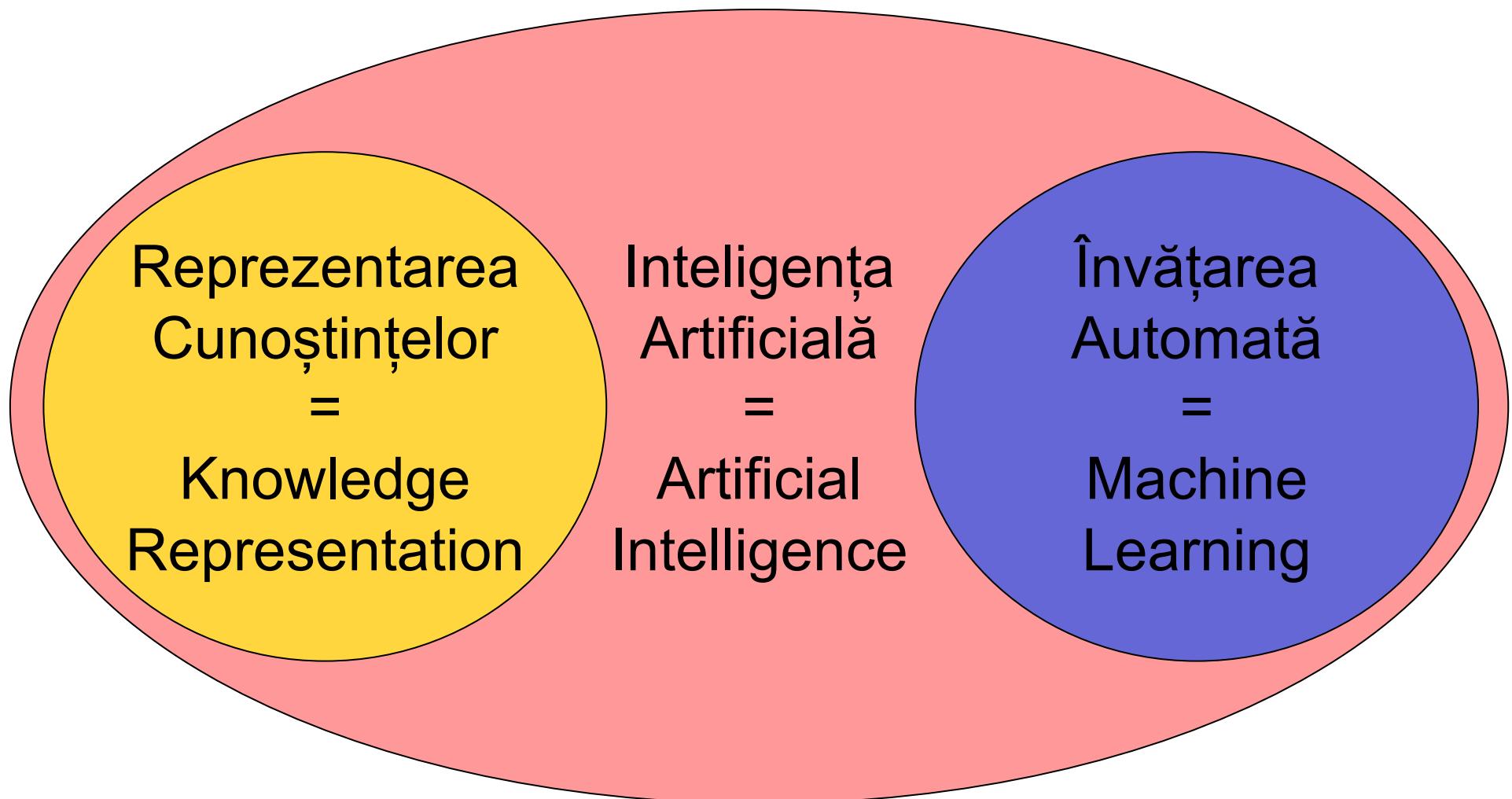


# Inteligenta artificială. Învățare automată. Concepte de bază.

Prof. Dr. Radu Ionescu  
[raducu.ionescu@gmail.com](mailto:raducu.ionescu@gmail.com)

Facultatea de Matematică și Informatică  
Universitatea din București

# Inteligenta artificială și învățarea automată



# Profesori - Învățarea automată

- Cursuri:
  - Radu Ionescu (raducu.ionescu@gmail.com)
- Laboratoare:
  - Alin Croitoru (alincroitoru97@gmail.com)
  - Vlad Hondru (vlad.hondru25@gmail.com)
  - Anca Iordăchescu (iordachescu.ancamihaela@gmail.com)
  - Dana Dăscălescu (danadascalescu00@gmail.com)
  - Sergiu Tălmăcel (sergiu.victor890@gmail.com)
  - Diana Grigore (diana.grigore13@s.unibuc.ro)

# 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":
  - 30% din notă se acordă pentru un proiect individual în săptămâna a 8-a (respectiv a 14-a)
  - 20% din notă se acordă pentru un test de laborator în săptămâna a 8-a (respectiv a 14-a)

# Sistem de notare

- 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 1-30\* => nota maximă 10
  - Locurile 31-50\* => nota maximă 9
  - Locurile 51-90\* => nota maximă 8
  - Locurile 91-120\* => nota maximă 7
  - Locurile 121-150\* => nota maximă 6
  - Locul 151\* sau mai jos => nota maximă 5
- \* configurația se poate schimba în funcție de numărul de participanți
- Proiectul trebuie prezentat după ultimul laborator (se pot scădea până la 2 puncte pentru prezentare și documentație)
- Pentru nota  $\geq 5$ , trebuie depășită performanța baseline

# Sistem de notare

- Proiectele (cod+documentație) se trimit la adresa:  
`fmi.unibuc.ia@gmail.com`
- Trimiteti doar fisiere .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 cateva cerințe (cu punctaj separat) care să vă ajute la găsirea modelului și hiperparametrilor optimi.

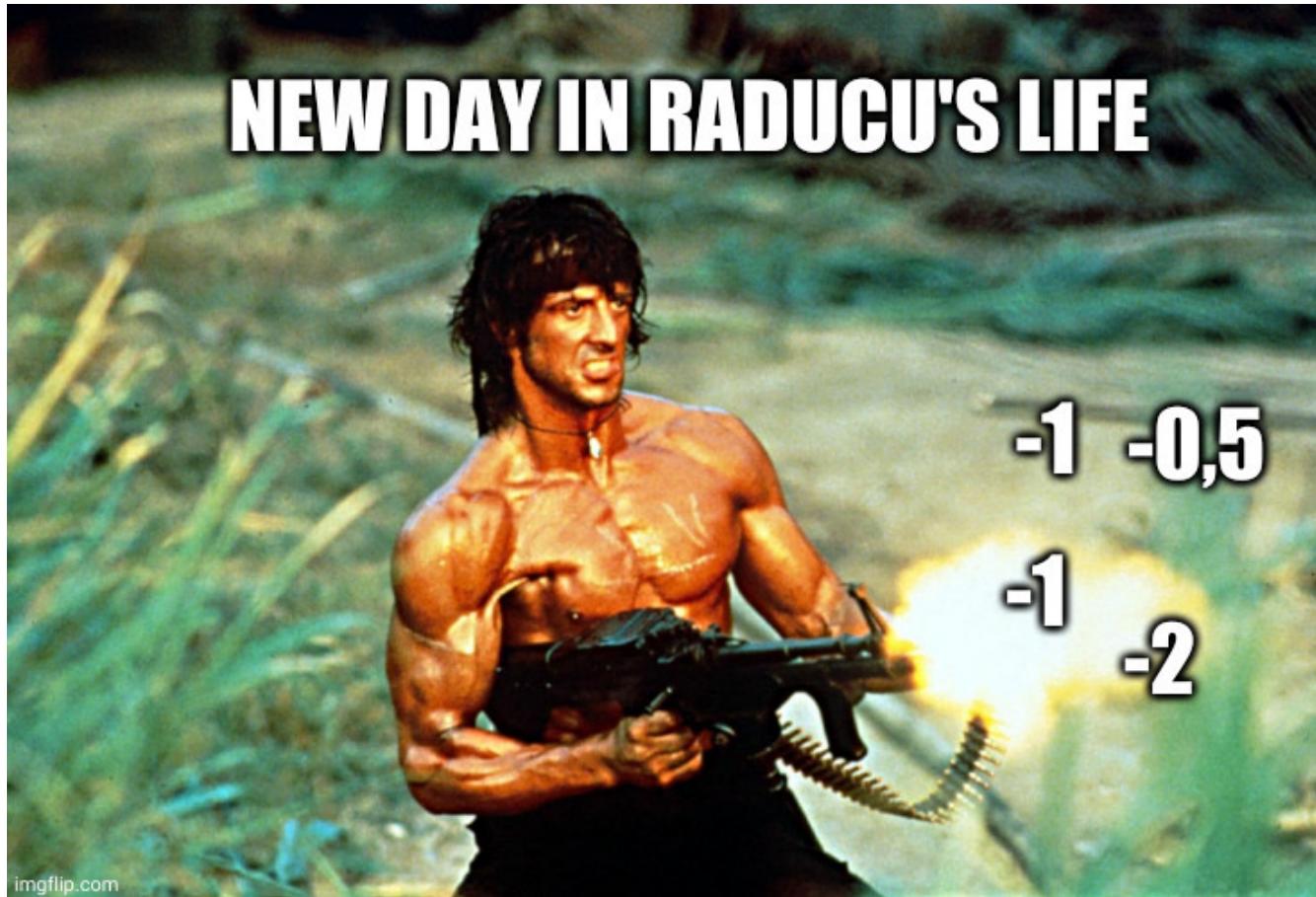
# Sistem de notare

- 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, și.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 punct în plus la nota din examen
- Până la 1 punct în plus la nota din laborator

# Sistem de notare

- Codul va fi verificat cu soft-uri anti-plagiat
- NU este permisă preluare codului de pe web (sub nicio formă, nici măcar din ...)
- NU este permisă preluare codului de la colegi

# Un aspect foarte important!



# Exemple de plagiarism

```
3 # 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

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'))
```

# Exemple de plagiarism

```
batch_size = 64
1 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 #
#####
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)
```

# 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))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

```
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())
```

# Exemple acceptable

3 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 import np\_utils 11 import to\_categorical  
import numpy as np  
import plot as plt

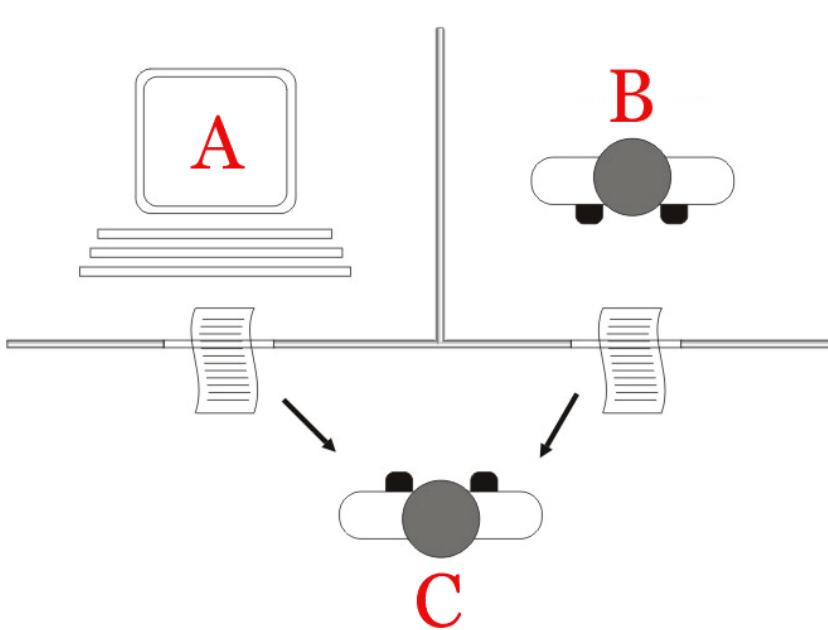
# Exemple acceptable

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

nume_imagini = []
i = 0
ordine = dict()
9 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)
    11 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 învăță
- **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 îmbunătășește 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

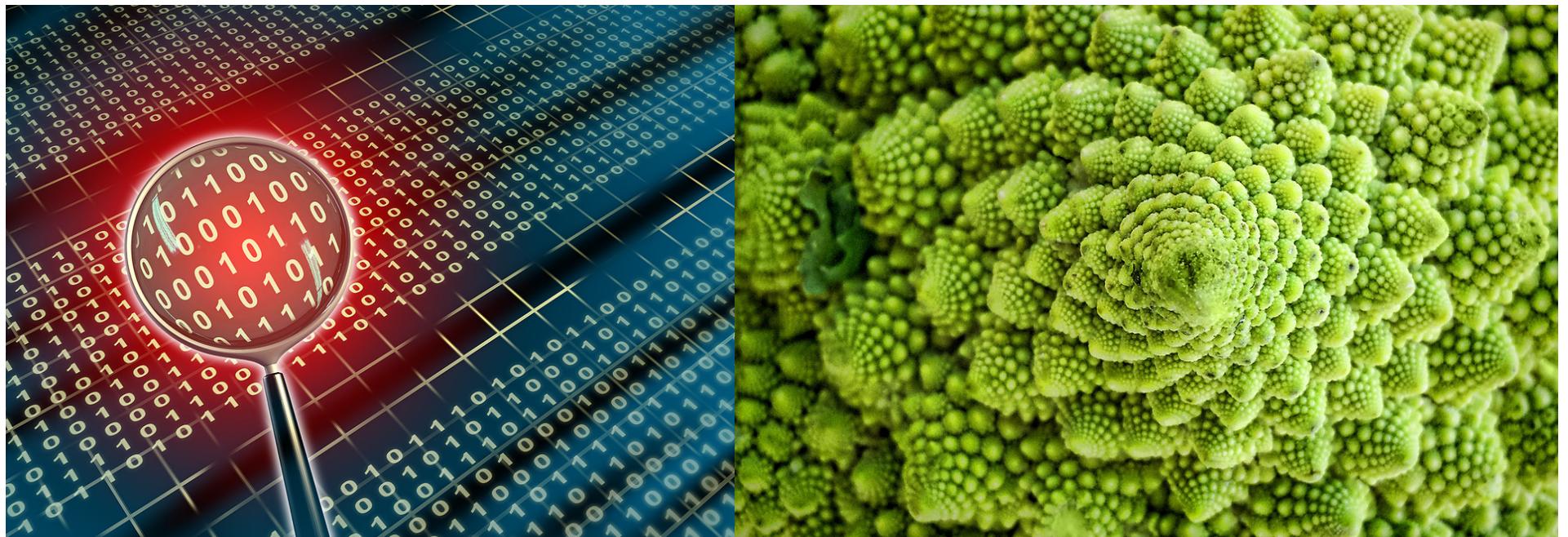
- strong / generic / true AI  
(vezi definiția lui Turing)
- weak / narrow AI  
(se focusează pe o anumită problemă)

# Când se aplică învățarea automată?

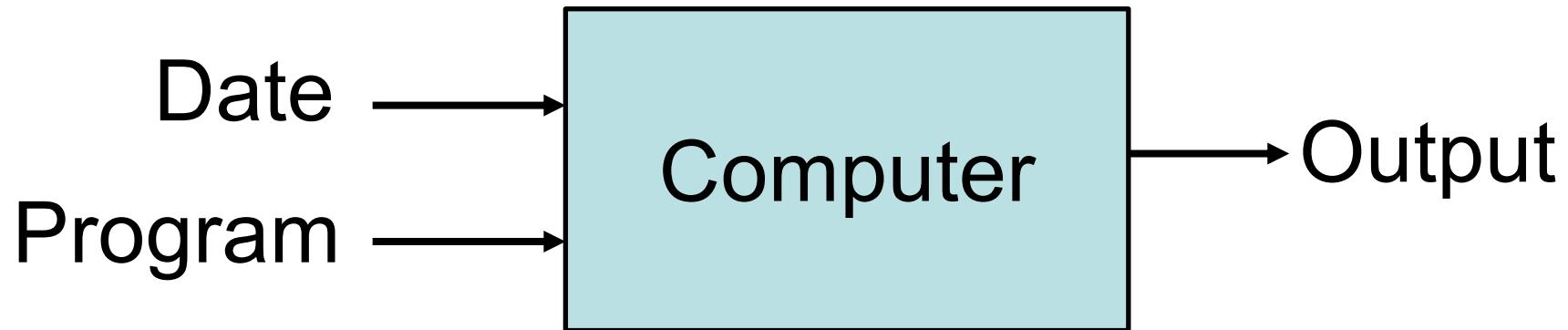
- Se aplică în situații în care este foarte greu (imposibil) să definim un set de reguli de mână / să scriem un program
- Exemple de probleme unde putem aplica învățarea automată:
  - Detectarea facială
  - Înțelegerea vorbirii
  - Prezicerea prețului acțiunilor
  - Recunoașterea obiectelor

# Esența învățării automate

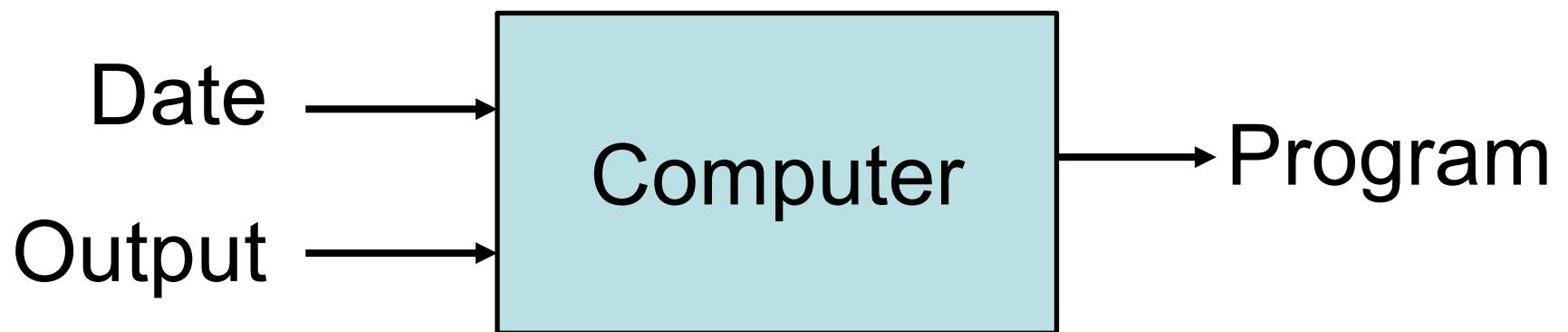
- Există un tipar
- Dar nu îl putem exprima programatic / matematic
- Avem date / exemple în care regăsim acest tipar



# 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

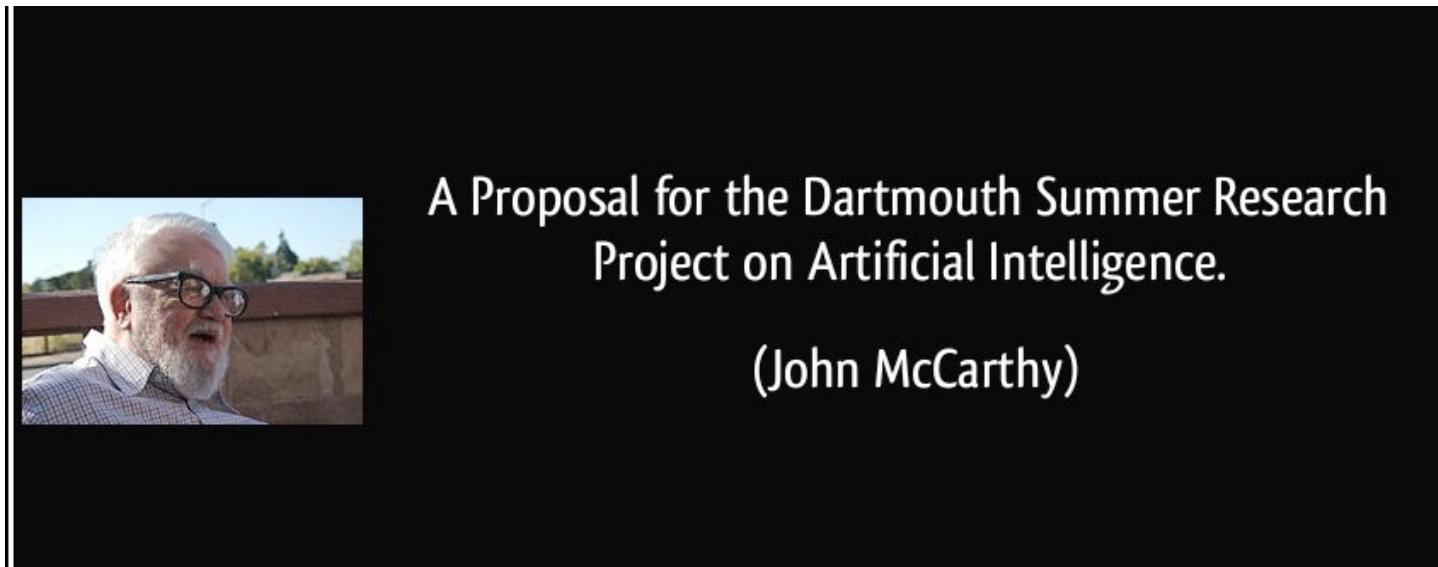
[Kevin Murphy] algorithms that

- 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

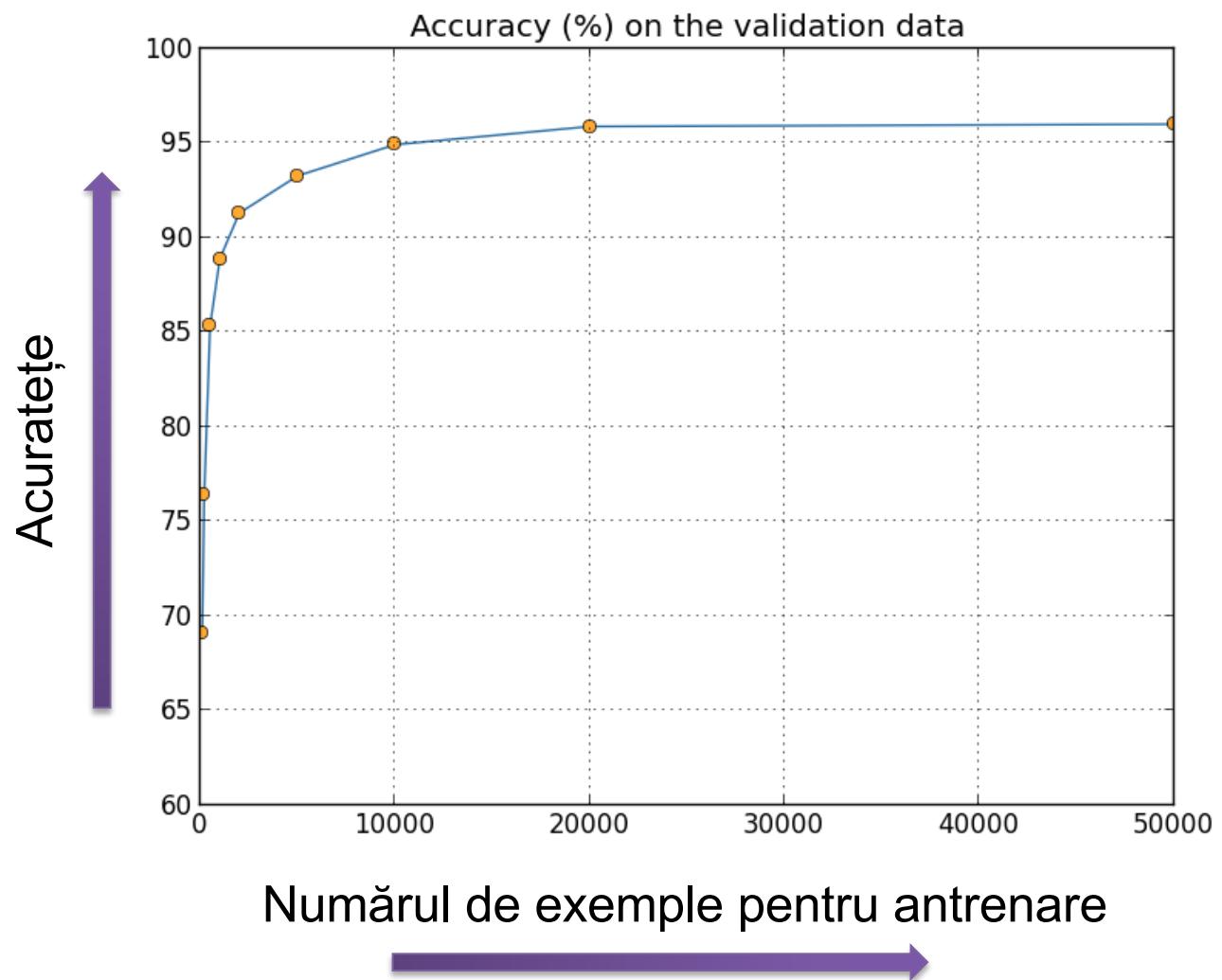
- “We propose that a 2 month, 10 man study of artificial intelligence be carried out during the summer of 1956 at Dartmouth College in Hanover, New Hampshire.”
- The study is to proceed on the basis of the conjecture that every aspect of learning or any other feature of intelligence can in principle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

