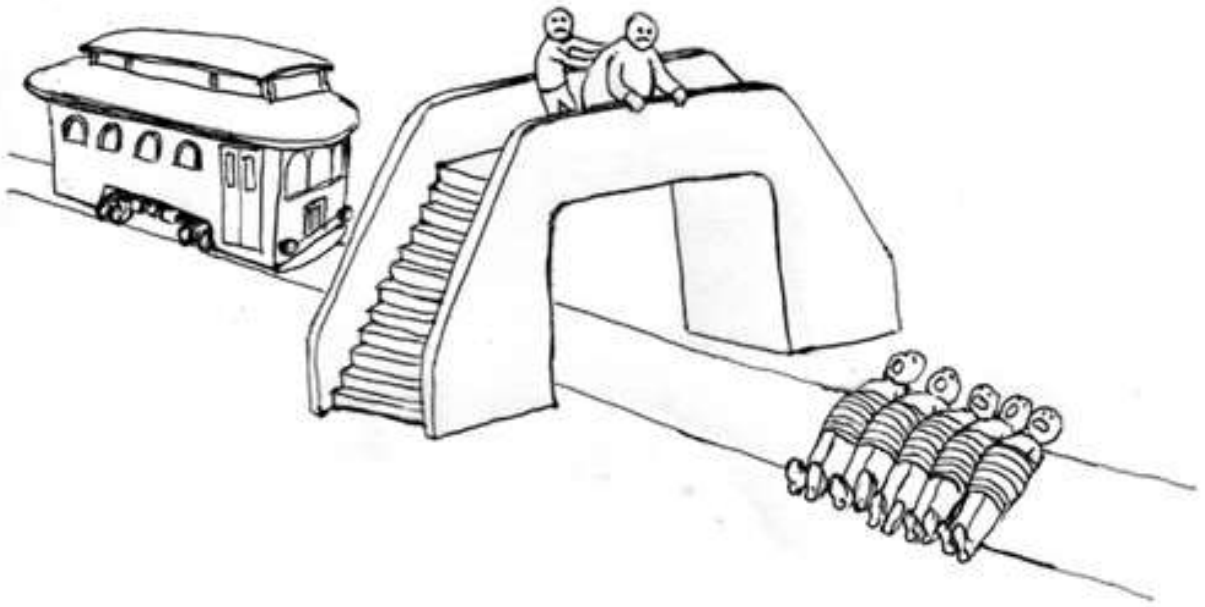
Multe aplicații interesante, dar…

- Ce este etic și ce nu?
- Trolley paradox



Multe aplicații interesante, dar…

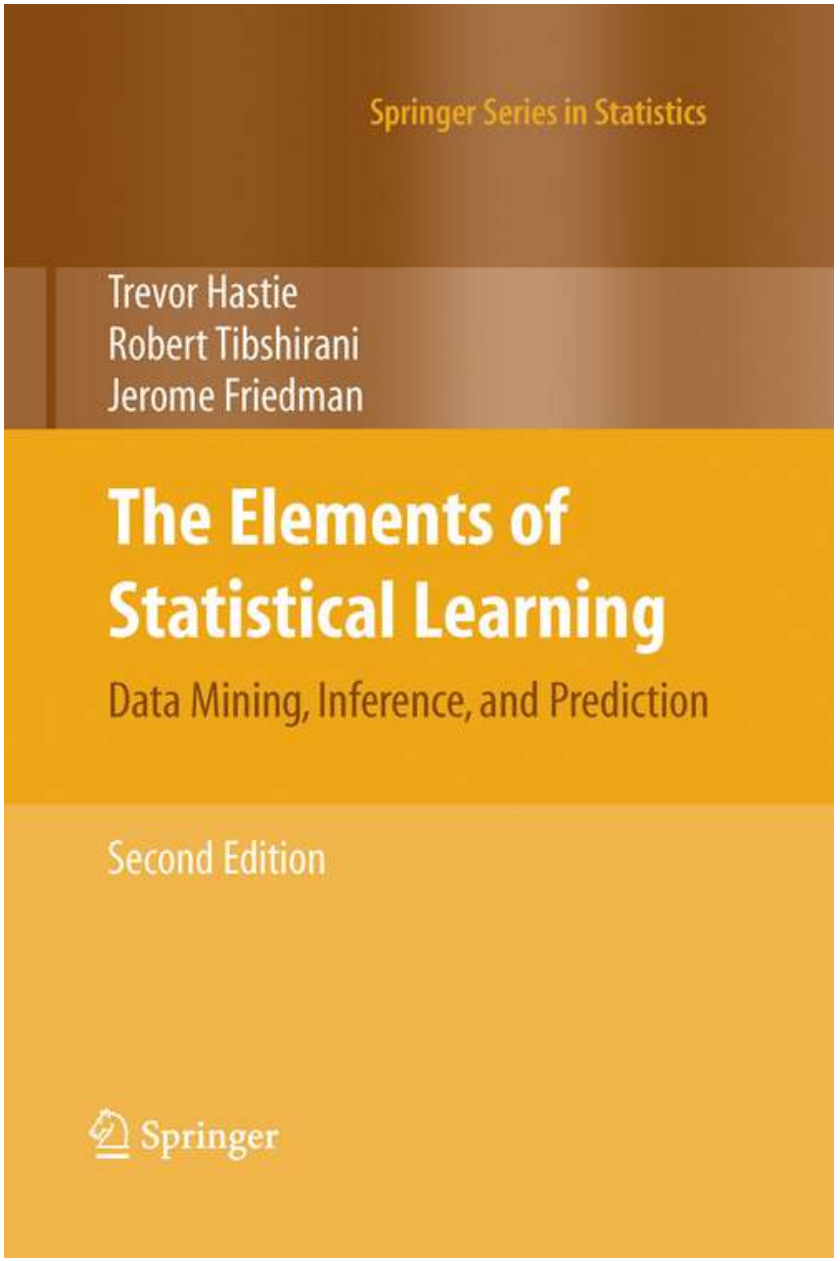
- Ce este etic și ce nu?
- Trolley paradox