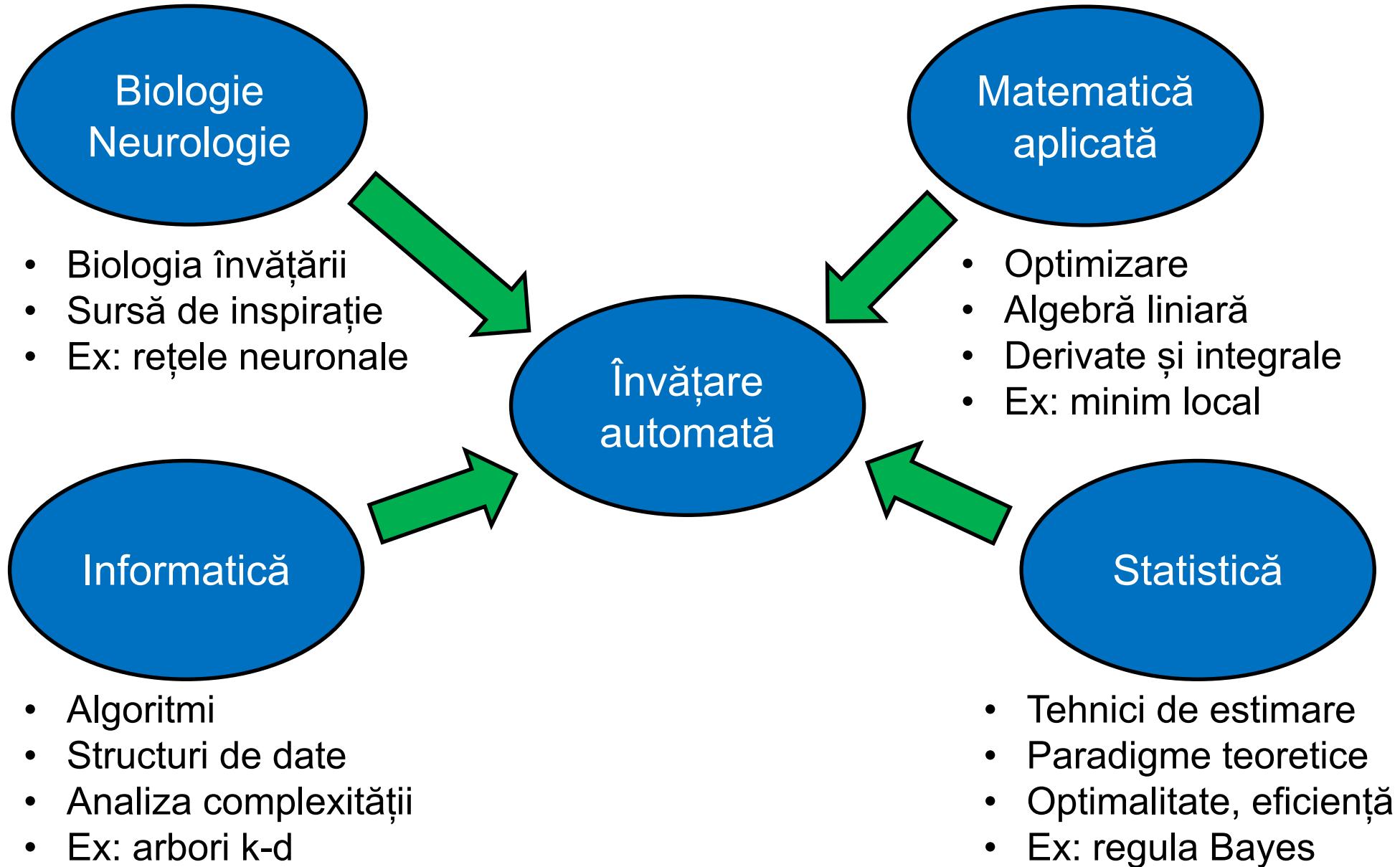
# Esența învățării automate

- Input: X (imagini, texte, email-uri...)
- Output: Y (spam sau non-spam...)
- Funcție Target (necunoscută)  
 $f: X \rightarrow Y$  (realitatea / "adevărata" mapare)
- Date  
 $(x_1, y_1), (x_2, y_2), \dots (x_N, y_N)$
- Model  
 $g: X \rightarrow Y$   
 $y = g(x) = sign(w^T x)$

# Esența învățării automate

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

# Ce cunoștințe sunt necesare?



# 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



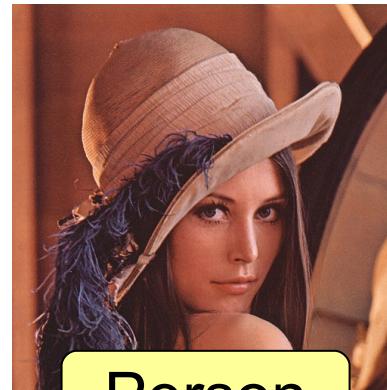
Car



Car



Person



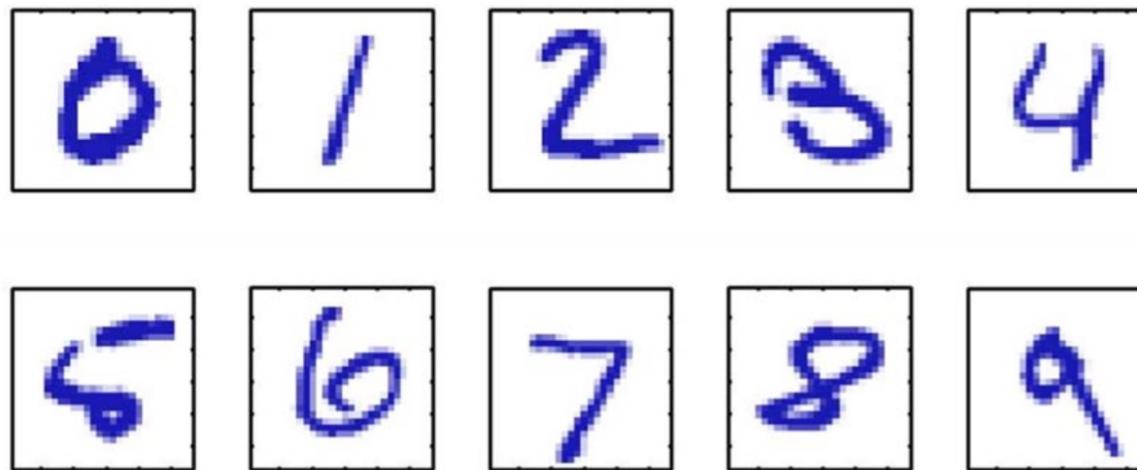
Person



Dog

# Învățare supervizată

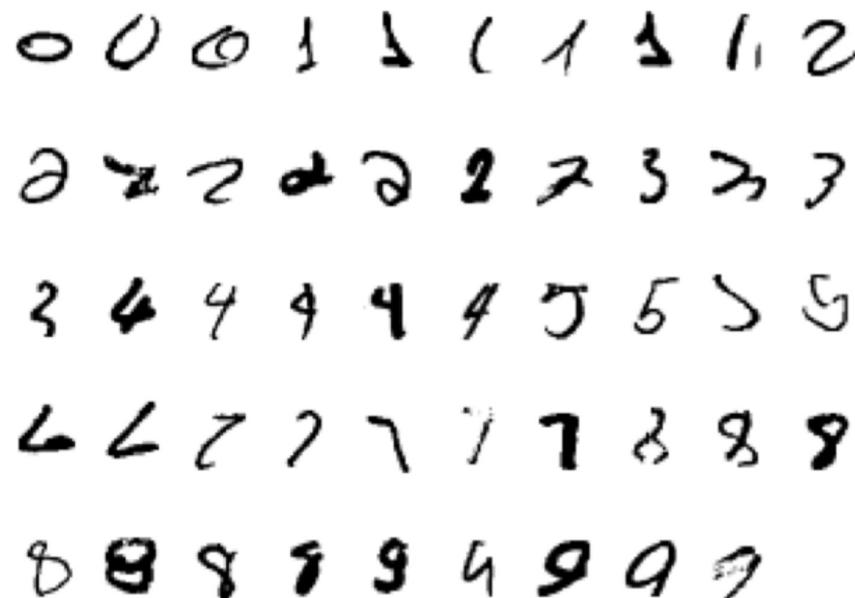
- Exemplu 2: recunoașterea caracterelor scrise de mână (setul de date MNIST)



- Imagini de  $28 \times 28$  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

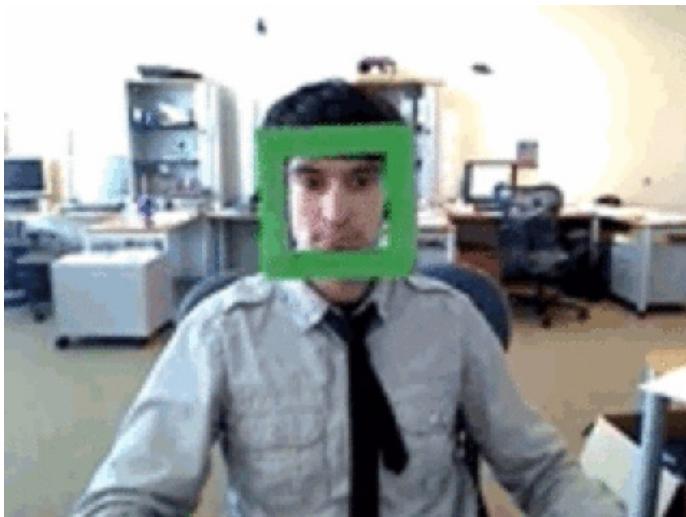
- Exemplu 2: recunoașterea caracterelor scrise de mână (setul de date MNIST)



- 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 convolutionale)
- 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

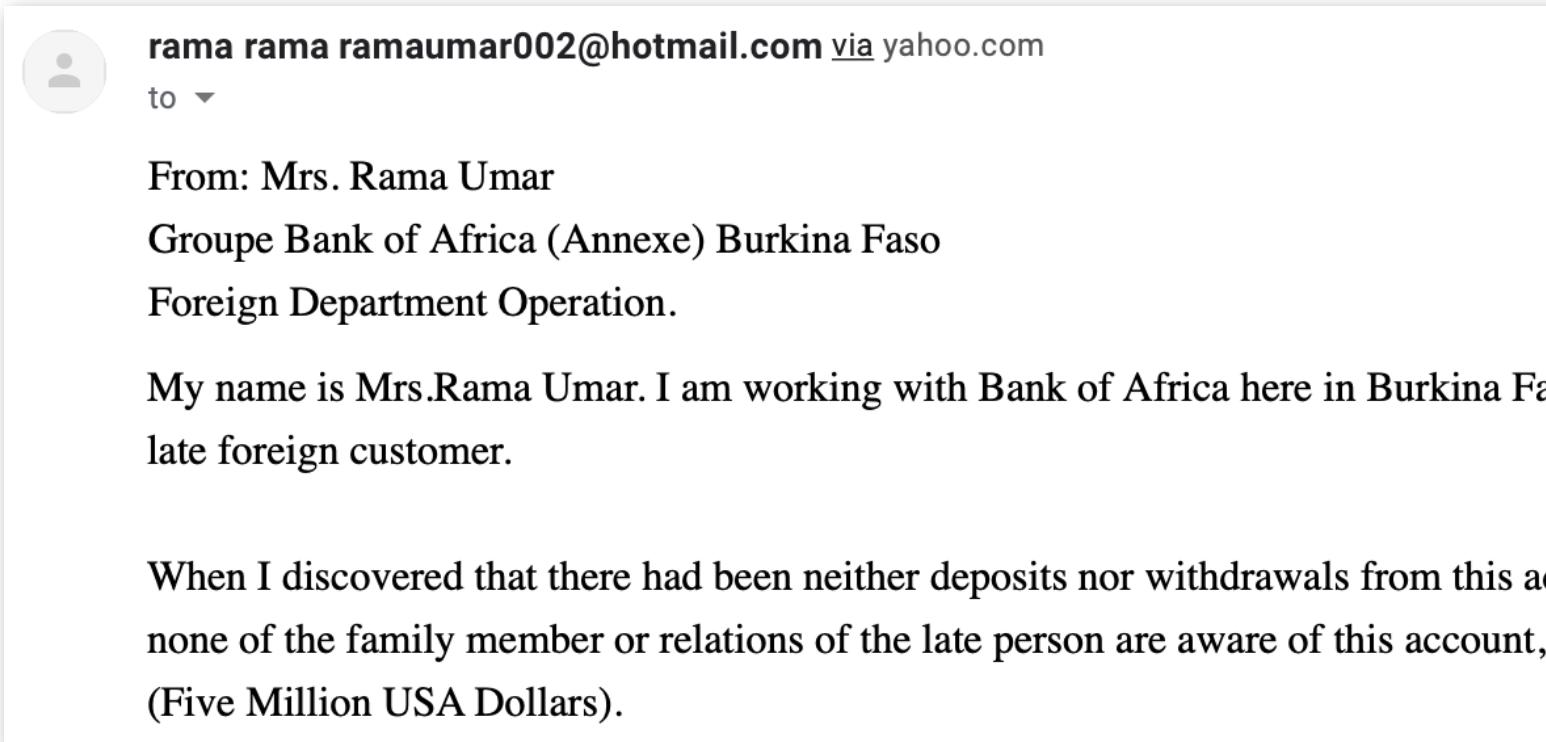
- Exemplu 3: detectare facială



- 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



The image shows a screenshot of an email inbox. A single email message is selected, indicated by a blue border. The message is from 'rama rama ramaumar002@hotmail.com via yahoo.com' to 'to'. The subject line is 'From: Mrs. Rama Umar' followed by three lines of text: 'Groupe Bank of Africa (Annexe) Burkina Faso', 'Foreign Department Operation.', and 'My name is Mrs.Rama Umar. I am working with Bank of Africa here in Burkina Faso. I have been assigned to handle a late foreign customer.' Below this, there is another block of text: 'When I discovered that there had been neither deposits nor withdrawals from this account, I realized that none of the family member or relations of the late person are aware of this account, especially since it has been inactive for so long. I would like to offer you a service to withdraw the amount of \$5,000,000 (Five Million USA Dollars).'. The rest of the message is cut off at the bottom.

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 Faso. I have been assigned to handle a late foreign customer.

When I discovered that there had been neither deposits nor withdrawals from this account, I realized that none of the family member or relations of the late person are aware of this account, especially since it has been inactive for so long. I would like to offer you a service to withdraw the amount of \$5,000,000 (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



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 Faso. I have been assigned to handle a late foreign customer.

When I discovered that there had been neither deposits nor withdrawals from this account, I was shocked. I checked with all of my colleagues and none of the family member or relations of the late person are aware of this account, which means it has been inactive for a long time. The balance in the account is approximately \$5,000,000 (Five Million USA Dollars).

free	100
money	2
:	:
account	2
:	:



**Yoshua Bengio** <yoshua.bengio@gmail.com>

to Dong-Hyun, Ian, Dumitru, Pierre, Aaron, Mehdi, Ben, Will, Charlie,

Nice slides!

See you next week,

—Yoshua

free	1
money	1
:	:
account	2
:	:

# 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?

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

$$= 3.2$$



Confidență /  
garanția  
performanței?

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

$$= 1.03$$

De ce combinație liniară?

De unde vin aceste ponderi?

# 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
- Aceasta este un task de regresie, deoarece output-ul este unul continuu

# Învățare supervizată

- Exemplu 6: prezicerea dificultății unei imagini



2.78



2.82



3.30



3.62



3.80

easy

image difficulty score

hard

2.81



3.15



3.45



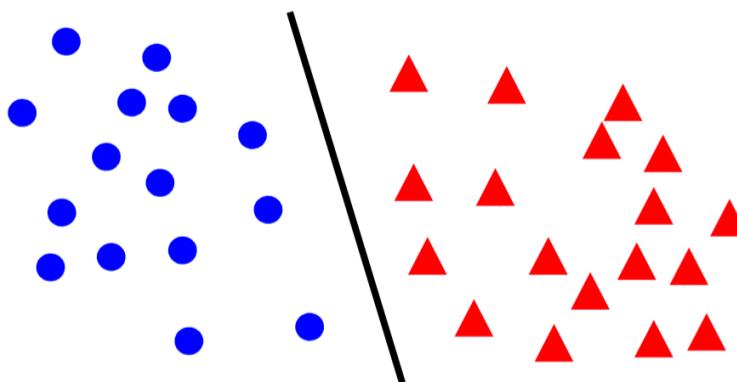
3.64



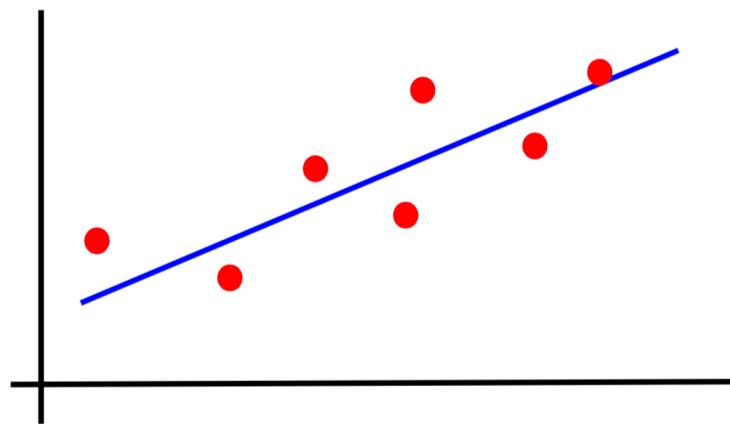
- Scopul este de a prezice cât de dificil ar fi pentru un om să recunoască obiectele din imagine
- Aceasta este un task de regresie, deoarece output-ul este unul continuu

# Formele canonice ale problemelor de învățare supervizată

- Clasificare



- Regresie



# Paradigma de învățare supervizată

Functions  $\mathcal{F}$

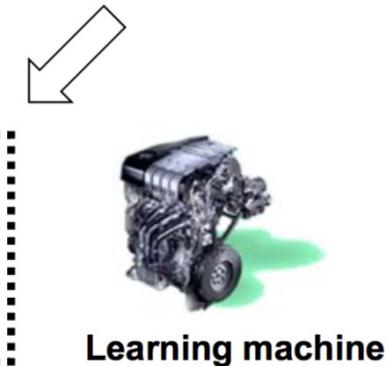
$$\textcolor{red}{f} : \mathcal{X} \rightarrow \mathcal{Y}$$

Training data

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

LEARNING

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



PREDICTION

$$\textcolor{red}{y} = \hat{f}(x)$$

New data

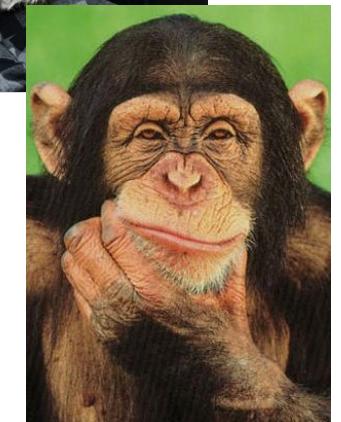
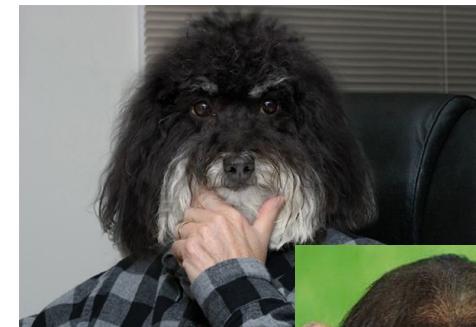
$$x$$

# Modele de învățare supervizată

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

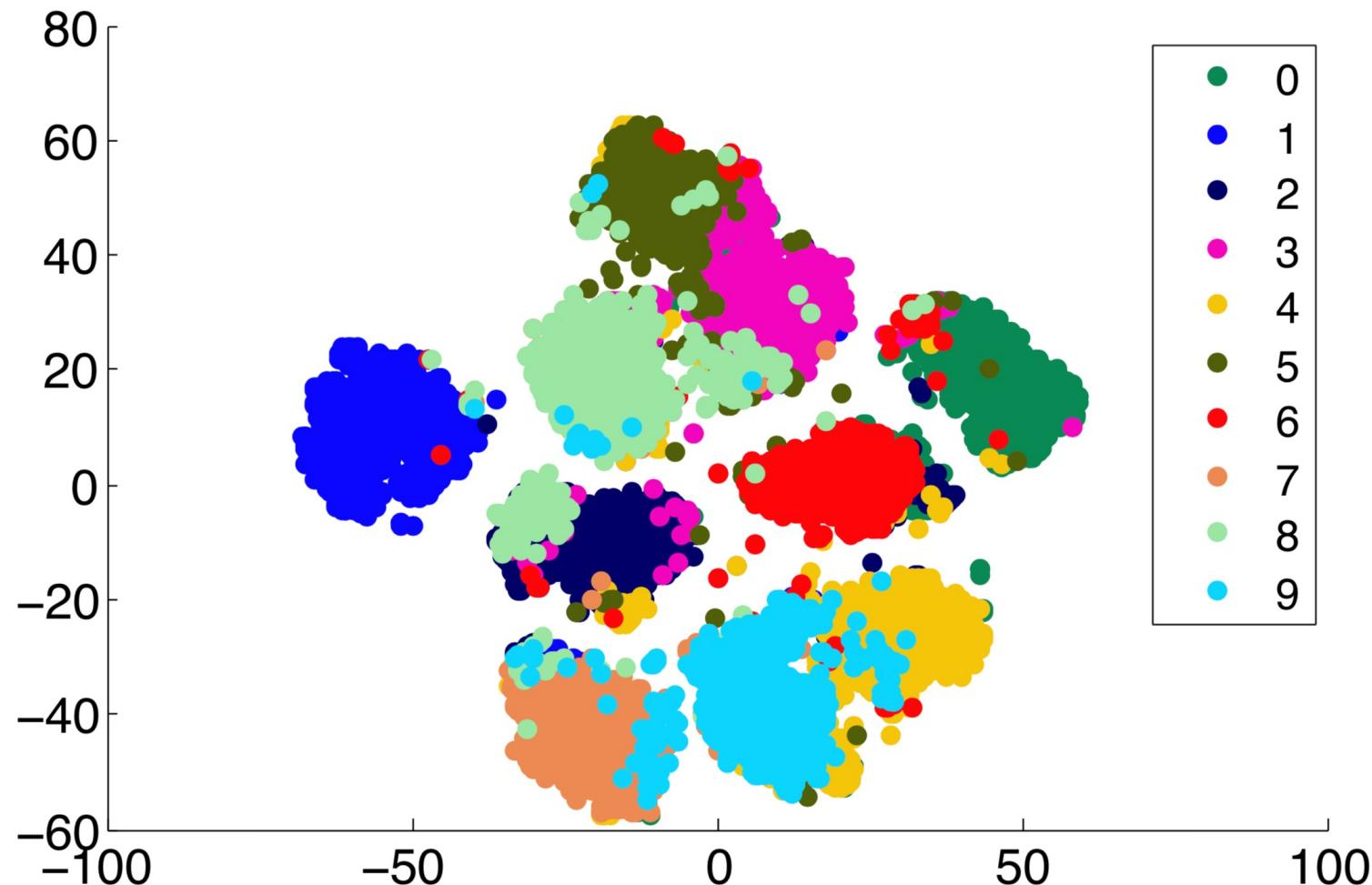
# Învățare nesupervizată

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



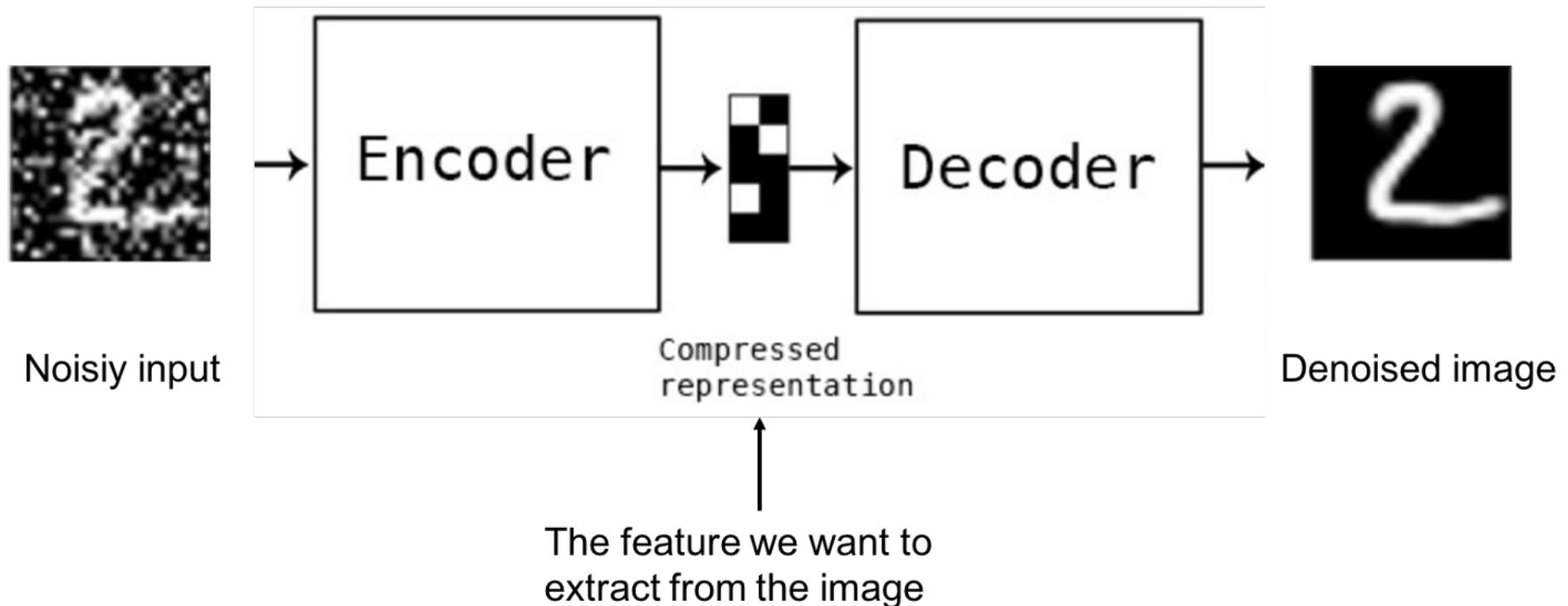
# Învățare nesupervizată

- Exemplu 1: clusterizarea aglomerativă a imaginilor MNIST [Georgescu et al. ICIP2019]



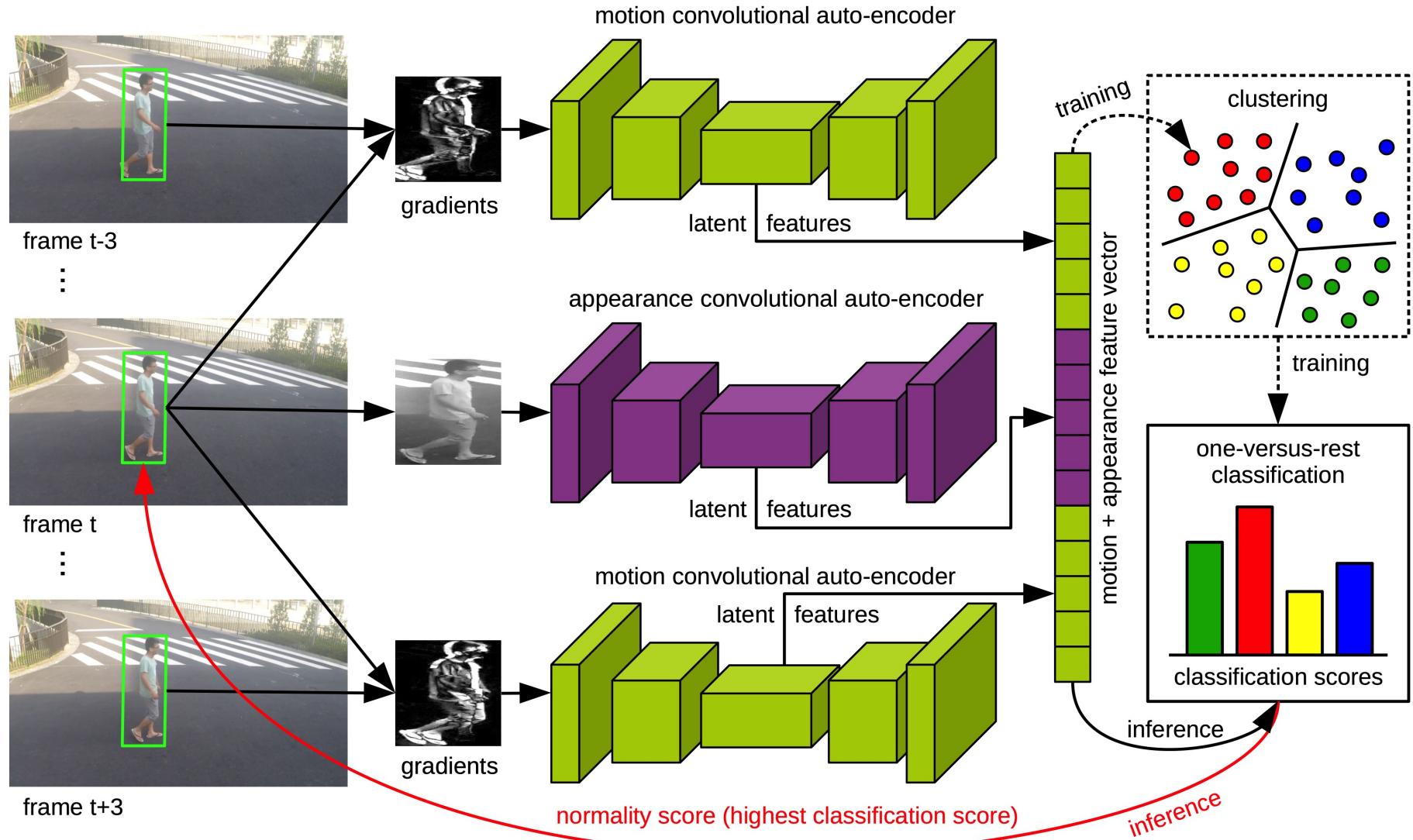
# Învățare nesupervizată

- Exemplu 2: Învățarea de trăsături folosind principiul “bottleneck”



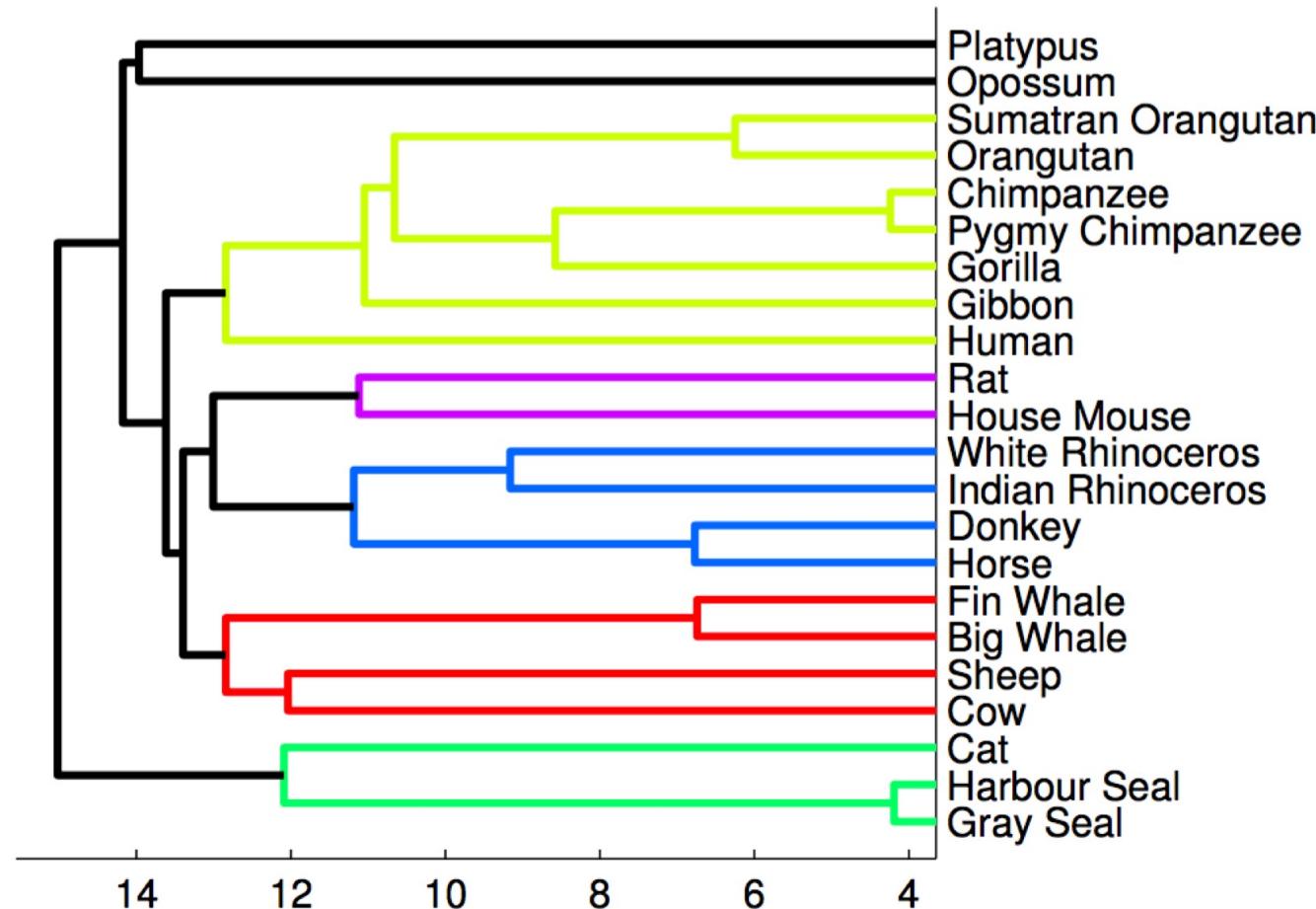
# Învățare nesupervizată

- Exemplu 2: Învățarea de trăsături pentru detectarea evenimentelor anormale [Ionescu et al. CVPR2019]



# Învățare nesupervizată

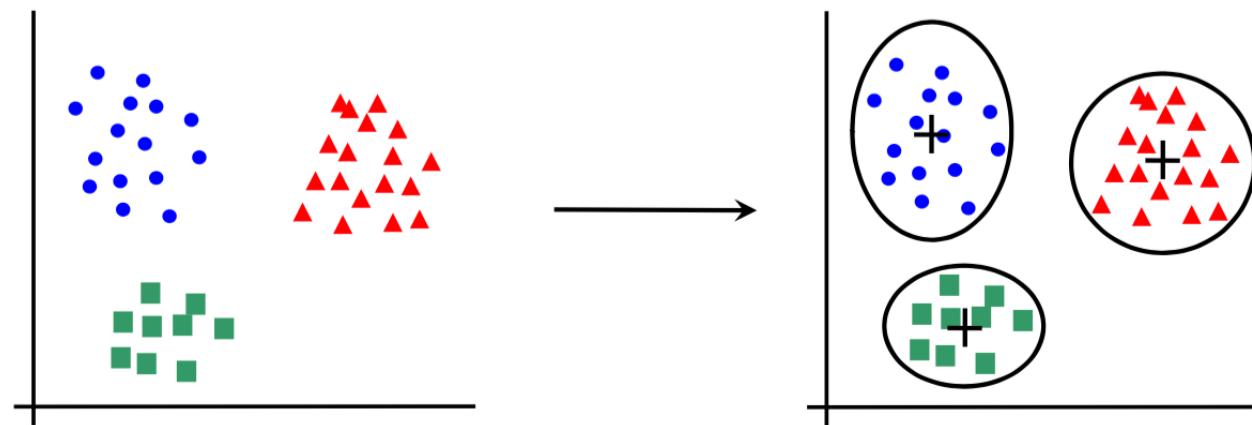
- Exemplu 2: gruparea mamiferelor pe familii, specii, etc.



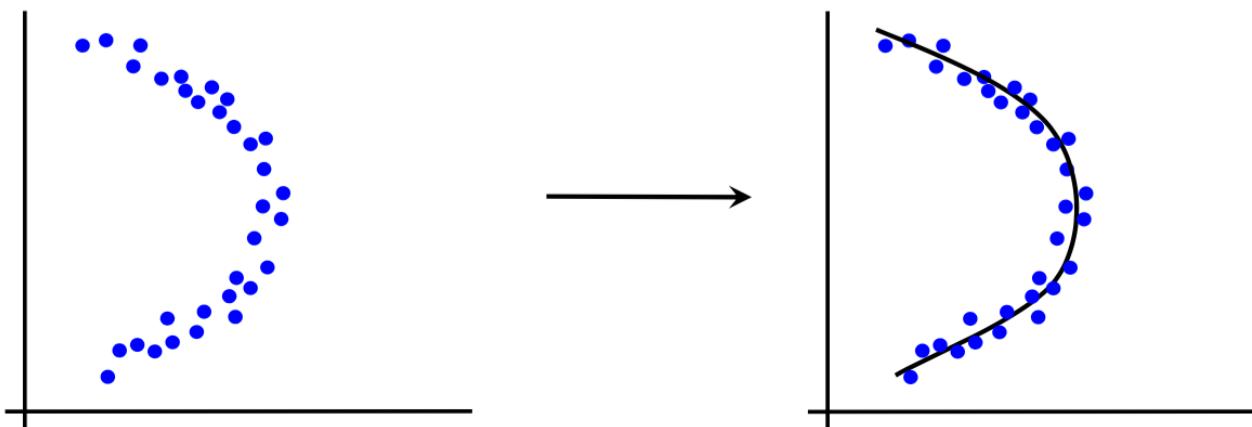
- Generarea arborelui filogenetic pe baza secvențelor ADN

# Formele canonice ale problemelor de învățare nesupervizată

- Grupare (clustering)



- Reducerea dimensiunii



# Modele de învățare nesupervizată

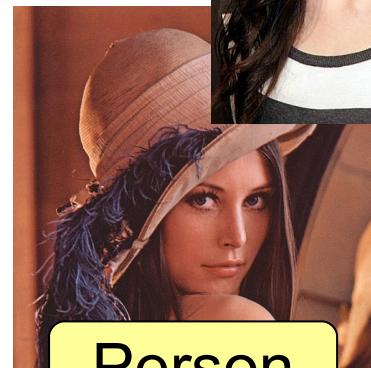
- K-means clustering (la master)
- Clustering ierarhic (la master)
- Analiza în componente principale (la master)
- Modele de tip auto-encoder (la master)
- Altele

# 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



Car



Person



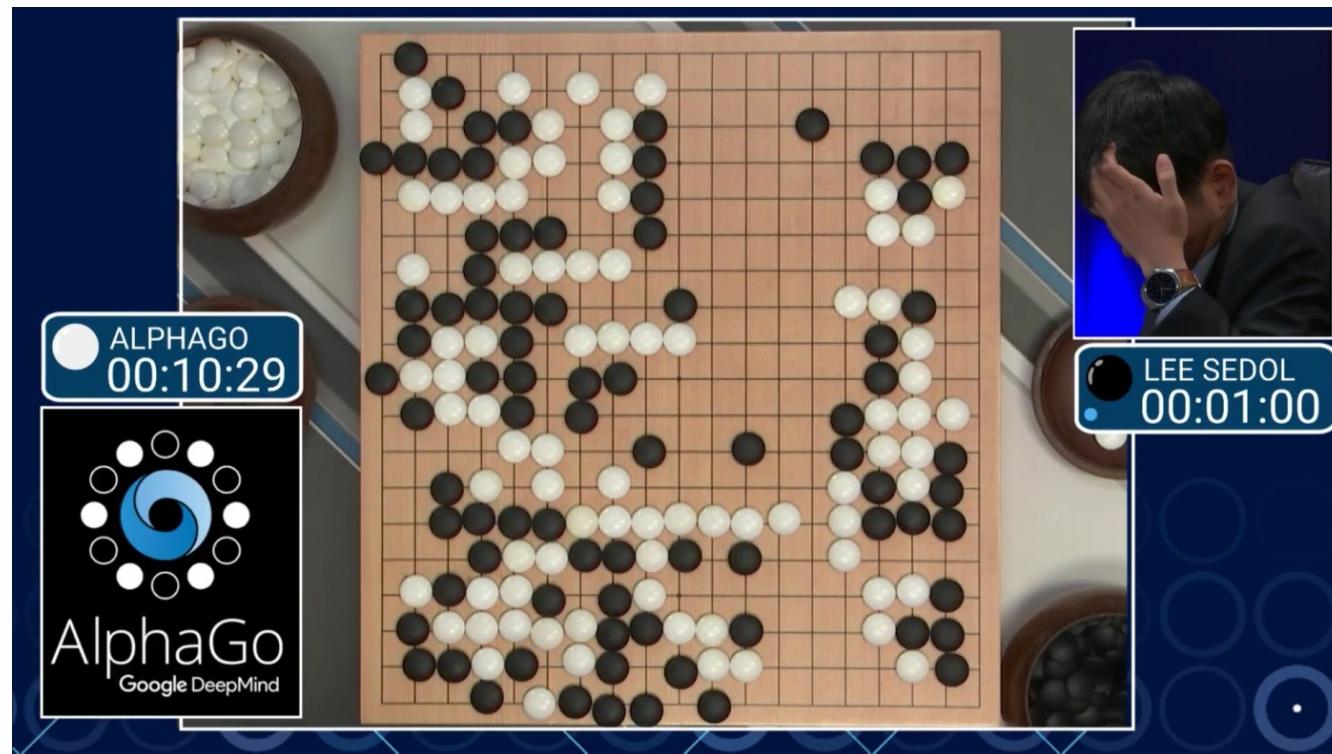
Dog

# Învățare ranforsată

- Cu ce diferă această paradigmă de învățare?
  - Sistemul învăță comportamentul intelligent 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 influențeză datele

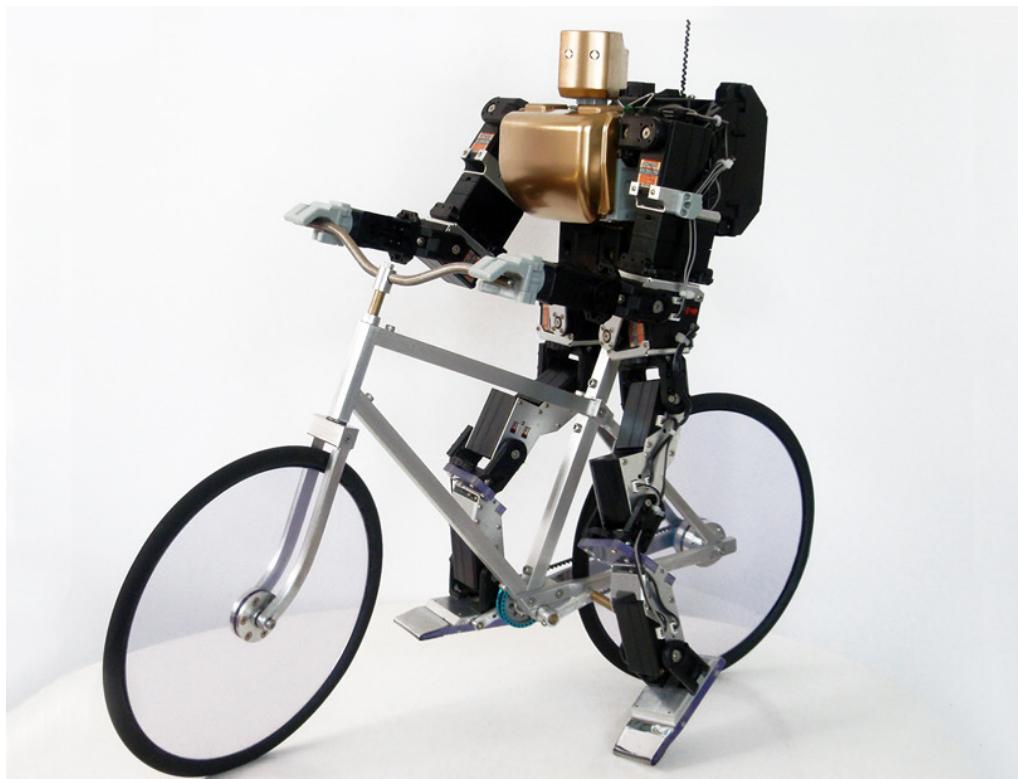
# Învățare ranforsată

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



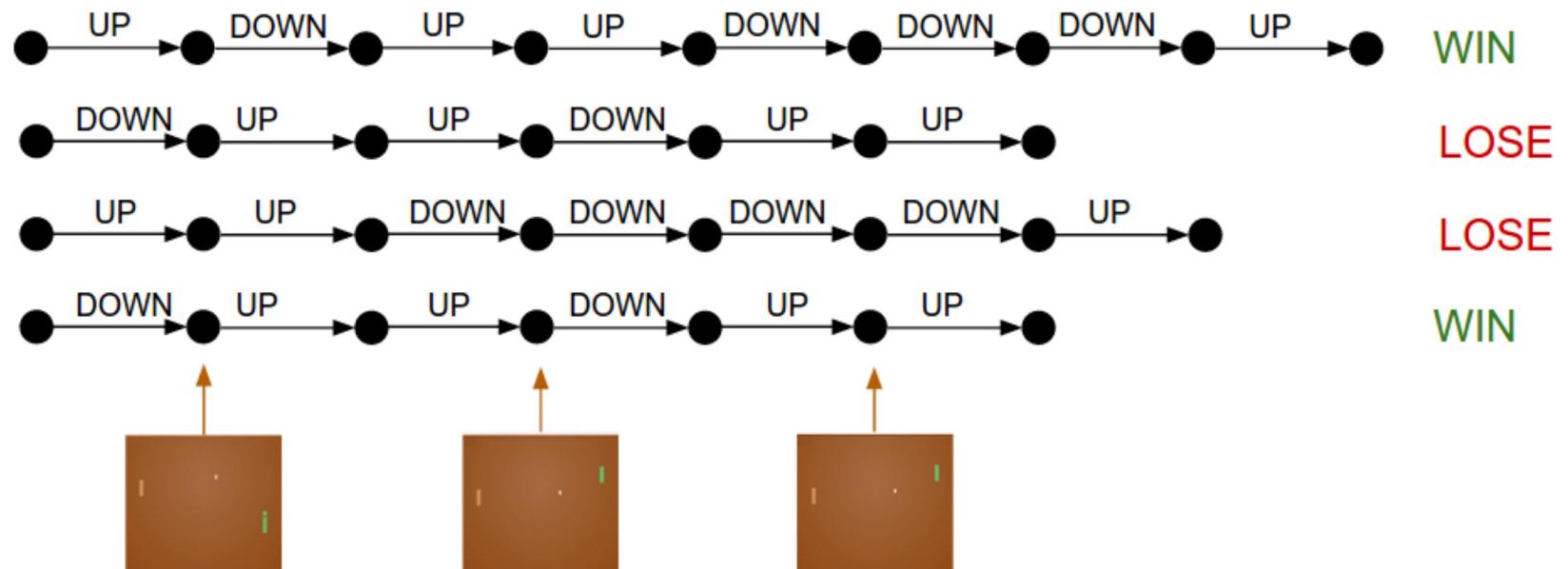
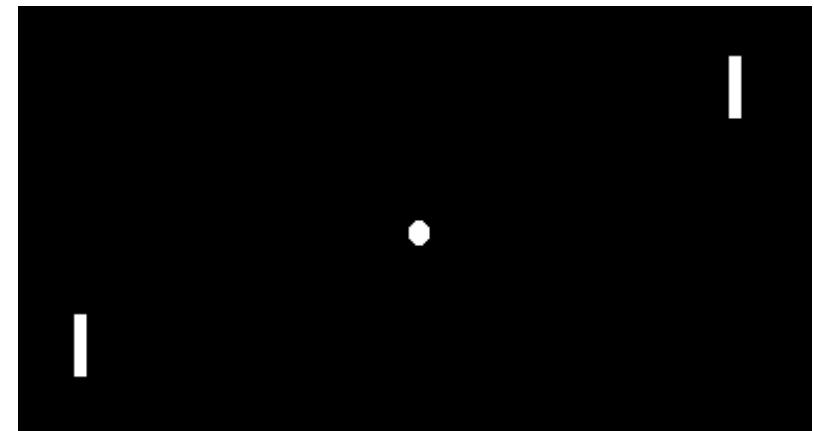
# Învățare ranforsată

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

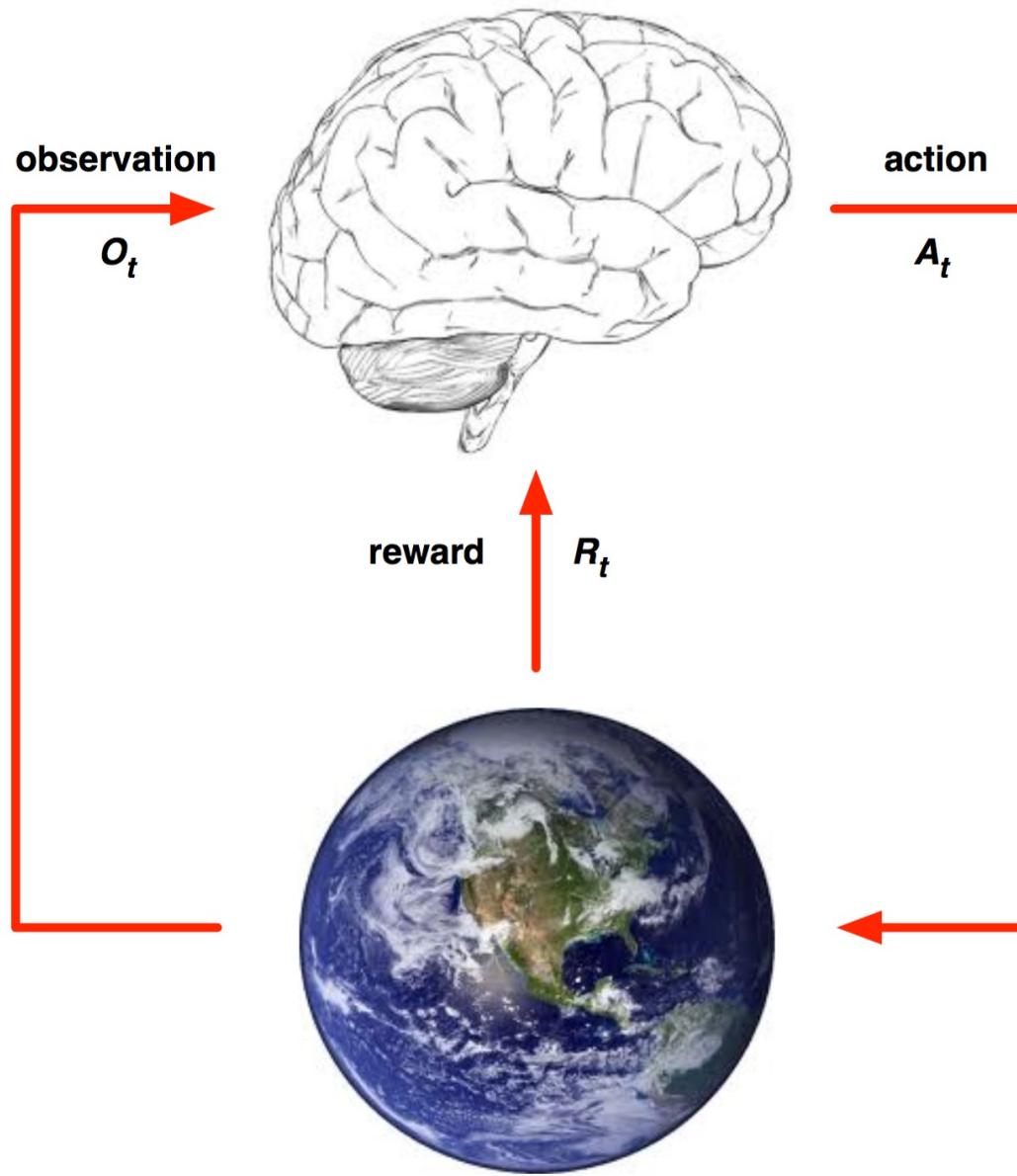


# Învățare ranforsată

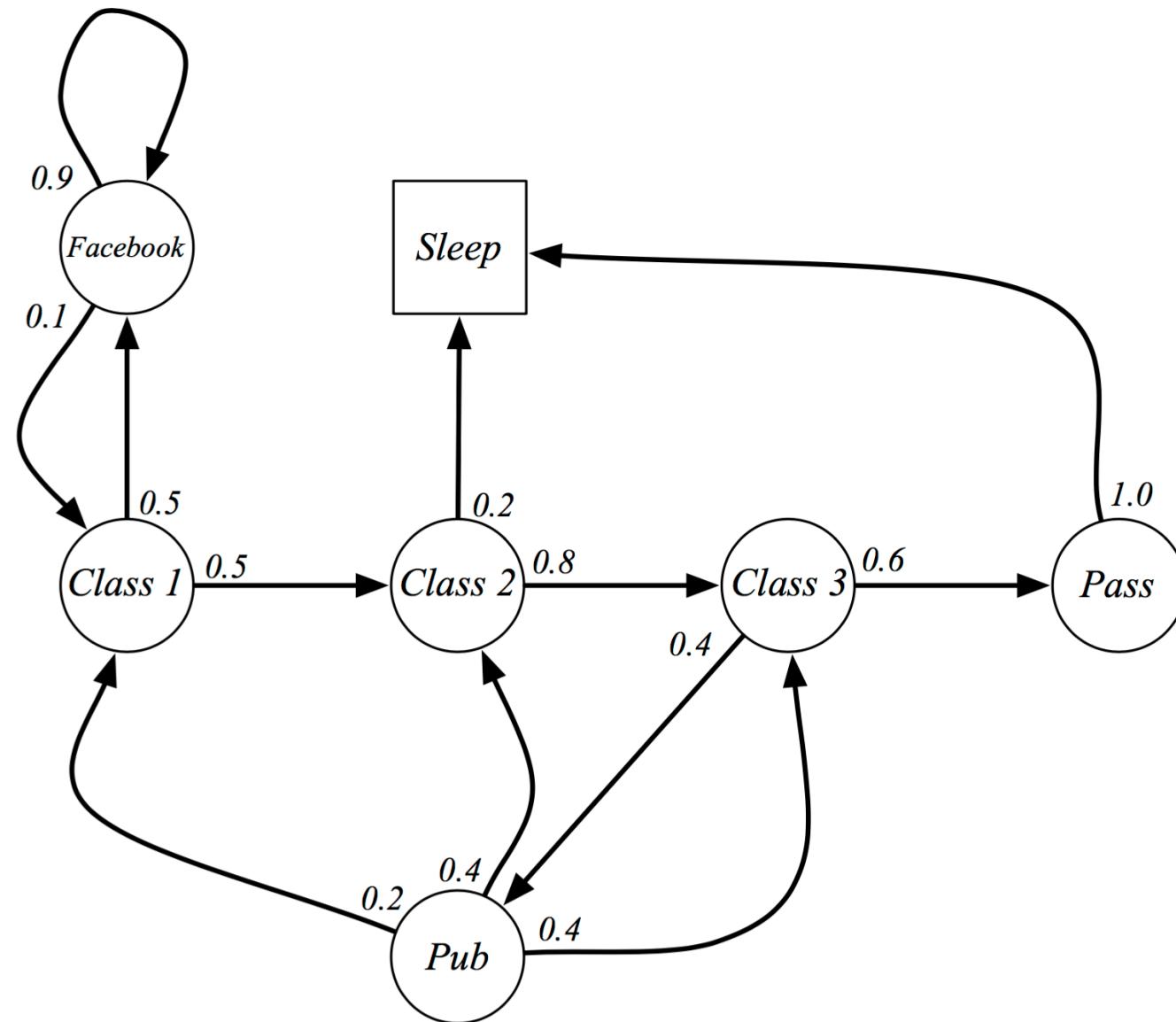
- Exemplu 3: Învățarea jocului Pong din pixeli
- recompensă +/- pentru creșterea scorului personal/al adversarului