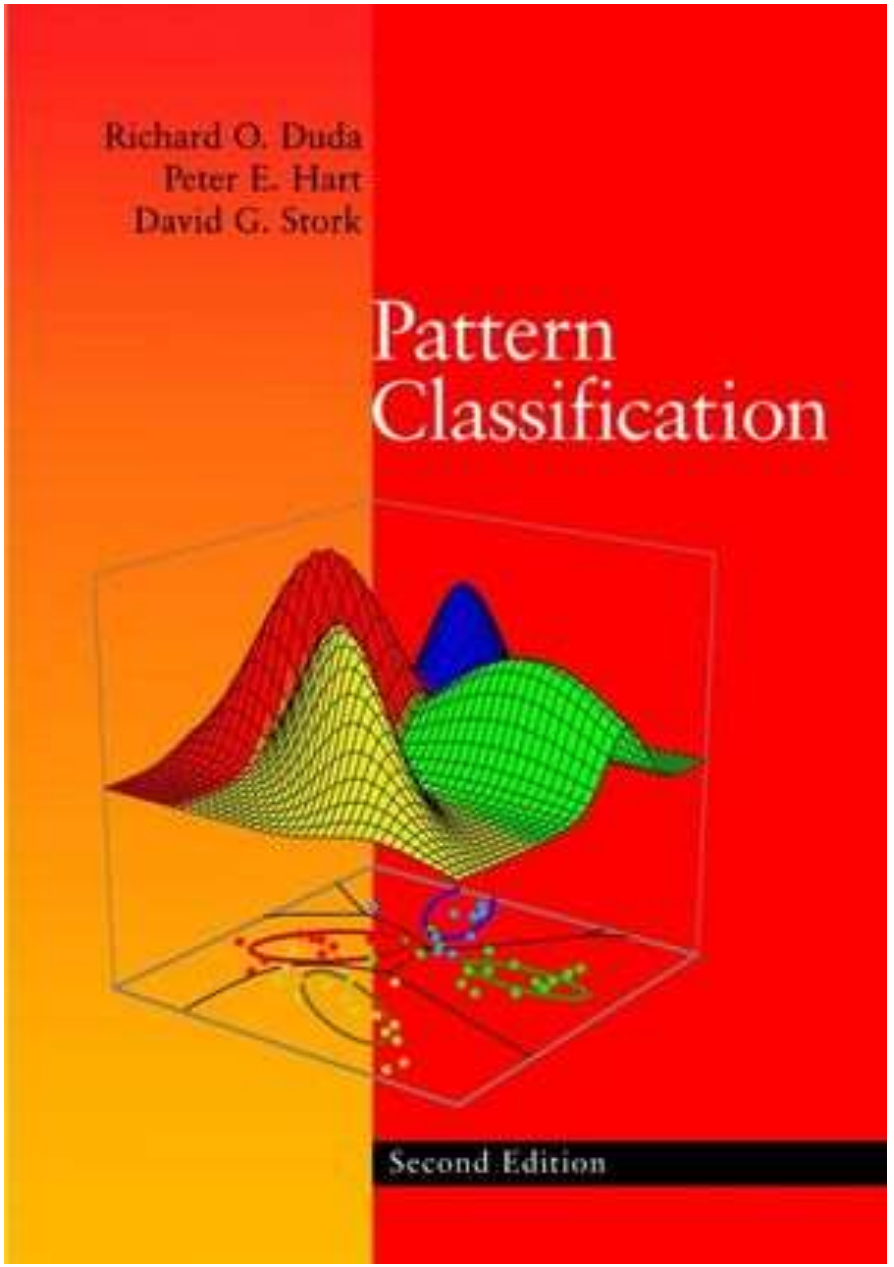


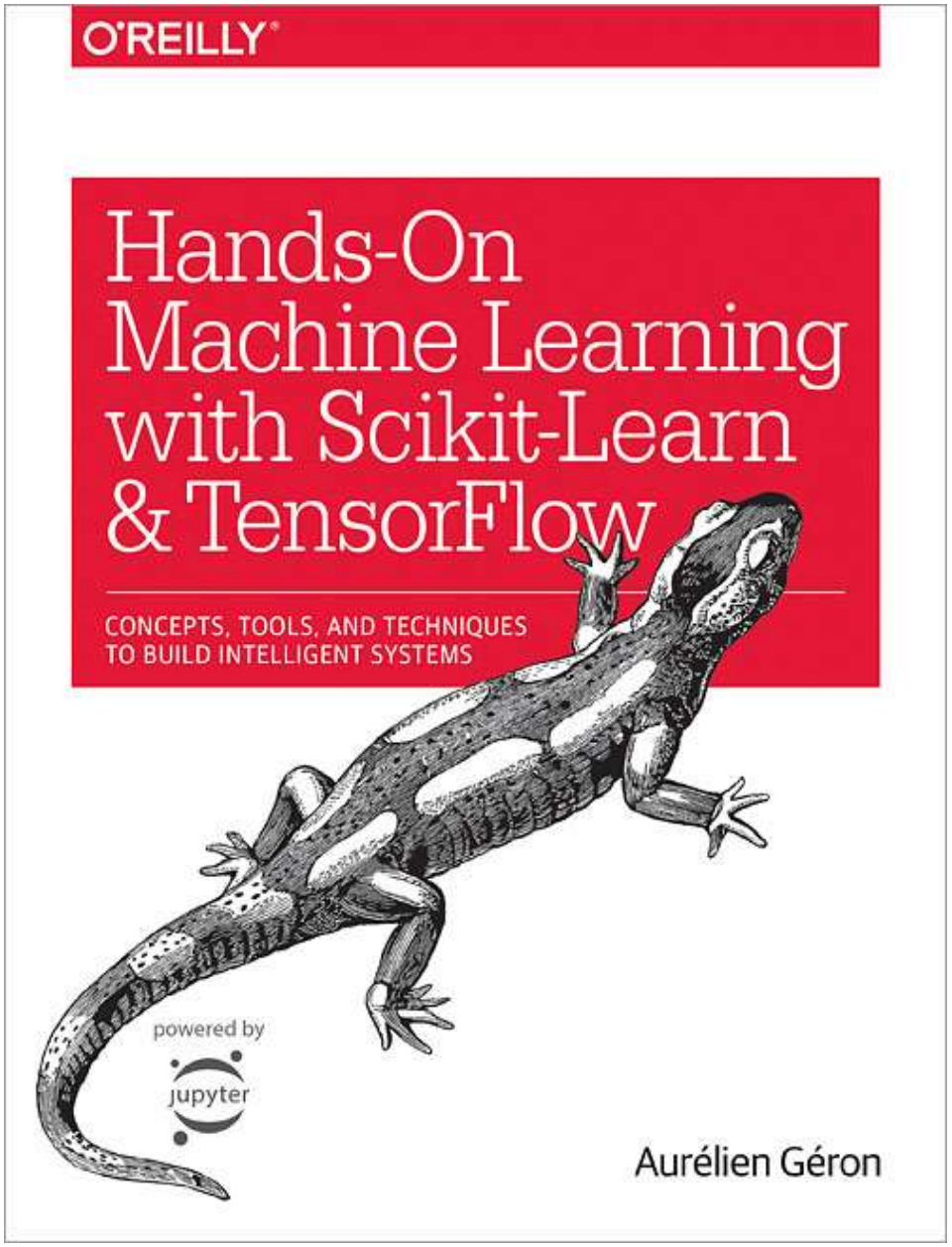
Multe aplicații interesante, dar…

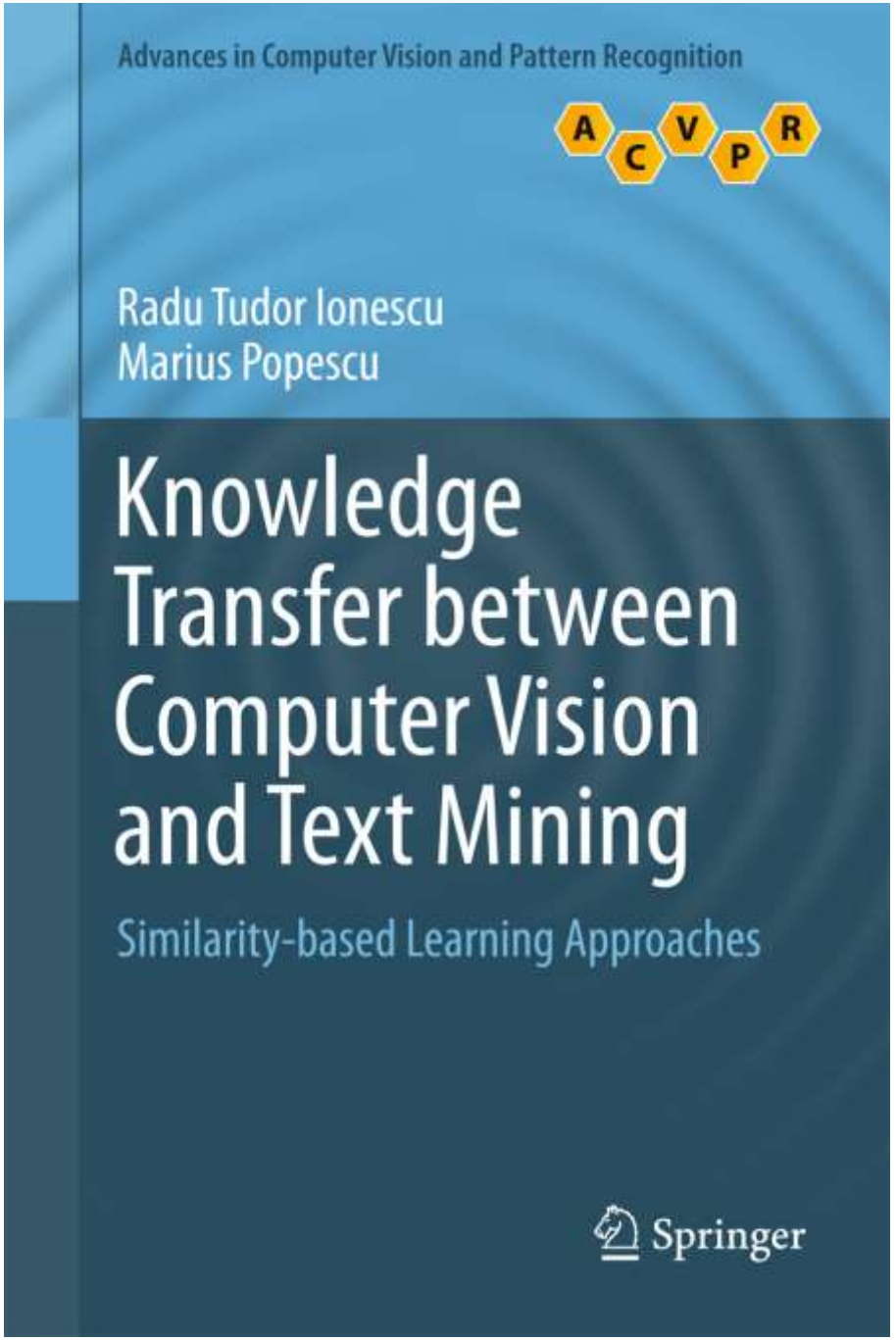
- Ce este etic și ce nu?
- Trolley paradox
- http://moralmachine.mit.edu

Bibliografie


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
A C V P R
Radu Tudor Ionescu
Marius Popescu
Knowledge Transfer between Computer Vision and Text Mining
Similarity-based Learning Approaches
Springer