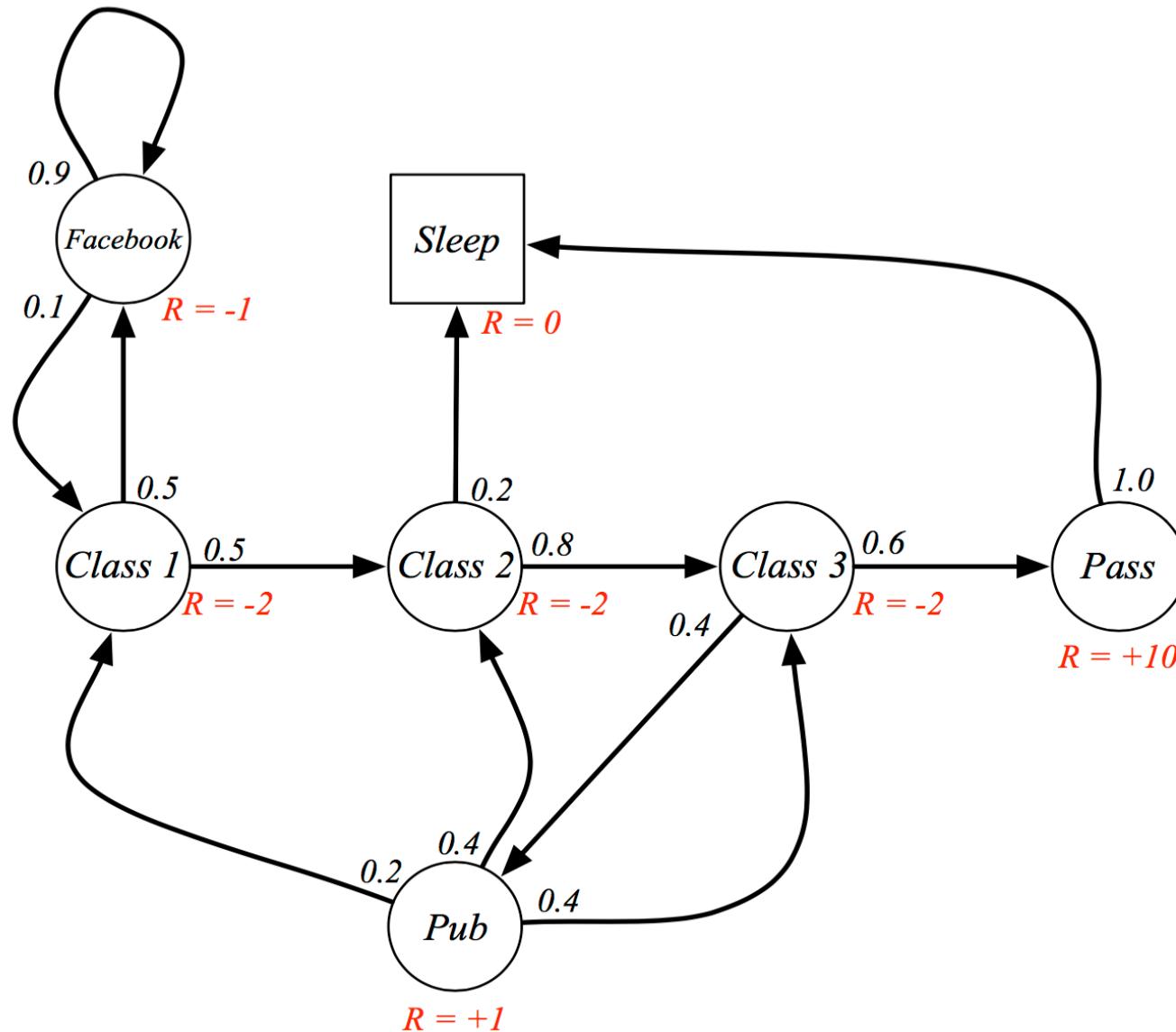
# Paradigma de învățare ranforsată



# Formalizarea cu Procese de Decizie Markov



# Formalizarea cu Procese de Decizie Markov



# Formalizarea cu Procese de Decizie Markov

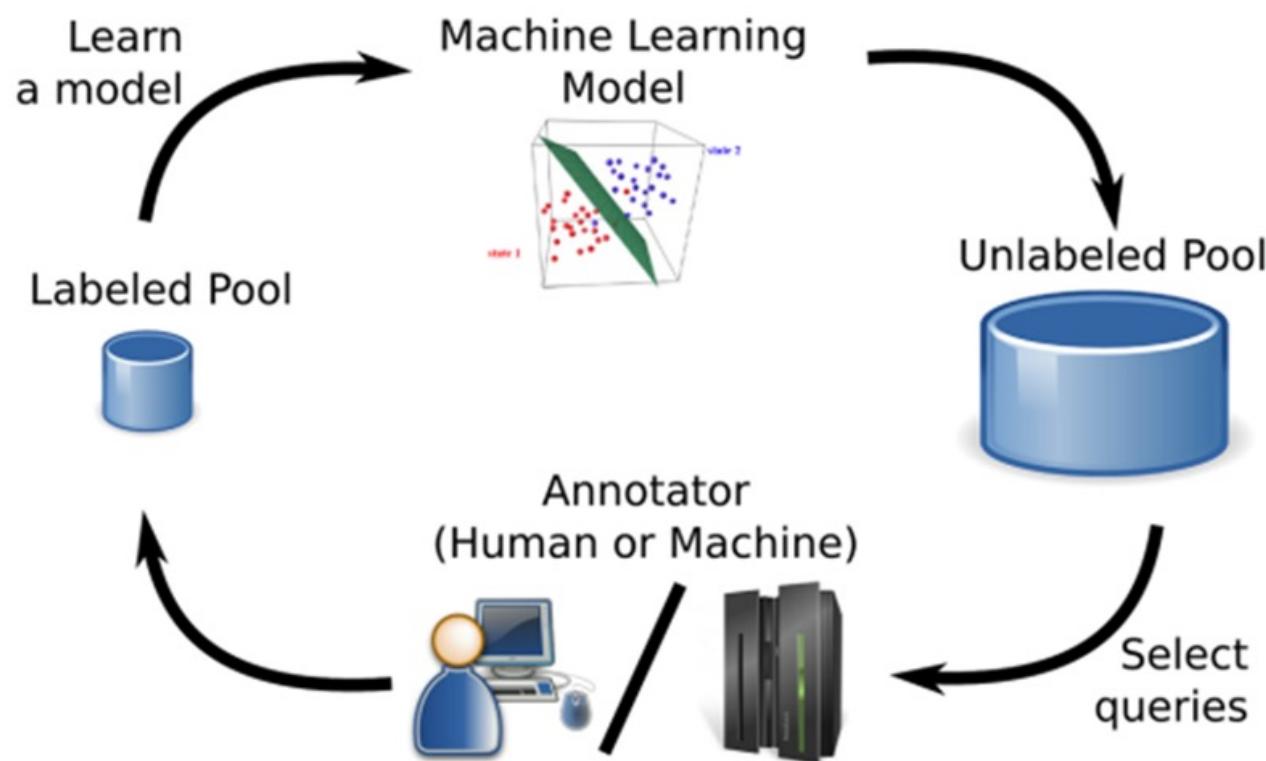
- Soluția bazată pe programare dinamică (grafuri mici) sau aproximare (grafuri mari)
- Scop: selectarea acțiunilor pentru a maximiza recompensa totală finală
- Acțiunile pot avea consecințe pe termen lung
- Sacrificarea unei recompense imediate poate conduce la câștiguri mai mari pe termen lung

# 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/>

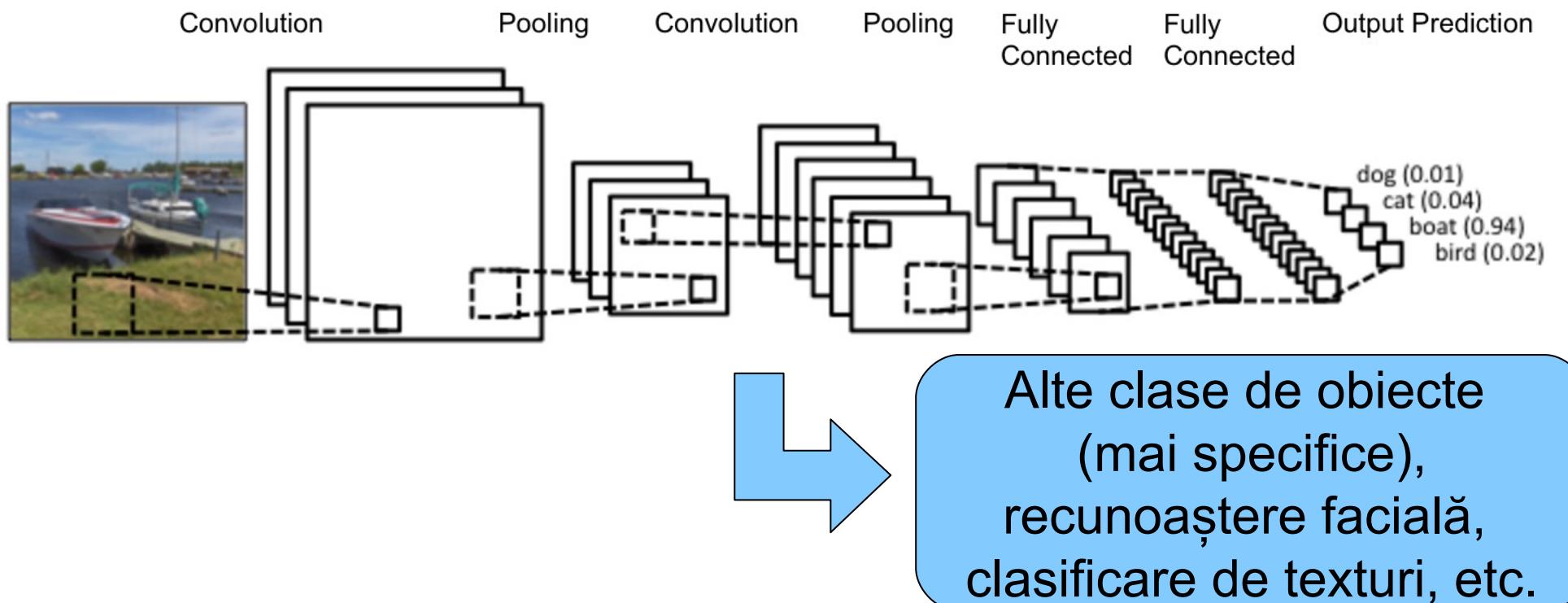
# Învățarea activă

- 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



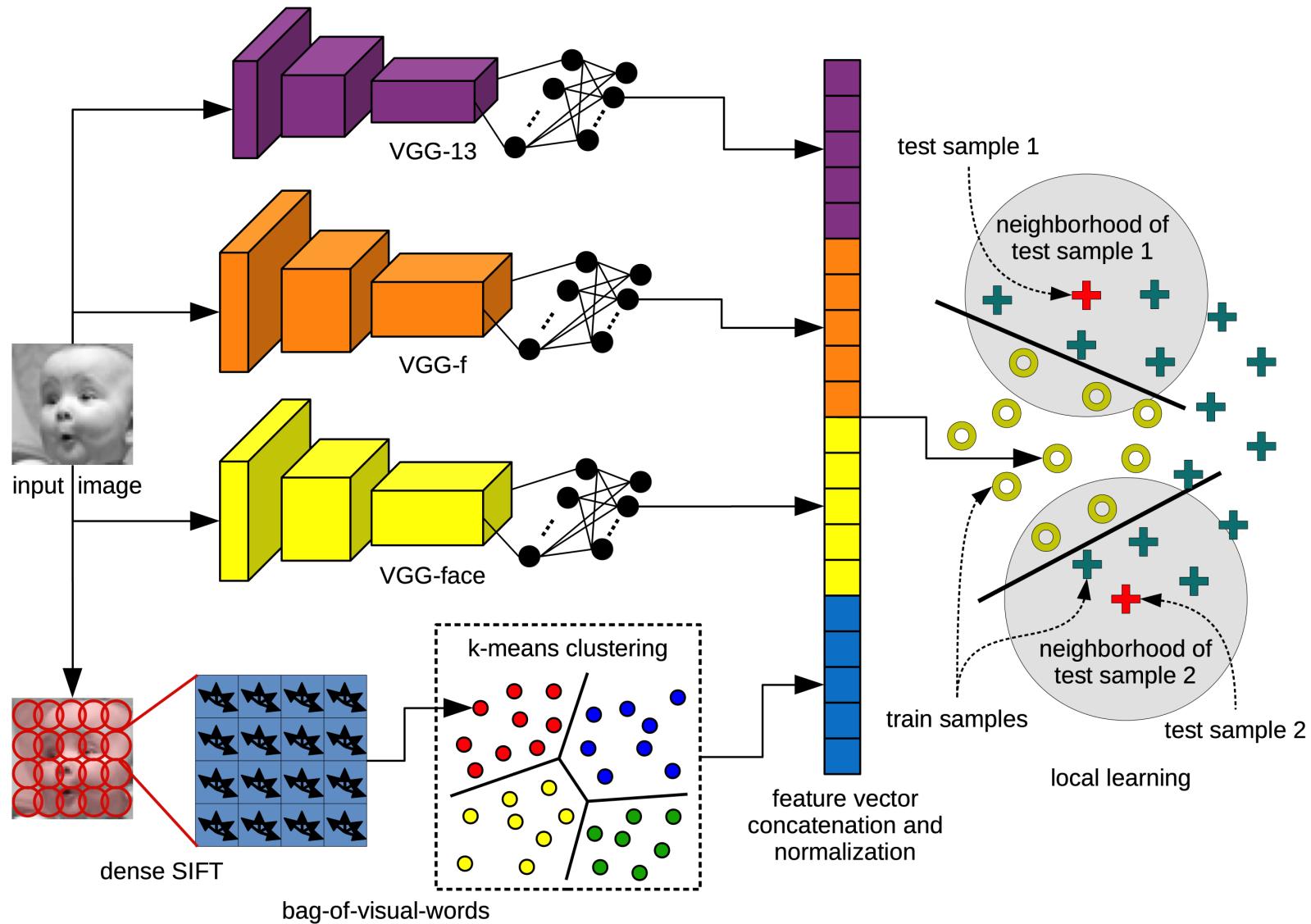
# Învățarea prin transfer

- 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 conoluționale



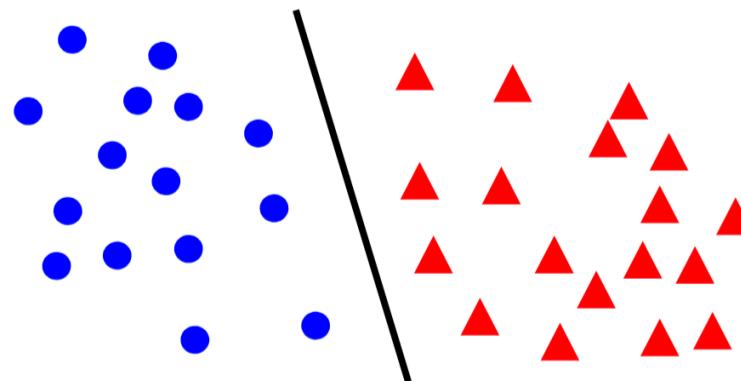
# Învățarea prin transfer

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

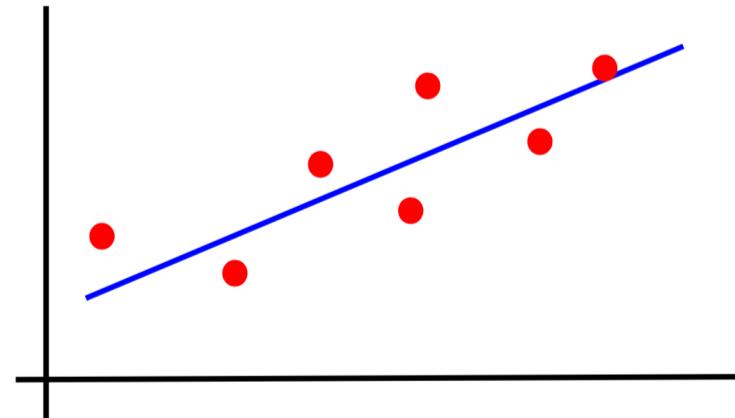


# Estimarea vârstei unei persoane din imagini

- Clasificare?



- Regresie?

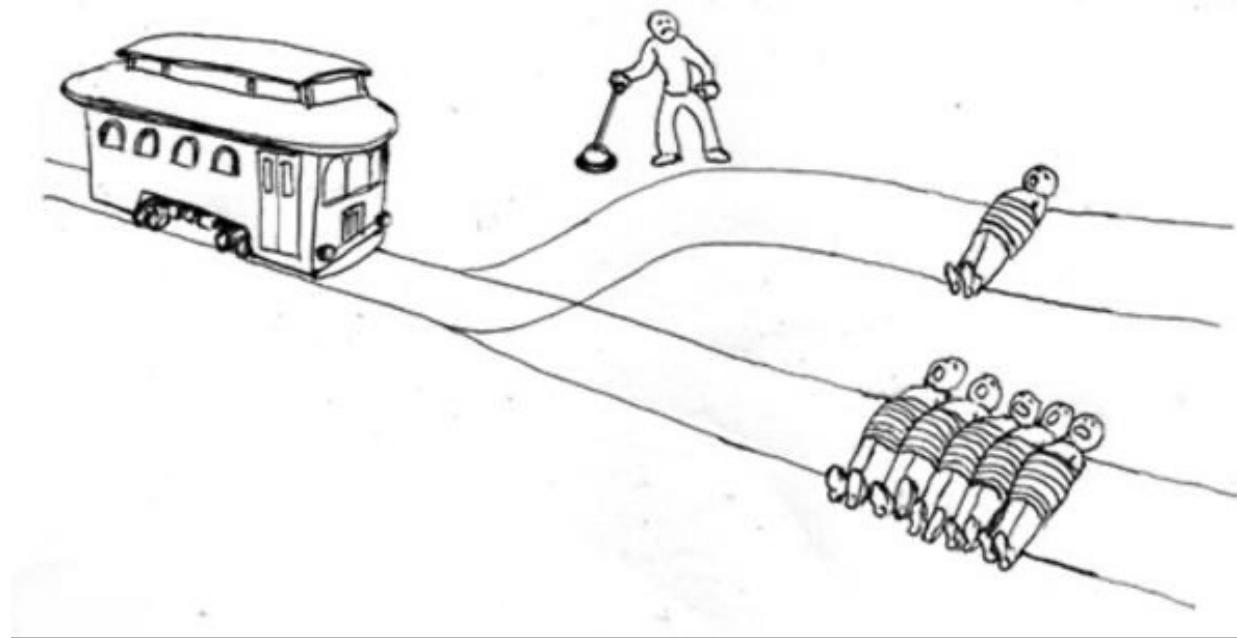


Ce vârstă?

- Alt tip de învățare?

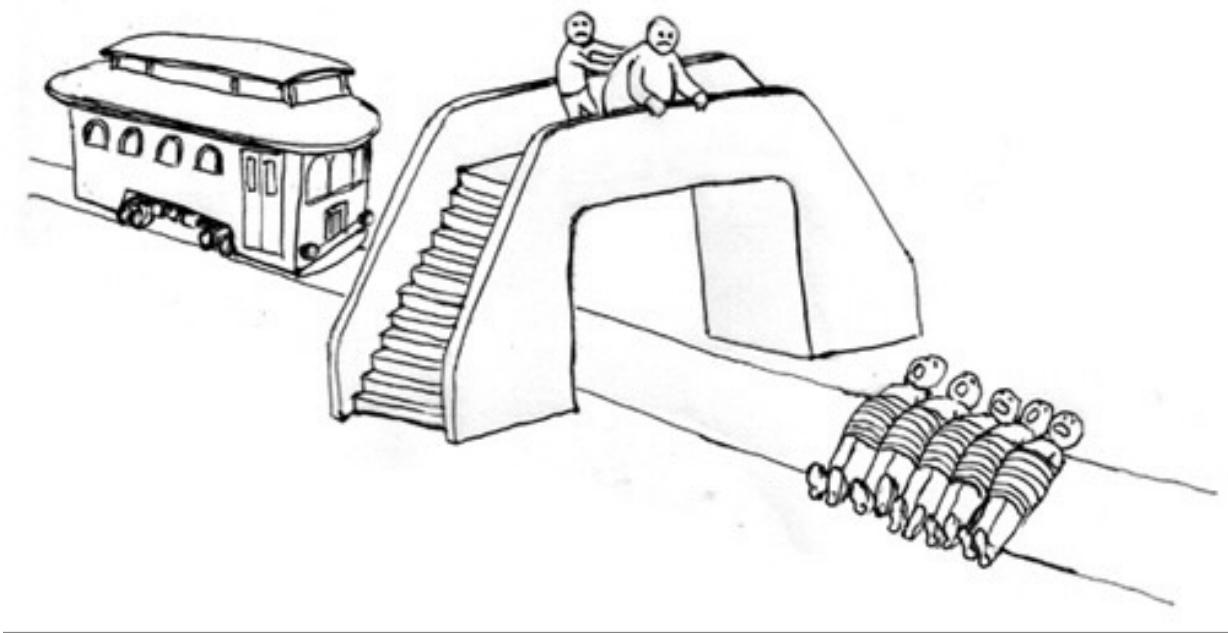
# Multe aplicații interesante, dar...

- 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

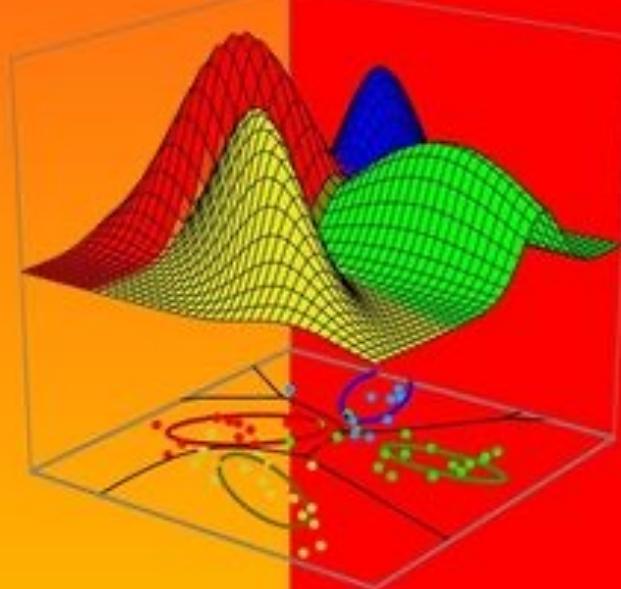
Data Mining, Inference, and Prediction

Second Edition



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



Radu Tudor Ionescu  
Marius Popescu

# Knowledge Transfer between Computer Vision and Text Mining

Similarity-based Learning Approaches

 Springer

# Concepte generale. Clasificatorul Bayes Naiv. Măsurarea performanței.

Prof. Dr. Radu Ionescu  
[raducu.ionescu@gmail.com](mailto:raducu.ionescu@gmail.com)

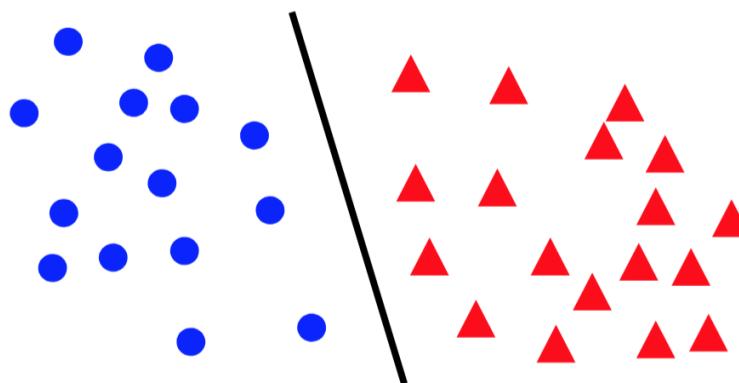
Facultatea de Matematică și Informatică  
Universitatea din București

# Paradigme ale învățării

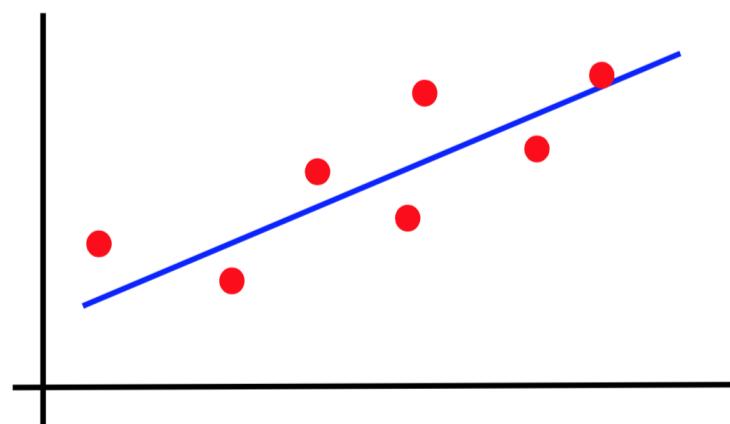
- Învățare supervizată (supervised learning)
- Învățare nesupervizată (unsupervised learning)
- Învățare semi-supervizată (semi-supervised learning)
- Învățare ranforsată (reinforcement learning)
- Paradigme non-standard:
  - Învățarea activă (active learning)
  - Învățare prin transfer (transfer learning)

# Formele canonice ale problemelor de învățare supervizată

- Clasificare



- Regresie



# Paradigma de învățare supervizată

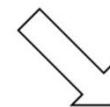
Functions  $\mathcal{F}$

$$\textcolor{red}{f} : \mathcal{X} \rightarrow \mathcal{Y}$$

Training data

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

LEARNING



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



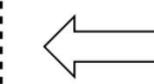
Learning machine

PREDICTION

$$\textcolor{red}{y} = \hat{f}(x)$$

New data

$x$



# Pașii necesari pentru învățare supervizată

- **Definirea** problemei de învățare supervizată
- **Colectarea datelor**

Pornim cu datele de antrenare, pentru care știm etichetele corecte (de la un profesor sau oracol)

- **Reprezentarea datelor**

Alegem cum să reprezentăm datele

- **Modelarea**

Alegerea spațiului de ipoteze:  $H = \{g: X \rightarrow Y\}$

- **Învățarea / Estimarea parametrilor**

Găsirea celei mai bune ipoteze din spațiul ales

- **Selectarea modelului**

Încercăm mai multe modele și îl păstrăm pe cel mai bun

- Dacă rezultatele sunt mulțumitoare atunci ne oprim

**Altfel rafinăm unul sau mai mulți pași anteriori**

# Clasificare între Banana și Furbish

- Date de antrenare
- Banana language:
  - baboi, bananonina, bello, hana, stupa
- Furbish:
  - doo, dah, toh, yoo, dah-boo, ee-tay
- Date de test: gelato
- Care este limba?
- De ce?
- Învățarea este grea fără a stabili un spațiu de ipoteze H!



# Antrenare versus testare

- Ce ne dorim?
- Performanță bună (pierdere scăzută) pe datele de antrenare?
- Nu, performanță bună pe datele de test (nevăzute)
- Date de antrenare:
- $\{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$
- Sunt date pentru a învăța funcția de mapare  $f$
- Date de testare:
- $\{x_1, x_2, \dots, x_M\}$
- Folosite pentru a vedea cât de bine am învățat

# Funcția de eroare / de pierdere

- Cum măsurăm performanța?
- Regresie:
  - Media pătratelor erorilor
  - Media erorilor în valoare absolută
- Clasificare:
  - Numărul de clasificări greșite (misclassification error)
  - Pentru clasificare binară:  
True Positive, False Positive, True Negative, False Negative
  - Pentru clasificare în mai multe clase:  
Matricea de confuzie

# Erori

- Eroarea de generalizare (generalization error):

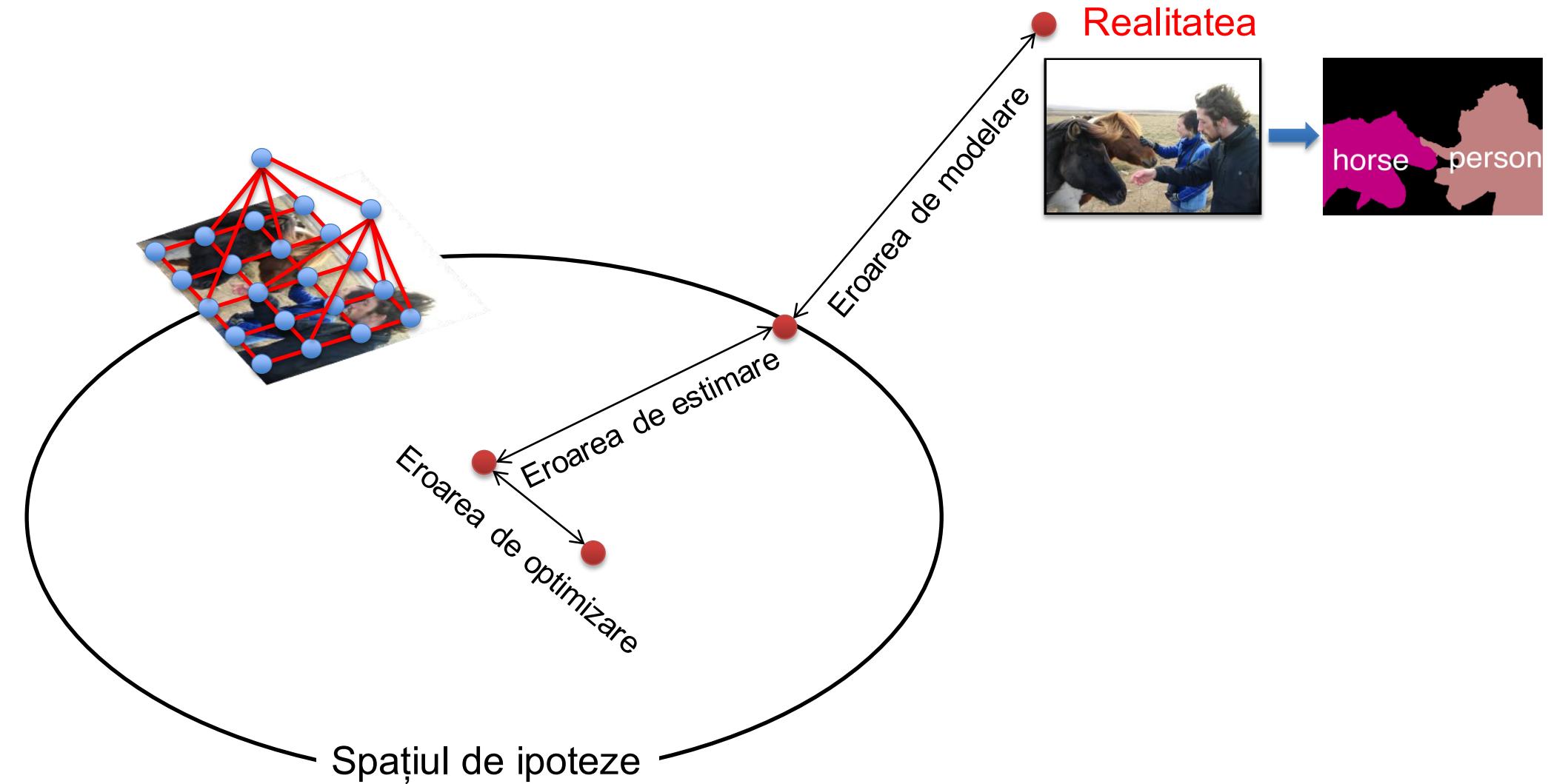
$$\varepsilon(h) = \int_{X \times Y} V(h(x), y) \rho(x, y) dx dy$$

- Probabilitatea comună  $\rho(x, y)$  este deobicei necunoscută
- Atunci calculăm eroare empirică (empirical error):

$$E(h) = \frac{1}{n} \sum_{i=1}^n V(h(x_i), y_i)$$

- Estimăm eroarea empirică pe datele de antrenare sau pe cele de test?
- Nu este corect să raportăm eroarea pe datele de antrenare!

# Descompunerea erorii

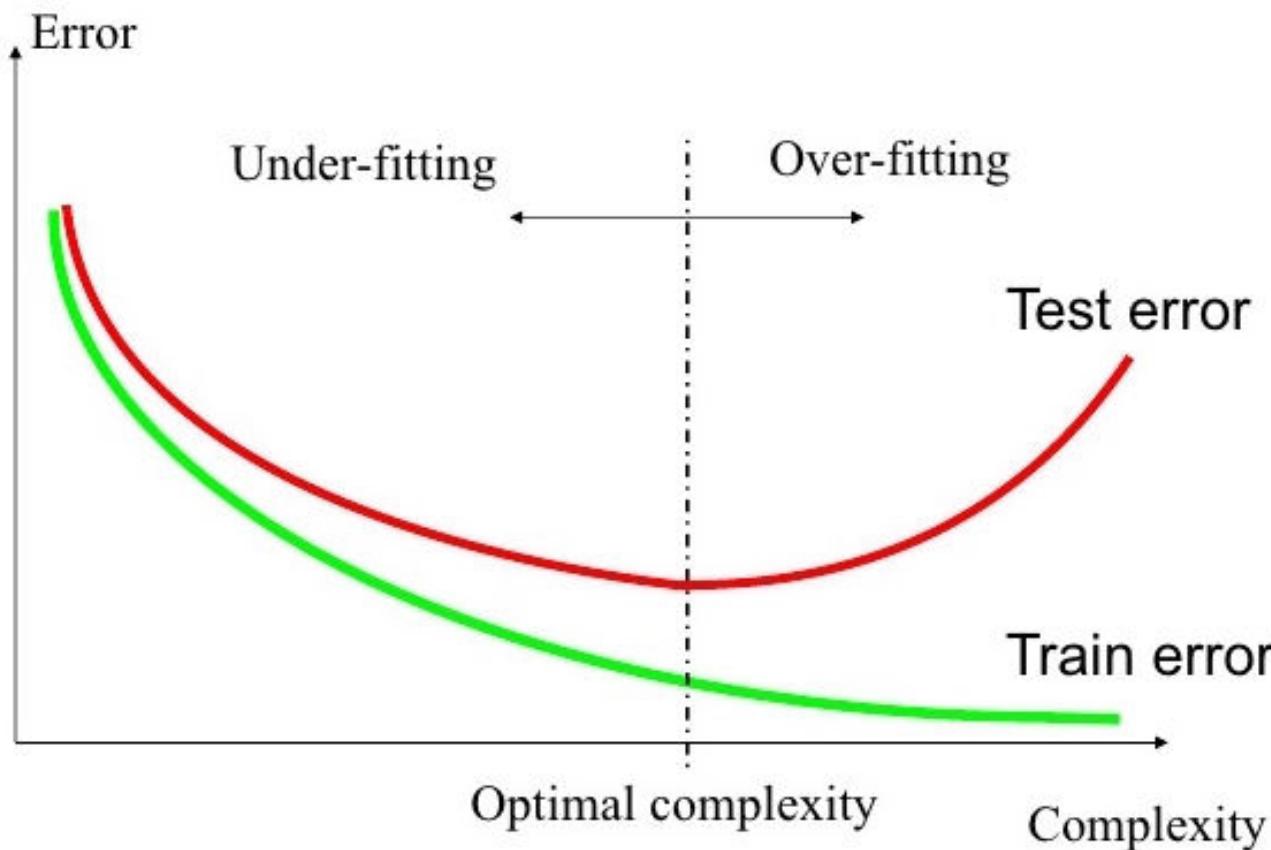


# Descompunerea erorii

- Eroare de modelare
  - Am încercat să modelăm realitatea cu un spațiu de ipoteze
- Eroarea de estimare
  - Am încercat să antrenăm un model cu o mulțime finite de date
- Eroarea de optimizare
  - Nu am reușit să optimizăm funcția până în punctul optim

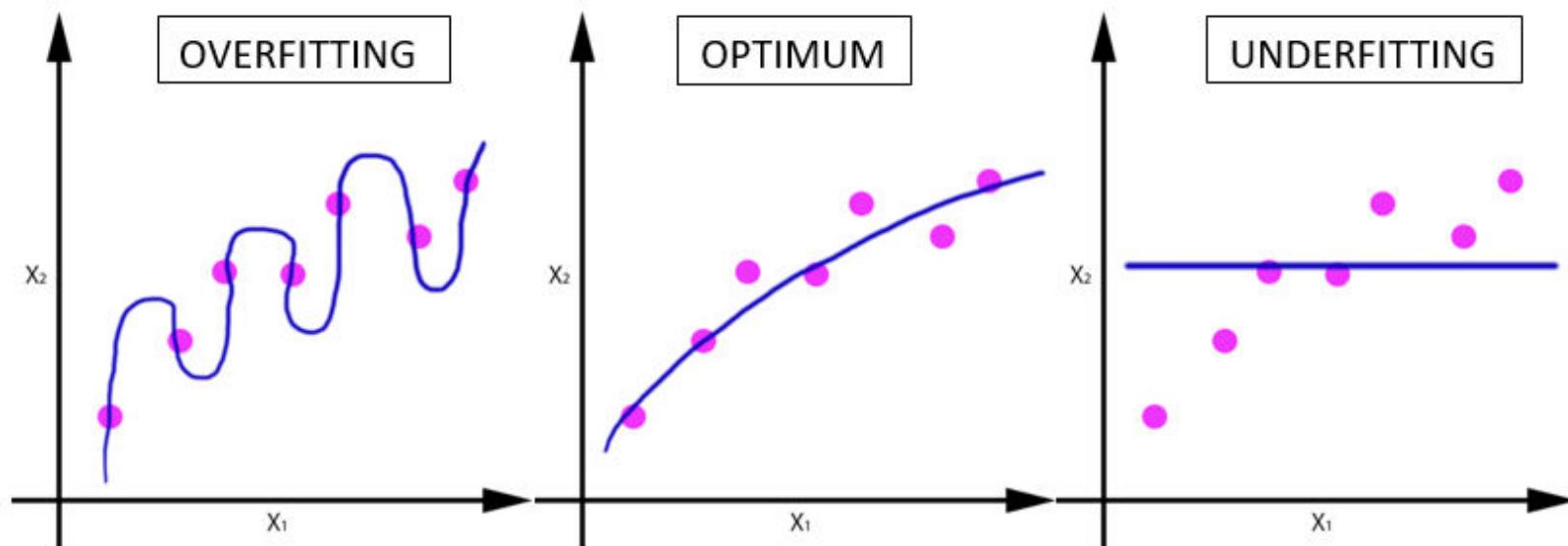
# Underfitting versus overfitting

- Problema cea mai importantă a învățării?
- Îmbunătățirea capacitatei de generalizare



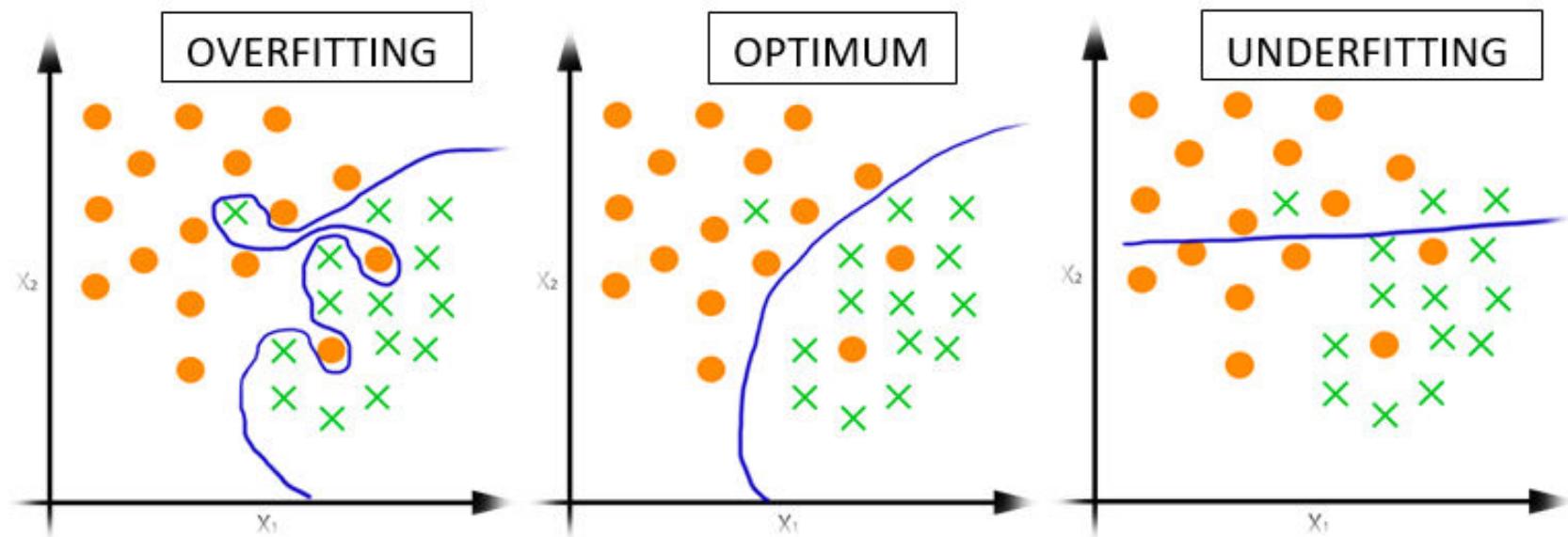
# Underfitting versus overfitting

- Exemplu 1: problemă de regresie



# Underfitting versus overfitting

- Exemplu 2: problemă de clasificare



# Bias-Variance Trade-off

- Bias
  - Eroare sistematică care provine din inabilitatea modelului de a învăța adevărata relație dintre trăsături și etichete (underfitting)
  - Poate fi corectată prin creșterea complexității modelului
- Variance
  - Eroare aleatoare care provine din sensibilitatea ridicată la mici fluctuații din date, cauzată de faptul că modelul a învățat și zgromotul din datele de antrenare (overfitting)
  - Poate fi corectată prin adăugarea de exemple de antrenare sau prin scăderea complexității modelului

# Bias-Variance Trade-off



# Bias-Variance Trade-off

Low Variance

Low Bias



High Bias



High Variance



# Abordarea procedurală

- Etapa de antrenare:
  - Date neprelucrate  $\rightarrow x$   
(extragerea trăsăturilor / caracteristicilor = feature extraction)
  - Date de antrenare  $\{(x,y)\} \rightarrow f$   
(învățare)
- Etapa de testare:
  - Date neprelucrate  $\rightarrow x$   
(extragerea trăsăturilor)
  - Date de testare  $x \rightarrow f(x)$   
(aplicarea funcției, calcularea erorii)

# Abordarea statistică

- Folosim probabilități:
  - x și y sunt variabile aleatoare
  - $D = (x_1, y_1), (x_2, y_2), \dots, (x_N, y_N) \sim P(X, Y)$
- Presupunem că datele sunt i.i.d. (independență și identic distribuite):
  - Datele de antrenare și testare sunt generate i.i.d. din  $P(X, Y)$
  - Învățăm pe setul de antrenare
  - Sperăm ca modelul să **generalizeze** pe datele de test

# Concepțe

- Capacitatea modelului
  - Cât de larg este spațiul de ipoteze  $H$ ?
  - Este sau nu restrâns spațiul de funcții?
- Supra-învățare (overfitting)
  - $f$  funcționează bine pe datele de antrenare
  - Dar foarte slab pe datele de testare
- Capacitatea de generalizare
  - Abilitatea de a obține eroare mică pe datele noi de test

# Garanții

- Simplificând 20 de ani de cercetare din Teoria Învățării...
- Dacă:
  - Avem suficiente date de antrenare D
  - Și spațiul de ipoteze  $H$  nu este foarte complex
- atunci **probabil** că modelul va avea capacitate de generalizare

# Probabilități (recapitulare)

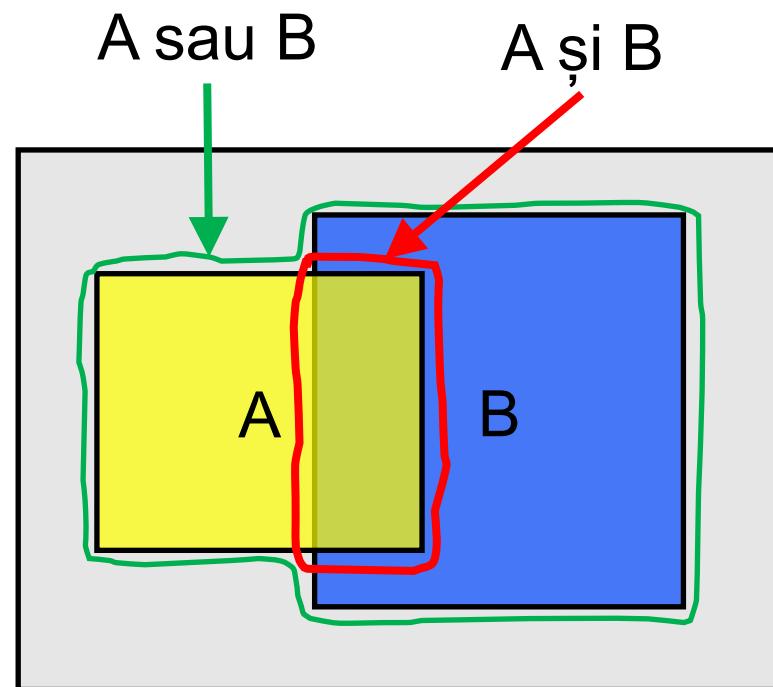
- A este un eveniment nedeterminist:  
 $A = \text{"Simona Halep va câștiga Roland Garros"}$
- Ce înseamnă  $P(A)$ ?
- Abordarea statistică:

$$\lim_{N \rightarrow \infty} \frac{\#(A = \text{true})}{N}$$

- Frecvența la limită a unui eveniment repetabil și nedeterminist
- Abordarea Bayesiană:
- $P(A)$  este ceea ce “credem” despre A
- Abordarea economică:
- $P(A)$  ne spune cât de mult “pariem” dacă alegem A

# Axiomele Probabilității (recapitulare)

- $0 \leq P(A) \leq 1$
- $P(\emptyset) = 0$
- $P(\mathbb{V}) = 1$
- $P(A \text{ sau } B) = P(A) + P(B) - P(A \text{ și } B)$



# Probabilități condiționate (recapitulare)

$$P(Y = y | X = x)$$

- Ce să credem despre  $Y = y$ , dacă știm că  $X = x$ ?
- $P(\text{Simona Halep va câștiga Roland Garros})$ ?
- Dacă știm următoarele:
  - În 2018, Simona Halep a câștigat Roland Garros
  - Simona Halep a pierdut două finale de Roland Garros
  - Simona se află pe poziția a treia în clasamentul WTA
  - În 2019, Ashleigh Barty (poziția întâi WTA) a câștigat Roland Garros

# Probabilități condiționate (recapitulare)

- $P(A | B) = \frac{\text{În cazurile în care } B \text{ este adevărat,}}{\text{proportia în care } A \text{ este adevărat}}$

- Exemplu:

➤ D: “Am dureri de cap”

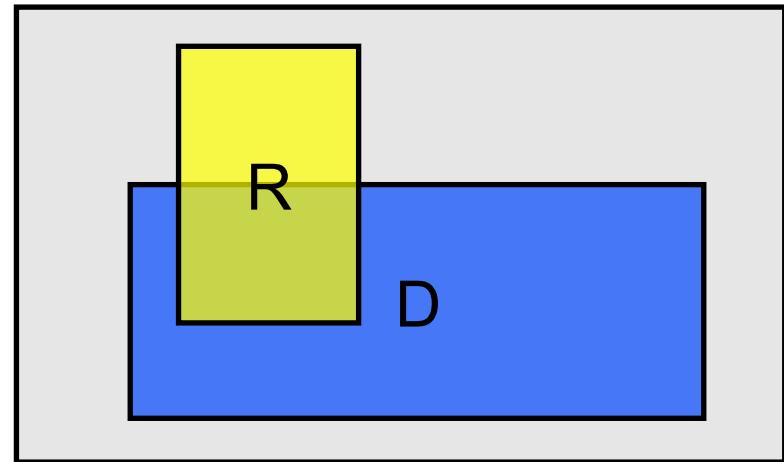
➤ R: “Sunt răcit”

- $P(D) = \frac{1}{10}$

- $P(R) = \frac{1}{40}$

- $P(D | R) = \frac{1}{2}$

- Durerile de cap sunt rare și răceala este și mai rară, dar dacă ești răcit atunci sunt 50% şanse să ai dureri de cap

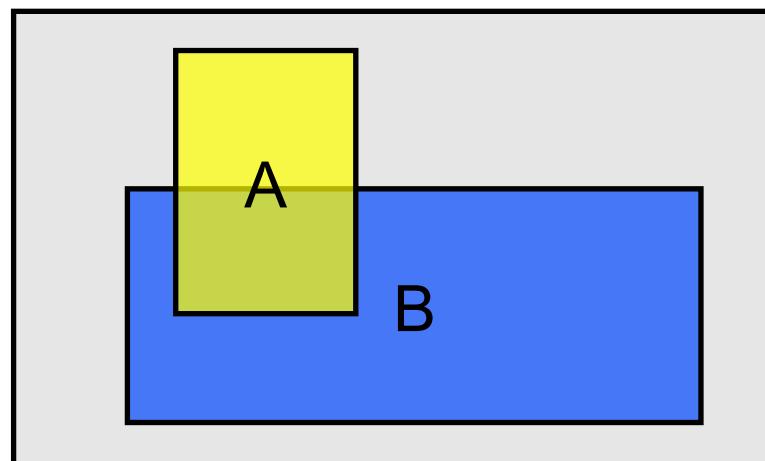


# Regula Bayes



$$P(B | A) = \frac{P(A \cap B)}{P(A)} = \frac{P(A | B) P(B)}{P(A)}$$

- Thomas Bayes "An Essay towards solving a Problem in the Doctrine of Chances" Royal Society, 1763.
- Simplu de înțeles dacă vă gândiți la arii

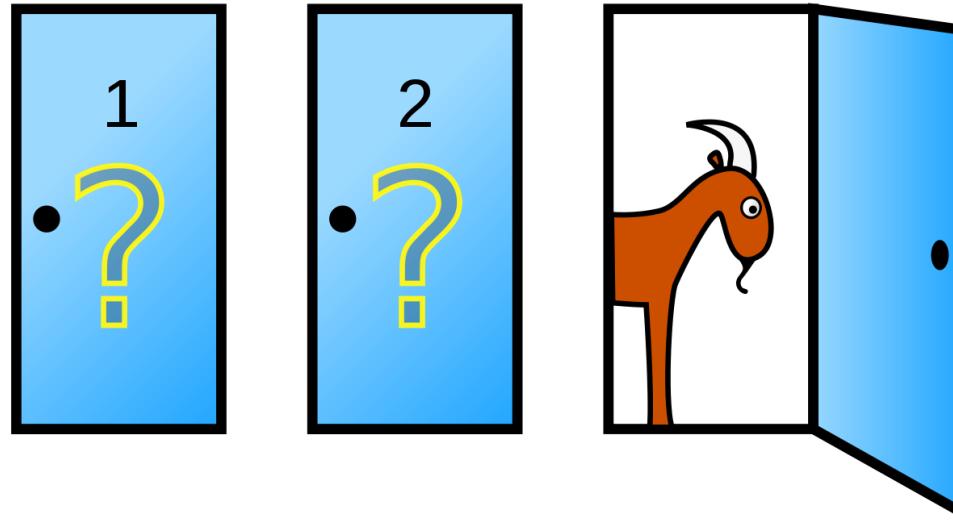


# Regula Bayes

Concepție:

- Probabilitate
  - Cât de bine explică datele o anumită ipoteză?
- Informații apriori
  - Ce credem înainte de a vedea datele?
- Informații aposteriori
  - Ce credem după ce vedem datele?

# Problema Monty Hall



- Sunt 3 uși numerotate cu 1, 2, 3.
- Un premiu mare (o mașină) este ascunsă în spatele unei uși. Celelalte două uși au câte o capră.
- Trebuie să alegem o ușă.
- Să presupunem că alegem poarta 1. Gazda deschide poarta 3, arătând capra din spate. Ce alegem mai departe?  
(a) Rămânem cu alegerea inițială (poarta 1);  
(b) Schimbăm și alegem poarta 2;  
(c) Este vreo diferență?

# Problema Monty Hall

- $H = i$  denotă ipoteza “premiul este după ușa  $i$ ”. Apriori toate cele 3 uși sunt egal probabile să ascundă premiul:

$$P(H = 1) = P(H = 2) = P(H = 3) = \frac{1}{3}$$

- Alegem poarta 1.
- Dacă premiul este în spatele ușii 1, gazda este indiferentă și va alege ușile 2 sau 3 cu probabilitate egală:

$$P(U = 2 | H = 1) = \frac{1}{2}, P(U = 3 | H = 1) = \frac{1}{2}$$

- Dacă premiul este în spatele ușii 2 (respectiv 3), gazda alege ușa 3 (respectiv 2):

$$P(U = 2 | H = 2) = 0, P(U = 3 | H = 2) = 1$$

$$P(U = 2 | H = 3) = 1, P(U = 3 | H = 3) = 0$$

- Gazda deschide poarta 3 ( $U=3$ ), descoperind capra. Observația este  $U=3$ . Premiul este în spatele ușii 1 sau 2?

# Problema Monty Hall

$$P(H = 1) = P(H = 2) = P(H = 3) = \frac{1}{3}$$

$$P(U = 2 | H = 1) = \frac{1}{2}, P(U = 3 | H = 1) = \frac{1}{2}$$

$$P(U = 2 | H = 2) = 0, P(U = 3 | H = 2) = 1$$

$$P(U = 2 | H = 3) = 1, P(U = 3 | H = 3) = 0$$

- Aplicăm regula Bayes:

$$P(H = 1 | U = 3) = \frac{P(U = 3 | H = 1) P(H = 1)}{P(U = 3)} = \frac{\frac{1}{2} \cdot \frac{1}{3}}{\frac{1}{2}} = \frac{1}{3}$$

$$P(H = 2 | U = 3) = \frac{P(U = 3 | H = 2) P(H = 2)}{P(U = 3)} = \frac{\frac{1}{2} \cdot \frac{1}{3}}{\frac{1}{2}} = \frac{2}{3}$$

# Clasificatorul optimal

- Învățăm:  $h: X \rightarrow Y$ 
  - $X$  – trăsături
  - $Y$  – etichete
- Presupunând cunoscută  $P(Y|X)$ , cum clasificăm datele?
  - Aplicăm clasificatorul Bayes:

$$y^* = h^*(x) = \operatorname{argmax}_y P(Y = y | X = x)$$

- **De ce?**

# Clasificatorul optimal

- **Teoremă:** Clasificatorul Bayes  $h_{Bayes}$  este optim!
  - Adică:
$$error_{true}(h_{Bayes}) \leq error_{true}(h), \forall h$$
- **Eroarea Bayes** este cea mai mică eroare posibilă:

$$error_{Bayes} = 1 - \sum_{y \neq y^*} \int_{x \in H_i} P(y | x)P(x)dx$$

# Clasificatorul optimal

- Cât de greu este să învățăm clasificatorul optimal?
  - Dar pentru date categorice?
- Cum reprezentăm datele? Câți parametrii trebuie estimati?
  - Probabilitatea apriori a claselor  $P(Y)$ :

Presupunem că  $Y$  este compus din  $k$  clase

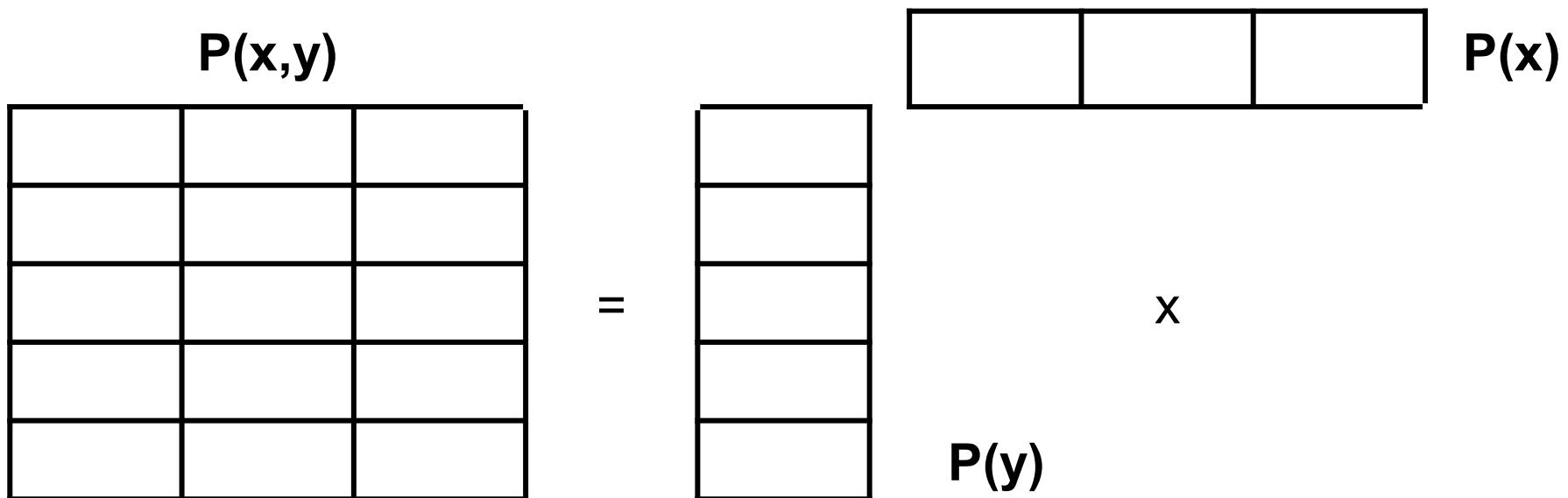
- Probabilitatea  $P(X | Y)$ :
- Presupunem că  $X$  este compus din  $n$  trăsături binare

**Model complex → Avem variantă mare cu date limitate!**

# Soluție: considerăm că trăsăturile sunt independente

- Două variabile sunt independente dacă și numai dacă:

$$P(x, y) = P(x) P(y)$$



- Două variabile sunt independente condiționat dacă, fiind dată o a treia variabilă, avem:

$$P(x, y | z) = P(x | z) P(y | z)$$

# Clasificatorul Naïve Bayes

- Presupunerea Naïve Bayes:

➤ Trăsăturile sunt independente:

$$P(X_1, X_2 | Y) = P(X_1 | Y)P(X_2 | Y)$$

➤ Mai general:

$$P(X_1 \dots X_n | Y) = \prod_i P(X_i | Y)$$

- Câți parametrii trebuie estimați acum?

➤ Presupunem că  $\mathbf{X}$  este compus din  $n$  trăsături binare

➤ Redus de la  $2^n$  la  $2 \cdot n$

# Clasificatorul Naïve Bayes

- Fiind date:
  - Probabilitatea apriori a claselor  $P(Y)$
  - $n$  trăsături independente  $\mathbf{X}$  condiționate de  $Y$
  - Pentru fiecare  $X_i$ , probabilitatea  $P(X_i | Y)$
- Regula de decizie Naïve Bayes este:

$$h_{NB}(x) = \operatorname{argmax}_y P(y) P(x_1, \dots, x_n | y)$$

$$h_{NB}(x) = \operatorname{argmax}_y P(y) \prod_i P(x_i | y)$$

- În practică folosim sumă de log!
- Dacă presupunerea este adevărată, NB este clasificatorul optimal!

# Estimarea parametrilor NB

- Se aplică metoda aproximării verosimilității maxime (Maximum Likelihood Estimation)
  - Fiind dat setul de antrenare, calculăm numărul de exemple pentru care  $A=a$  și  $B=b$ :

$\text{count}(A=a, B=b)$

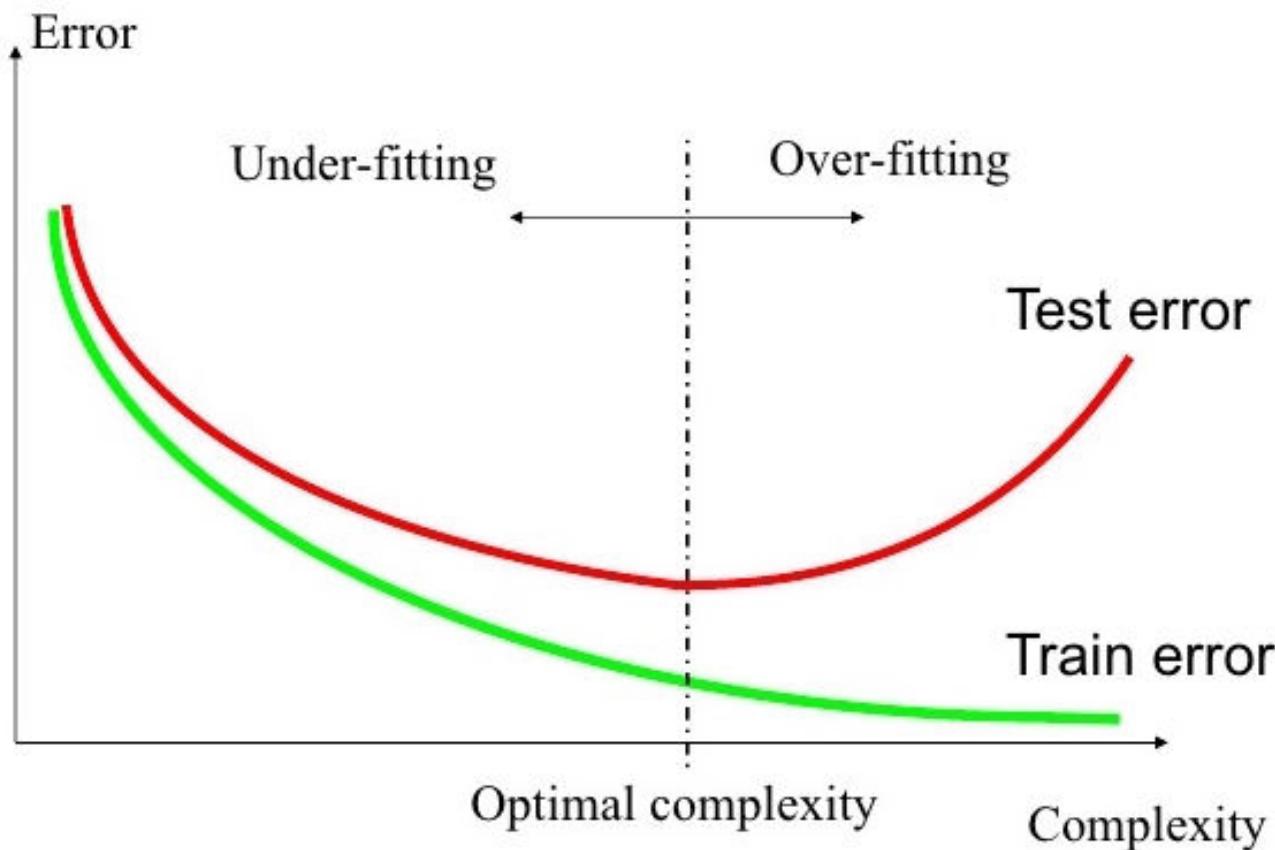
- Estimarea parametrilor:
  - Probabilitatea apriori a fiecărei clase:  $P(Y = y) = \dots$
  - Probabilitatea condiționată de clase:  $P(X_i = x_i | Y = y) = \dots$

# Încălcarea presupunerii NB

- Deobicei, trăsăturile nu sunt independente condiționat:
$$P(X_1 \dots X_n | Y) \neq \prod_i P(X_i | Y)$$
- Probabilitățile  $P(Y|X)$  sunt deseori 0 sau 1
- Totuși, clasificatorul NB este foarte popular
  - Deorece se descurcă bine, chiar dacă presupunerea este încălcată

# Underfitting versus overfitting

- Îmbunătățirea capacitatei de generalizare



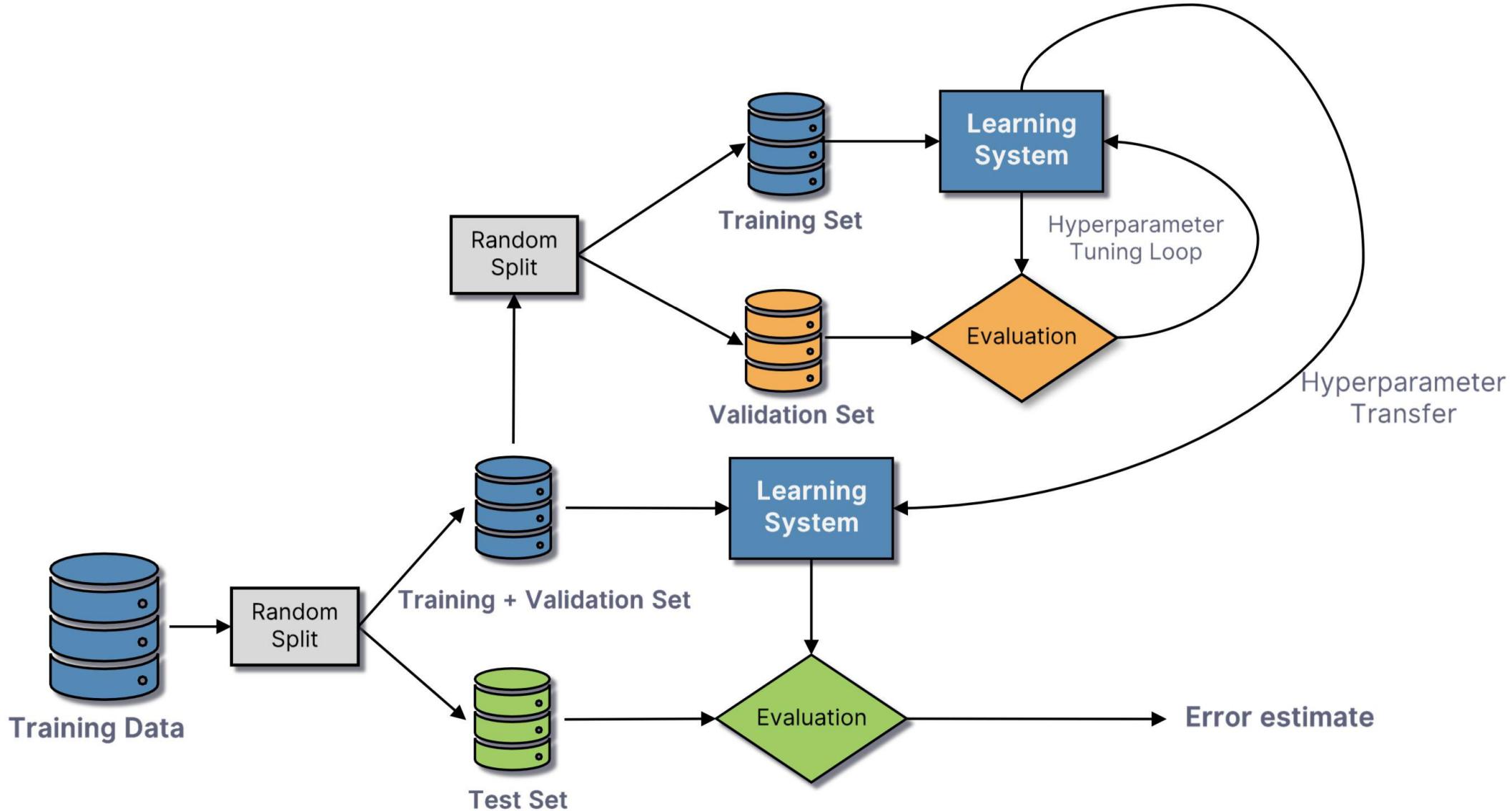
# Împărțirea datelor în date de antrenare, validare și test

- Pentru a construi un model cât mai performant, trebuie să îl testăm pe date “necunoscute”
  - O posibilă abordare (atunci când avem la dispoziție multe date):
    - 50% exemple pentru antrenare
    - 25% exemple pentru validare
    - 25% exemple pentru testare
- (procentele pot să varieze)

# De ce nu este suficient să împărțim datele în train și test?

- Utilizarea repetată a unei împărțiri atunci când încercăm diverse hiperparametrii poate să “uzeze” setul de test:
  - **Facem overfitting în spațiul hiperparametrilor!**
- Obținem o estimare mai bună a erorii dacă tunăm hiperparametrii pe un set diferit, anume setul de validare

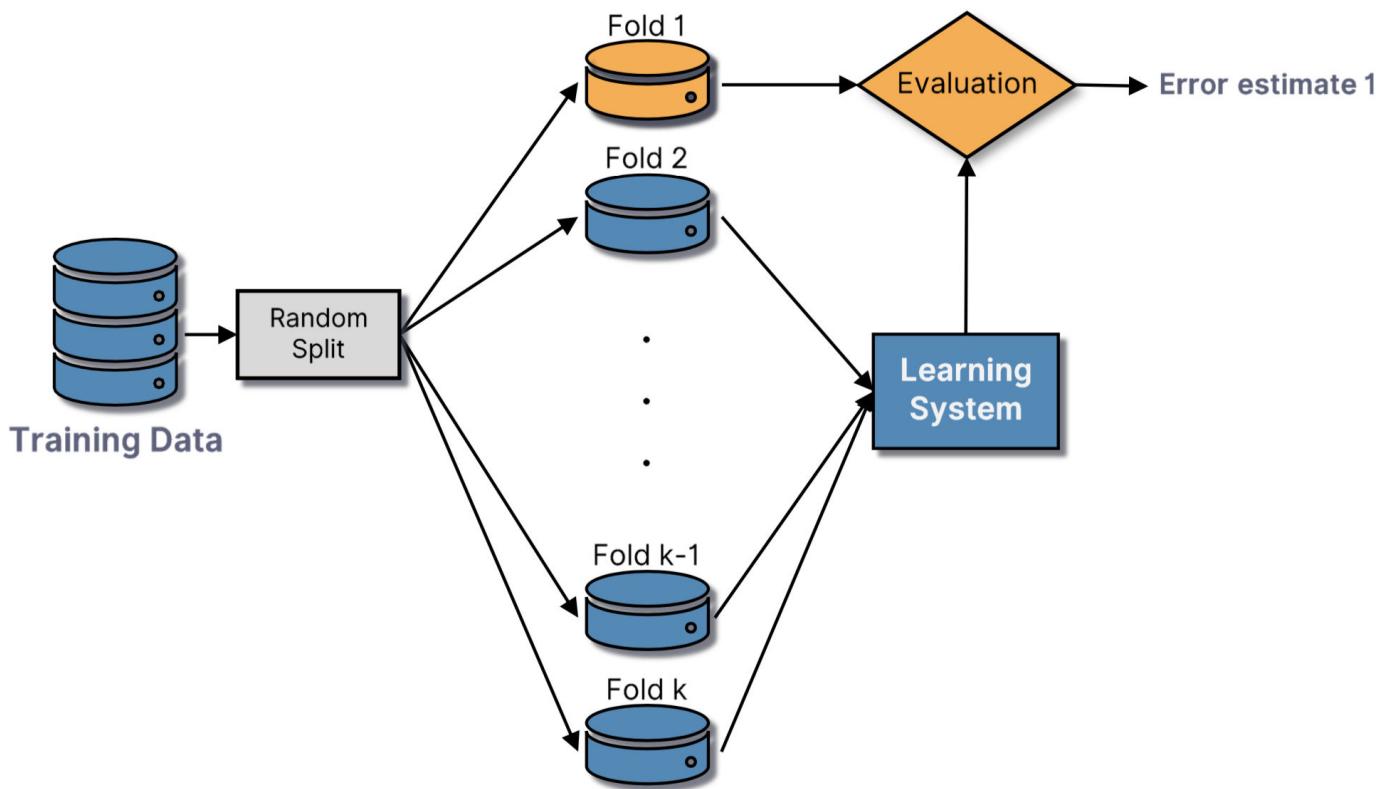
# Training, validation, test



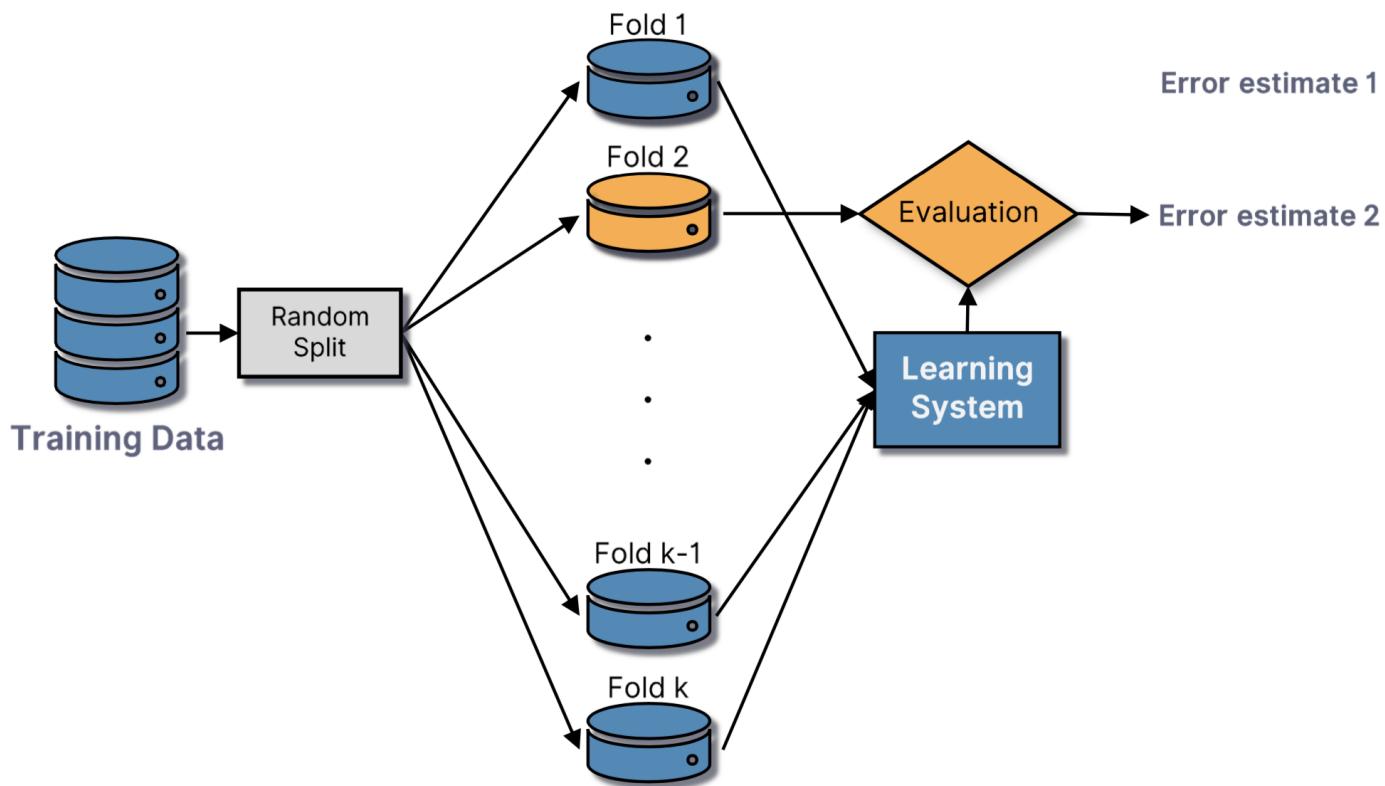
# Cross-validation

- O altă abordare (funcționează bine cu date puține):
  - Împărțim datelor în k părți egale (fold-uri)
  - Antrenăm pe  $k-1$  fold-uri și testăm pe fold-ul dat deoparte
  - Repetăm de  $k$  ori
  - Calculăm media rezultatelor
- Atunci când numărul de fold-uri este egal cu numărul de exemple:
  - Leave-one-out cross-validation

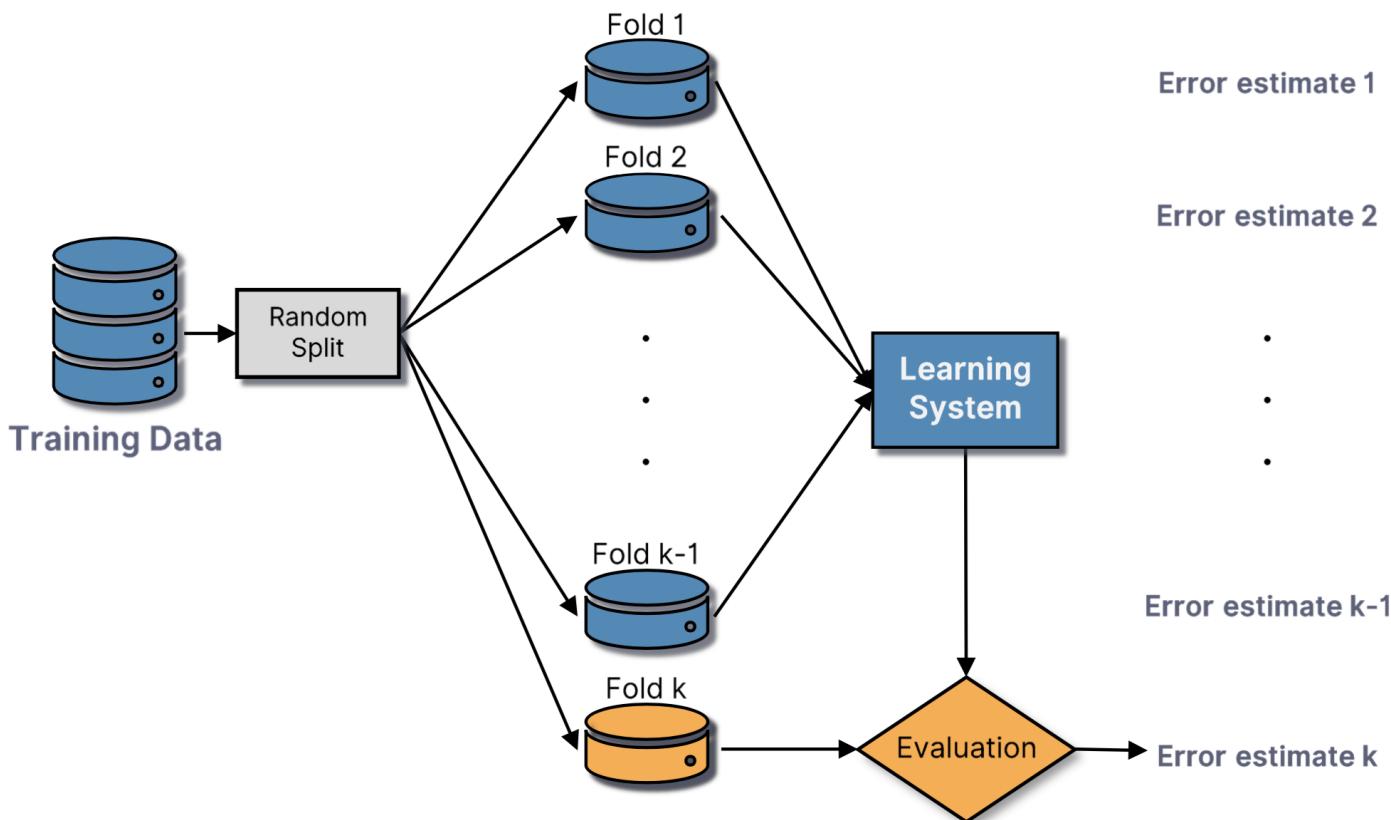
# Cross-validation



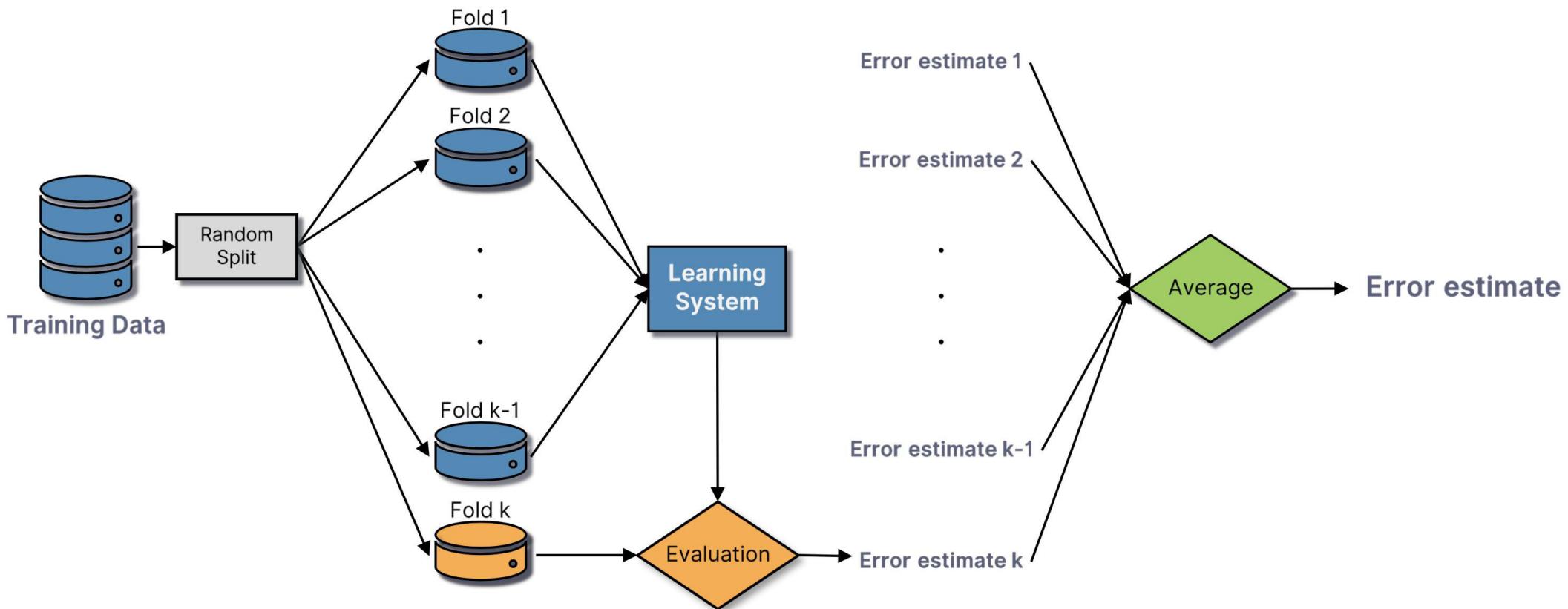
# Cross-validation



# Cross-validation



# Cross-validation



# Îmbunătățirea capacitatii de generalizare

## Early stopping

- Oprirea învățării atunci când observăm că eroarea pe validare începe să crească

## Regularizare

- Adăugarea unui termen care să penalizeze complexitatea funcției de învățare, impunând restricții de netezire sau limite asupra normei vectorului de ponderi

$$\min_f \sum_{i=1}^n V(f(\hat{x}_i), \hat{y}_i) + \lambda R(f)$$

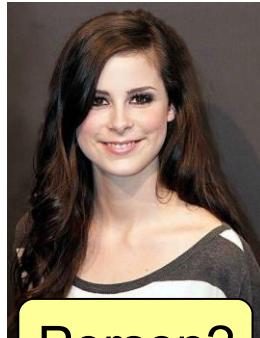
# Evaluare performanței

# Cum evaluăm un sistem de învățare automată?

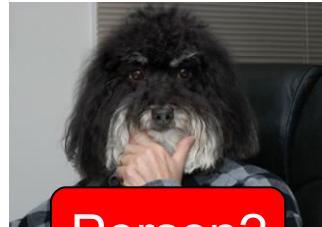
- Măsurăm acuratețea / eroarea pe datele de test:



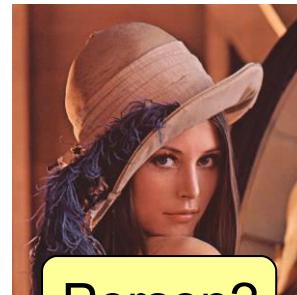
Car?



Person?



Person?



Person?



Dog?



Dog?

- Acuratețea: 4 corecte din 6 = 66.67%
- Eroarea: 2 greșite din 6 = 33.33%

# Cum evaluăm un sistem de învățare automată?

- Construim matricea de confuzie



- Acuratețea: suma elementelor de pe diagonală principală supra numărul de componente diferite de zero (4/6)
- Eroarea: suma elementelor rămase în afara diagonalei supra numărul de componente diferite de zero (2/6)

Predicted Actual	Car	Dog	Person
Car	1	1	0
Dog	0	1	1
Person	0	0	2

# Cum evaluăm un sistem de învățare automată?

- Matricea de confuzie în cazul binar



	Predicted YES	Predicted NO
Actual YES	True Positive	False Negative
Actual NO	False Positive	True Negative

# Cum evaluăm un sistem de învățare automată?

- Matricea de confuzie în cazul binar



	Predicted YES	Predicted NO
Actual YES	2	False Negative
Actual NO	False Positive	True Negative

# Cum evaluăm un sistem de învățare automată?

- Matricea de confuzie în cazul binar



	Predicted YES	Predicted NO
Actual YES	2	1
Actual NO	False Positive	True Negative

# Cum evaluăm un sistem de învățare automată?

- Matricea de confuzie în cazul binar



	Predicted YES	Predicted NO
Actual YES	2	1
Actual NO	1	True Negative

# Cum evaluăm un sistem de învățare automată?

- Matricea de confuzie în cazul binar



	Predicted YES	Predicted NO
Actual YES	2	1
Actual NO	1	2

# Cum evaluăm un sistem de învățare automată?

- Calculul măsurilor Precision și Recall



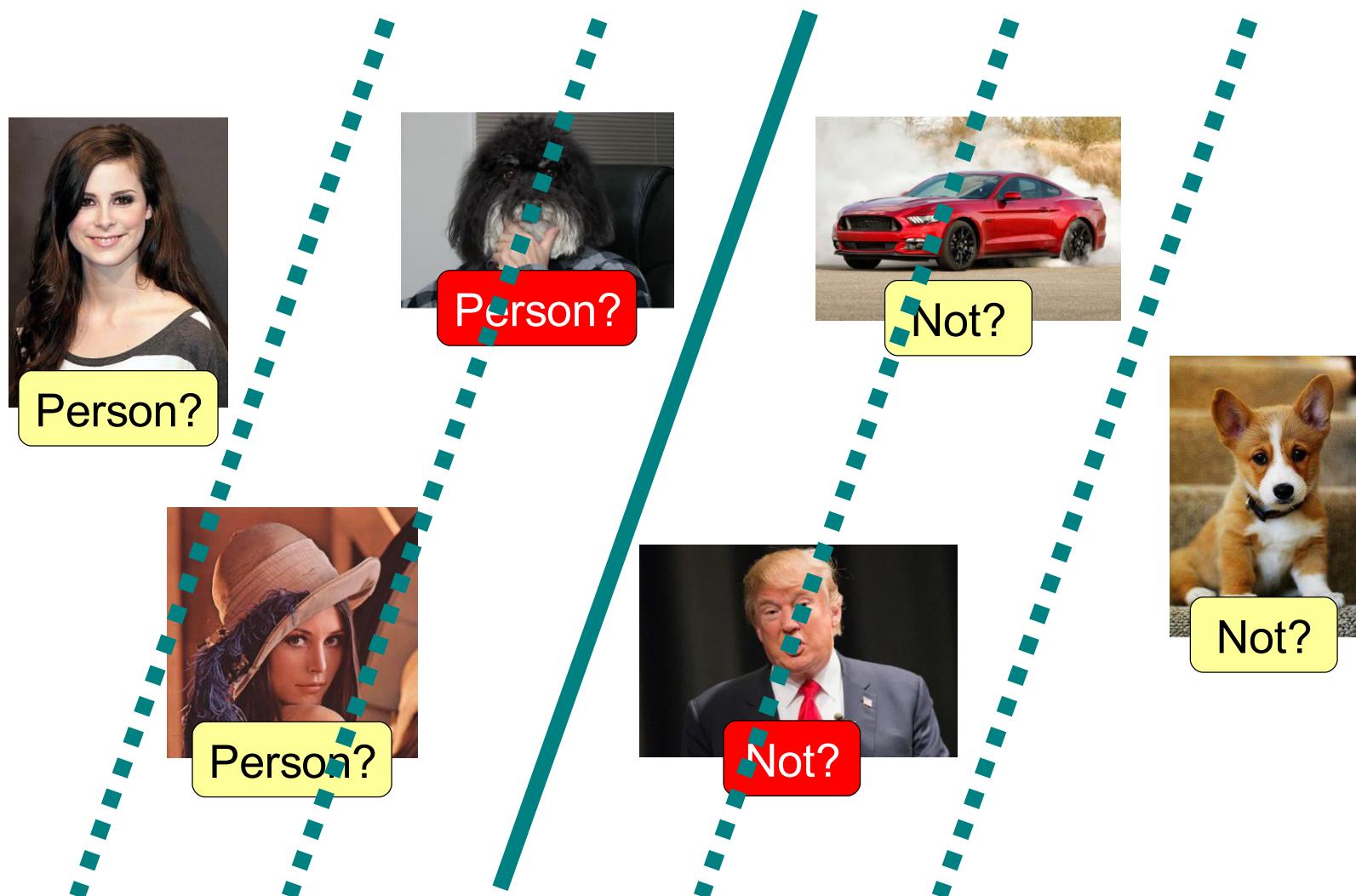
- $\text{Precision} = \text{TP} / (\text{TP} + \text{FP})$   
 $= 66.67\%$

- $\text{Recall} = \text{TP} / (\text{TP} + \text{FN})$   
 $= 66.67\%$

	Predicted YES	Predicted NO
Actual YES	2	1
Actual NO	1	2

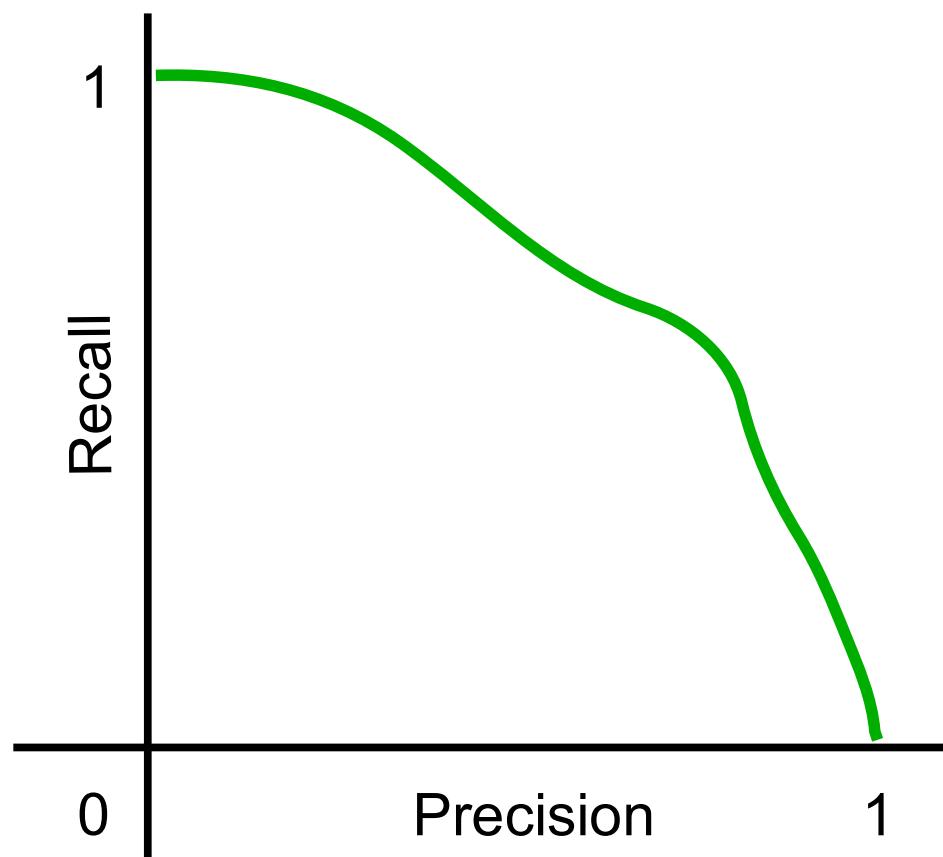
# Cum evaluăm un sistem de învățare automată?

- Curba Precision-Recall



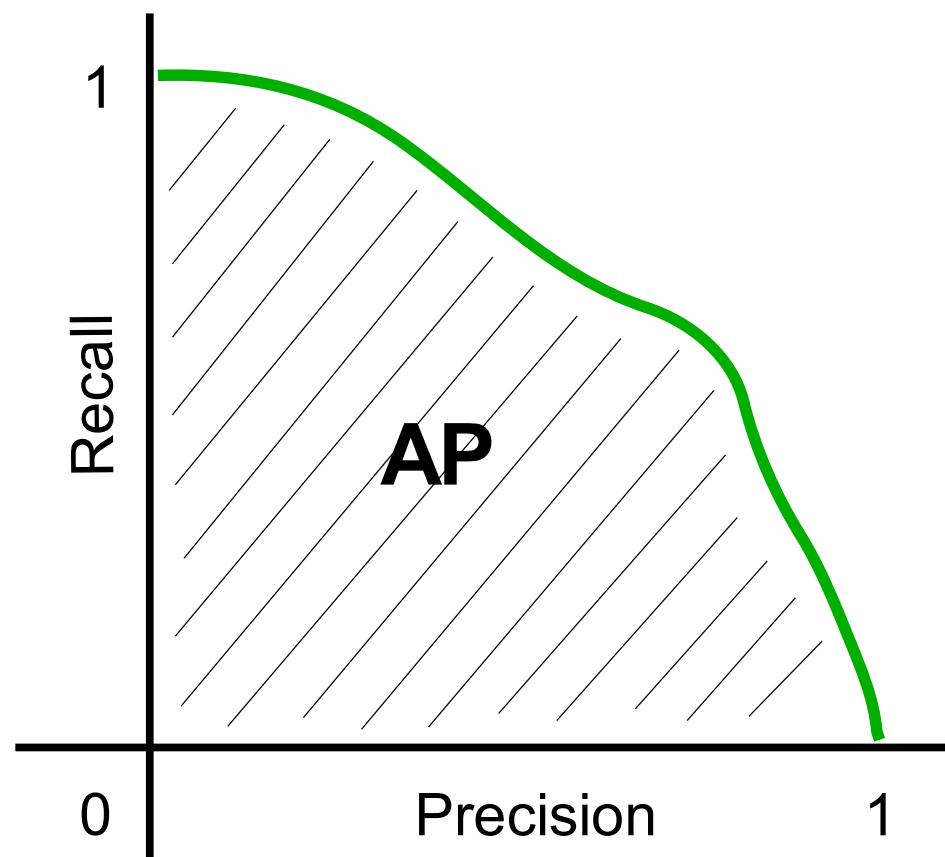
# Cum evaluăm un sistem de învățare automată?

- Curba Precision-Recall



# Cum evaluăm un sistem de învățare automată?

- Average Precision



# Cum evaluăm un sistem de învățare automată?

- Calculul măsurilor TPR și FPR

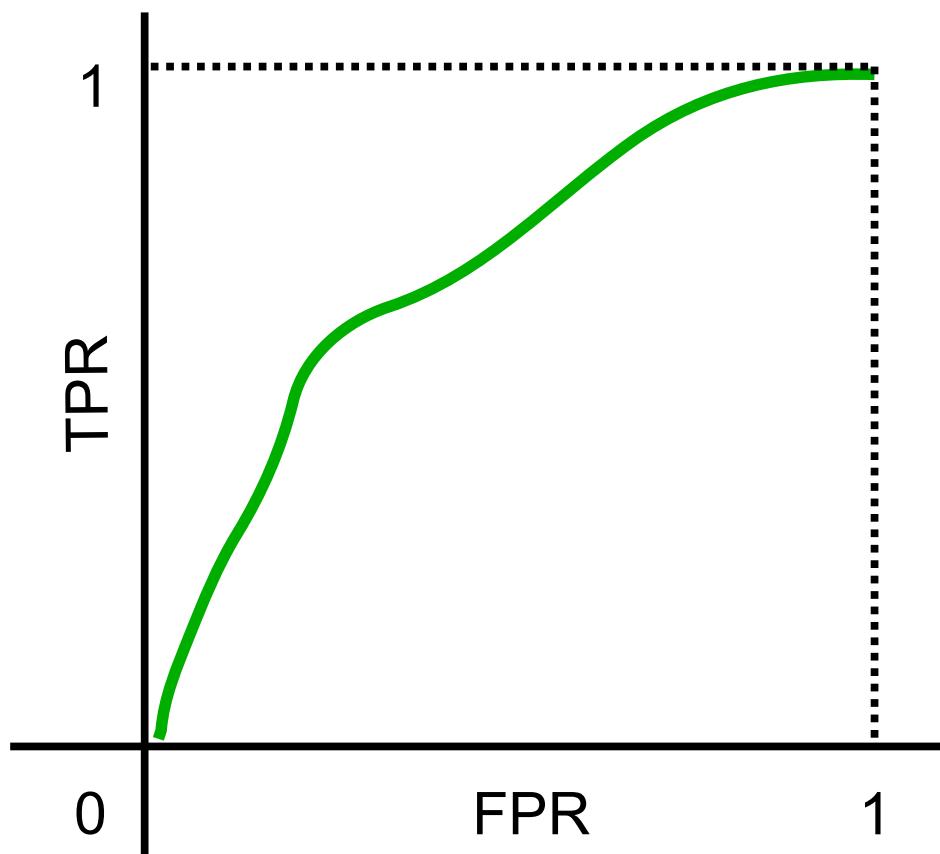


- $TPR = TP / (TP + FP)$   
= 66.67%
- $FPR = FP / (FP + TN)$   
= 33.33%

	Predicted YES	Predicted NO
Actual YES	2	1
Actual NO	1	2

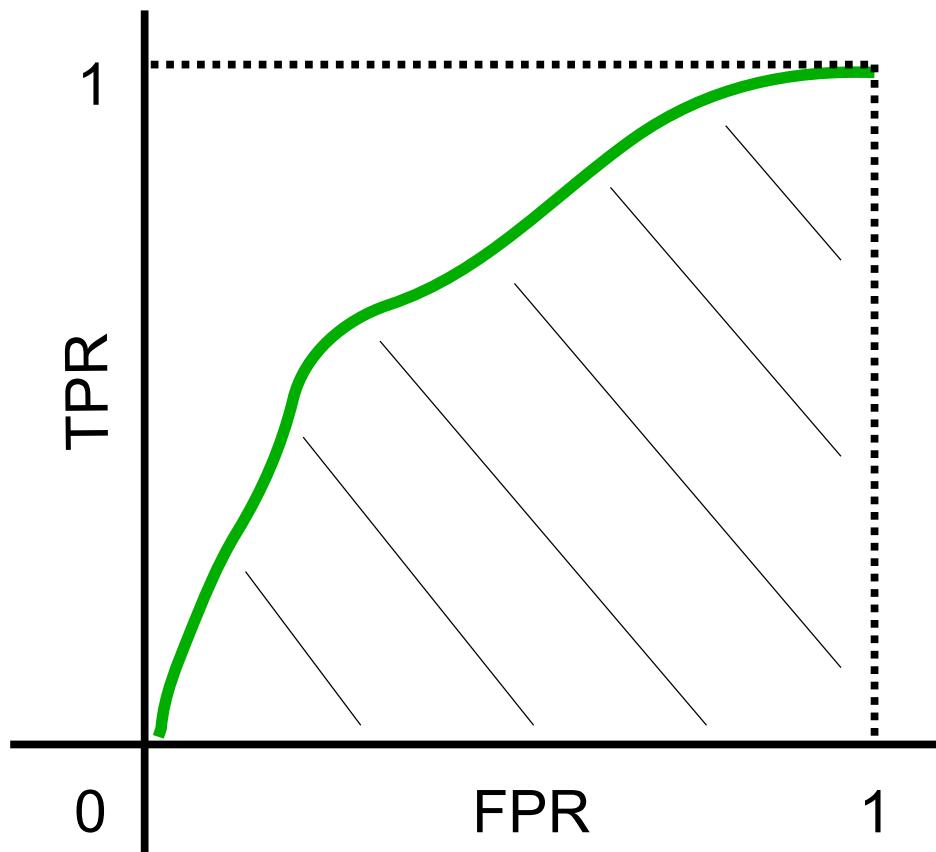
# Cum evaluăm un sistem de învățare automată?

- Curba ROC (Receiver Operating Characteristic)



# Cum evaluăm un sistem de învățare automată?

- Măsura AUC: Aria de sub curba ROC



# Cum evaluăm un sistem de învățare automată?

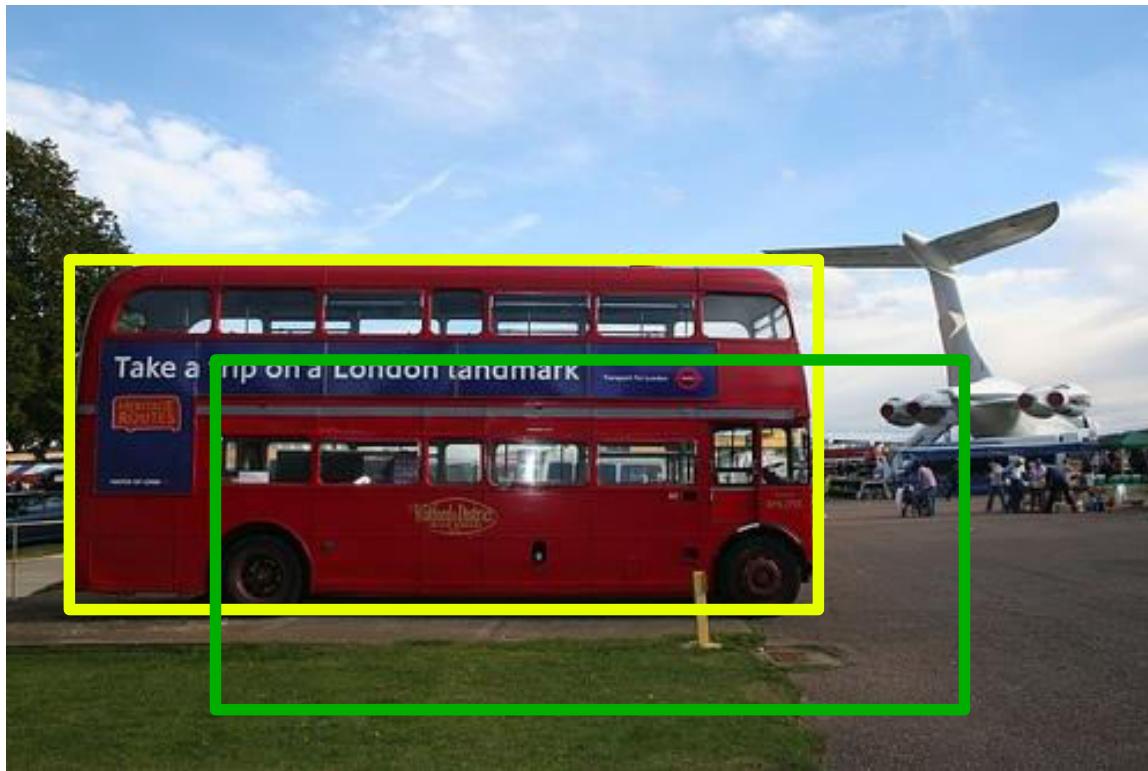
- Măsura  $F_\beta$

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}}$$

- Măsura  $F_1$  este poate cea mai folosită măsură de tipul  $F_\beta$

# Cum evaluăm un sistem de învățare automată?

- Intersecție supra Reuniune (indexul Jaccard)



# Cum evaluăm un sistem de învățare automată?

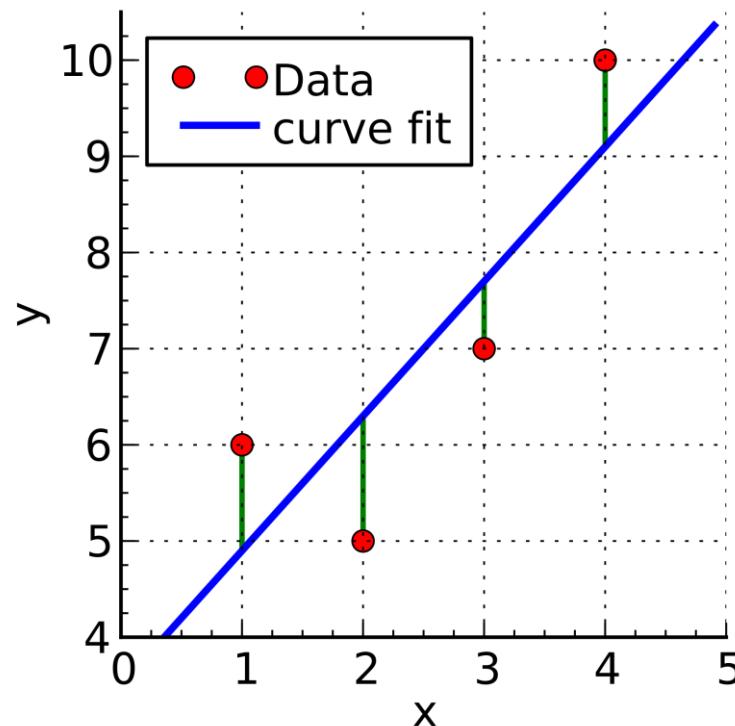
- Intersecție supra Reuniune (indexul Jaccard)
- Detectie corectă dacă  $J(A,B) > 0.5$



# Cum evaluăm un sistem de regresie?

- Media pătratelor erorilor (MSE)

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



# Cum evaluăm un sistem de regresie?

- Ordinea dificultății conform oamenilor



- Ordinea dificultății prezisă de sistem



# Cum evaluăm un sistem de regresie?

- Corelația Kendall Tau:

$$\tau_a = \frac{P - Q}{\frac{n(n-1)}{2}}$$

- Măsură ordinală bazată pe perechi concordante (P) și discordante (Q)

$$P = |\{(i, j) : 1 \leq i < j \leq n, (x_i - x_j)(y_i - y_j) > 0\}|$$

$$Q = |\{(i, j) : 1 \leq i < j \leq n, (x_i - x_j)(y_i - y_j) < 0\}|$$

# Cum evaluăm un sistem de regresie?

- Ordinea dificultății conform oamenilor



- Concordantă cu ordinea prezisă de sistem



# Cum evaluăm un sistem de regresie?

- Ordinea dificultății conform oamenilor



- Discordanță cu ordinea prezisă de sistem



# Cum evaluăm un sistem de regresie?

- Cât este corelația Kendall Tau?



- $P = ?, Q = ?$



# Cum evaluăm un sistem de regresie?

- Cât este corelația Kendall Tau?



- $P = 8, Q = 2, \text{Kendall Tau} = (8-2) / 10 = 0.6$



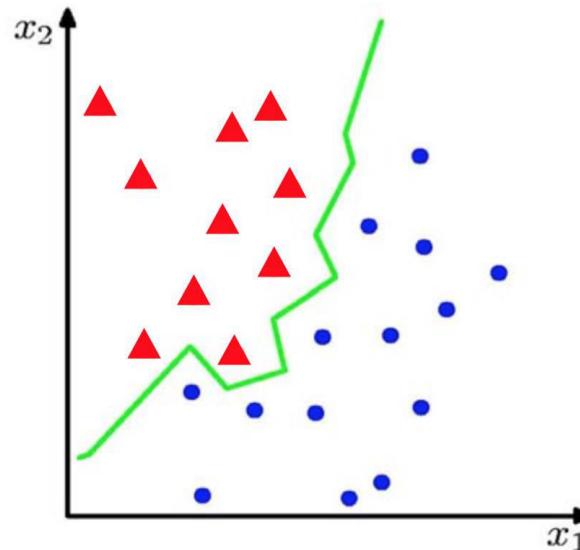
# Metoda celor mai apropiati vecini. “Blestemul dimensionalitatii”.

Prof. Dr. Radu Ionescu

[raducu.ionescu@gmail.com](mailto:raducu.ionescu@gmail.com)

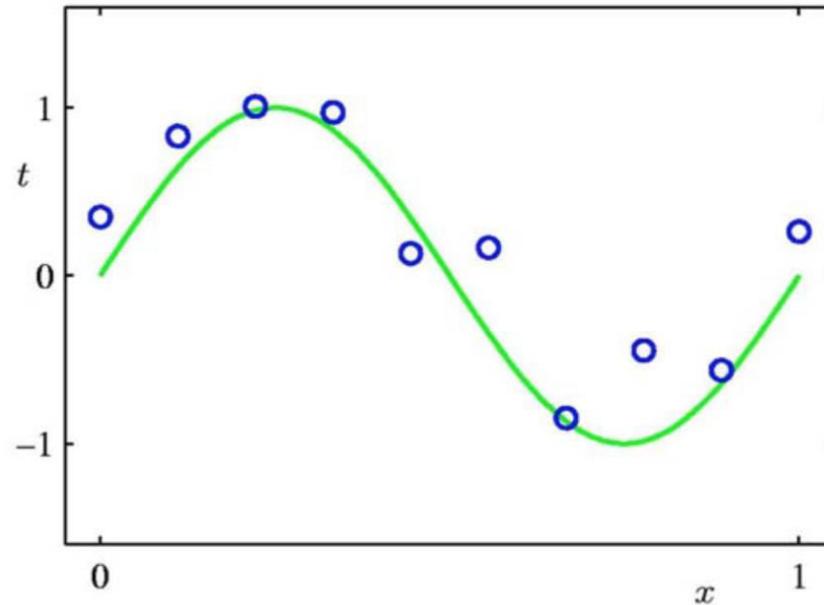
Facultatea de Matematică și Informatică  
Universitatea din București

# Clasificare din exemple etichetate



- Presupunem că avem un set de N exemple de antrenare:  
 $(x_1, \dots, x_N)$  and  $(y_1, \dots, y_N)$ ,  $x_i \in \mathbb{R}^d$ ,  $y_i \in \{-1, 1\}$
- Problema clasificării constă în estimarea funcției  $g(x)$  a.î.:  
$$g(x_i) = y_i$$

# Regresie din exemple etichetate



- Presupunem că avem un set de  $N$  exemple de antrenare:  
 $(x_1, \dots, x_N)$  and  $(y_1, \dots, y_N)$ ,  $x_i, y_i \in \mathbb{R}$
- Problema regresiei constă în estimarea funcției  $g(x)$  a.î.:  
$$g(x_i) = y_i$$

# Învățare din exemple etichetate

- Presupunem că avem un set de N exemple de antrenare:  
 $(x_1, \dots, x_N)$  and  $(y_1, \dots, y_N)$ ,  $x_i, y_i \in \mathbb{R}$
- Problema învățării constă în estimarea funcției  $g(x)$  a.î.:

$$g(x_i) = y_i$$

- Funcție de pierdere (de exemplu MSE):

$$\mathcal{L}(y, g(\mathbf{x}))$$

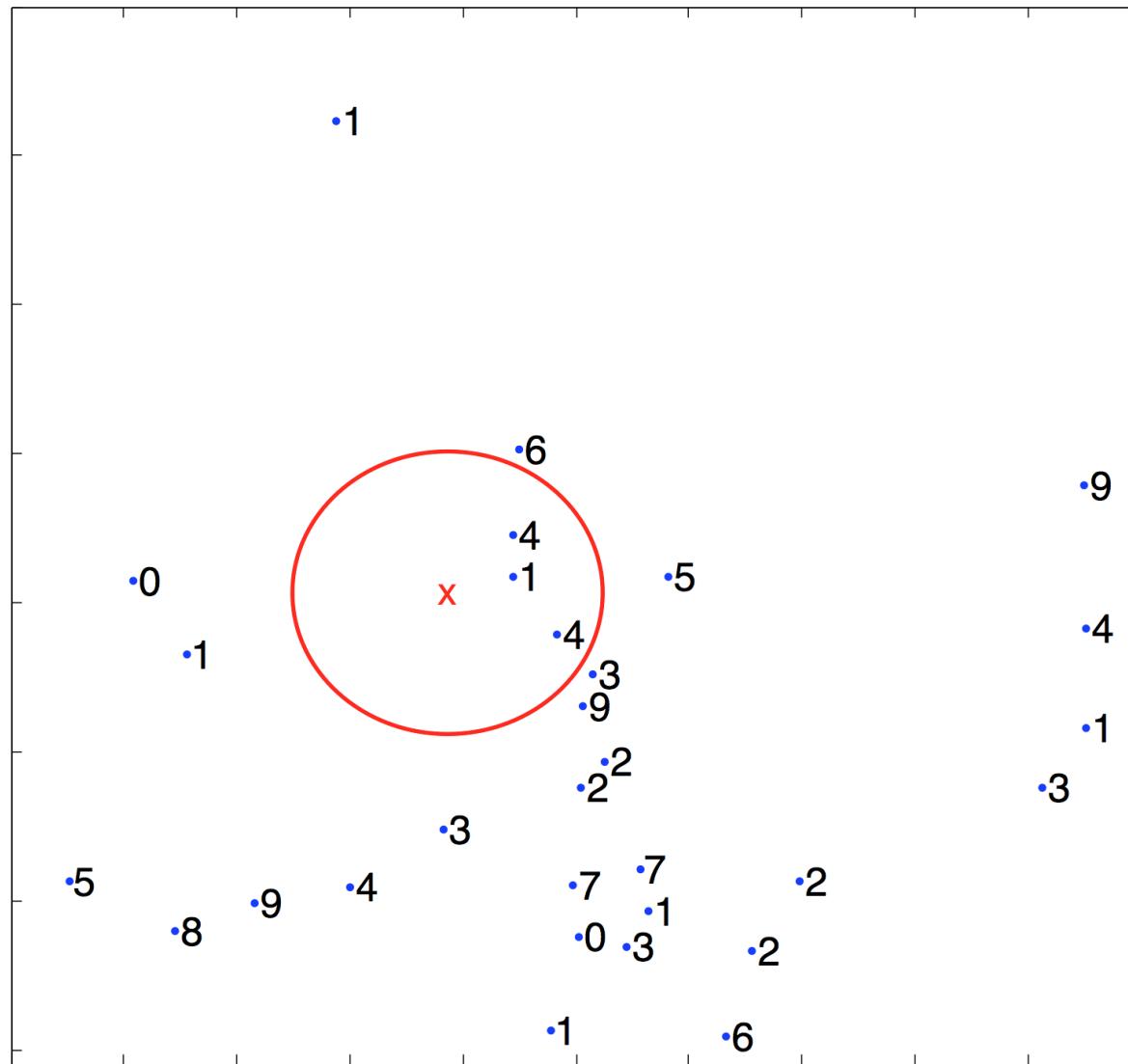
- Eroarea de generalizare:

$$L(g) = E_P \mathcal{L}(y, g(\mathbf{x})) = \int \mathcal{L}(y, g(\mathbf{x})) dP(\mathbf{x}, y)$$

- Eroarea empirică (estimată):

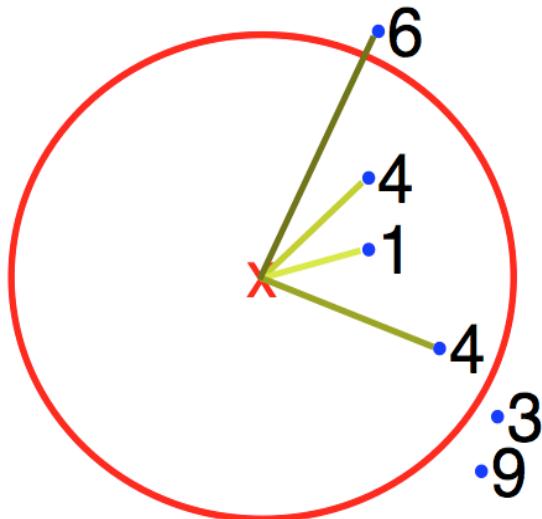
$$L_e(g) = \frac{1}{\ell} \sum_{i=1}^{\ell} \mathcal{L}(y_i, g(\mathbf{x}_i))$$

# Care este eticheta exemplului de test x?



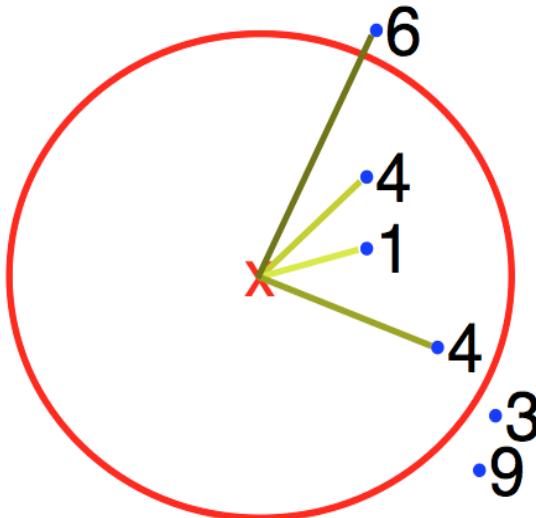
Metoda celor mai apropiati vecini

# Metoda celor mai apropiati vecini



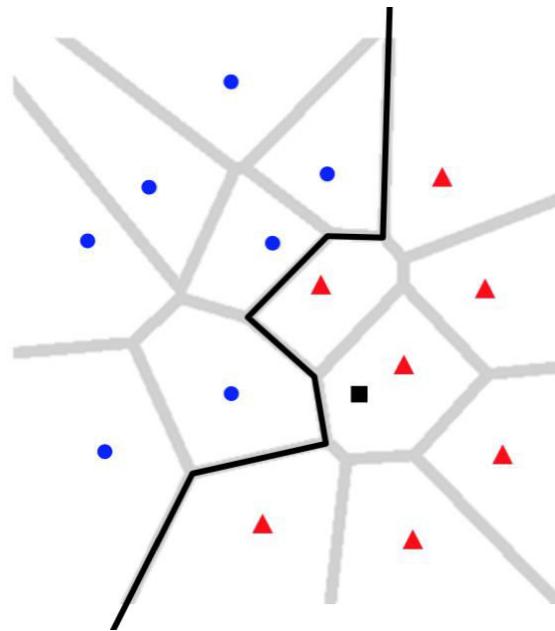
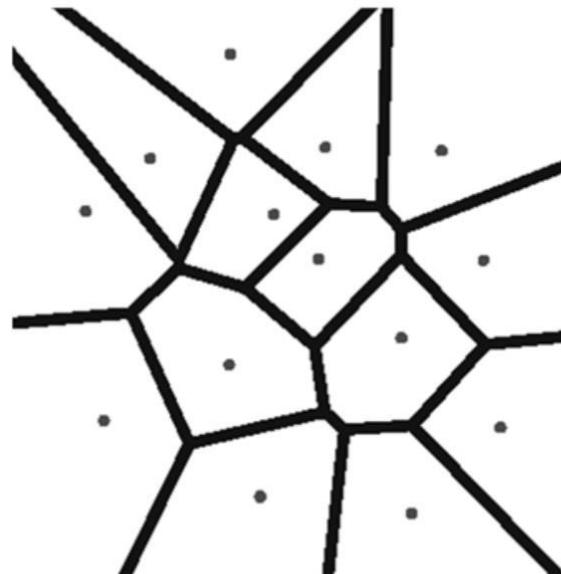
- Algoritmul k-NN:
  - 1) Pentru fiecare exemplu de test  $x$ , găsim cei mai apropiati  $k$  vecini
  - 2) Atribuim eticheta majoritară conform celor  $k$  vecini

# k-Nearest Neighbors (k-NN)



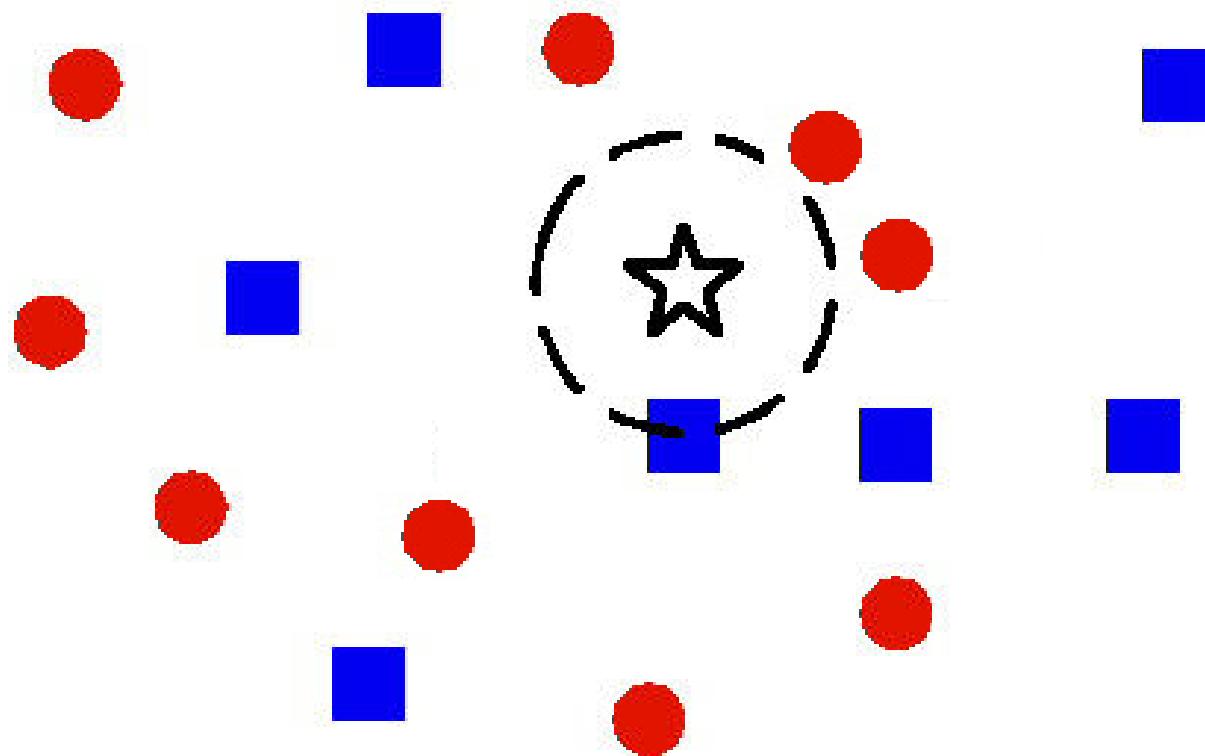
- Cum luăm o decizie în caz de egalitate?
  - 1) Alegem o etichetă din cele egale în mod aleator
  - 2) Aplicăm modelul 1-NN (nu există egalități)
  - 3) Utilizăm distanțele până la exemplu de test ca ponderi

# Ce se întâmplă în cazul $k = 1$ ?

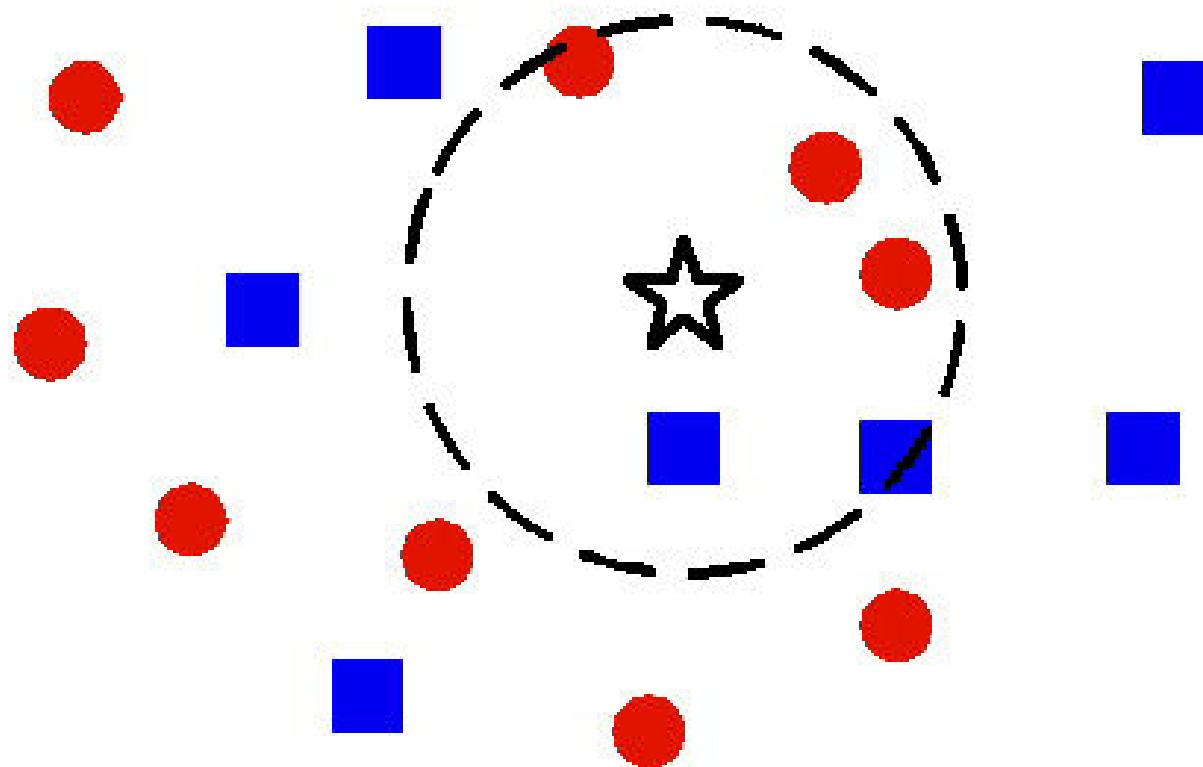


- Obținem o diagramă Voronoi:
  - spațiul este partit ionat în regiuni
  - granțiele de separare trec prin zonele în care distanțele între exemplele de training sunt egale
- Granița de separare este neliniară

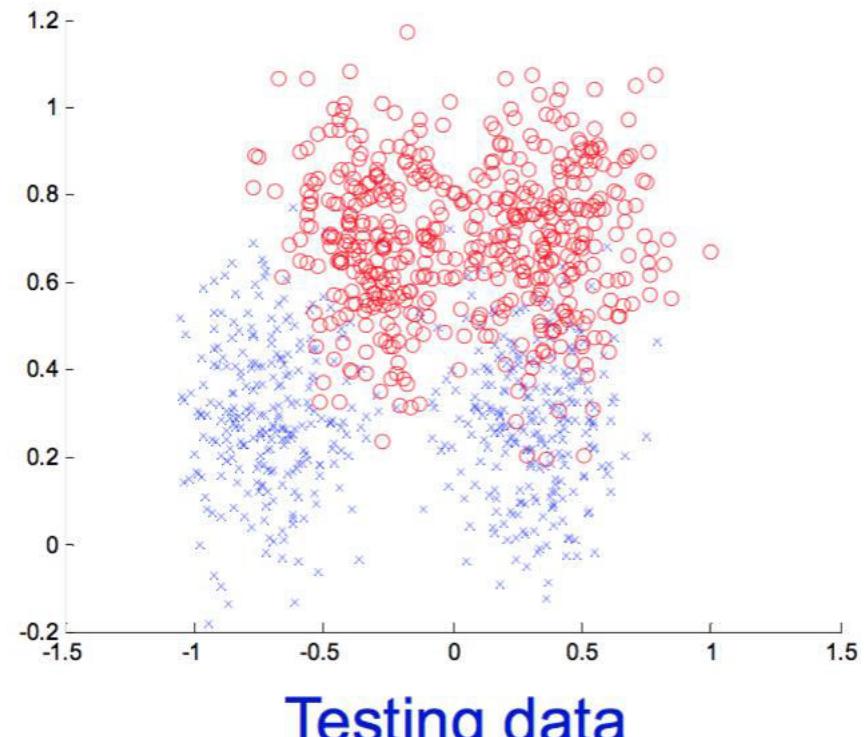
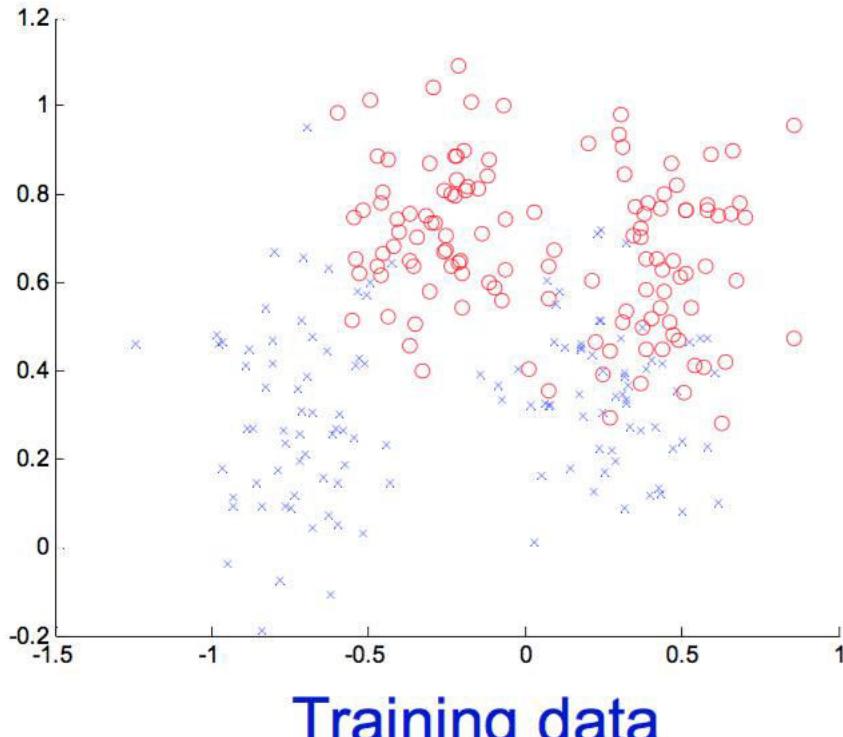
# 1-NN versus k-NN



# 1-NN versus k-NN



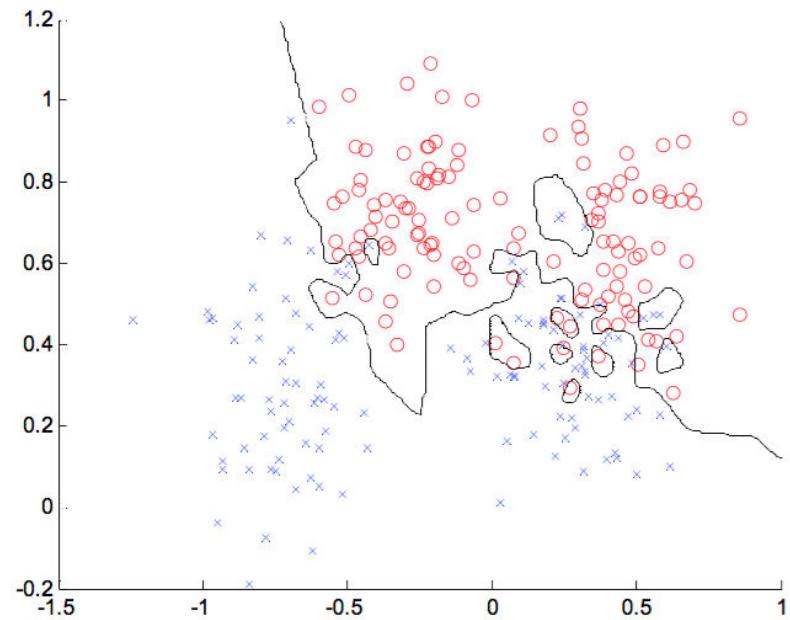
# Presupunerea pe care se bazează modelul k-NN



- Datele de antrenare și cele de testare provin din aceeași distribuție
- Devine puțin probabil ca un pattern reprezentativ în setul de antrenare să fie absent în datele de test

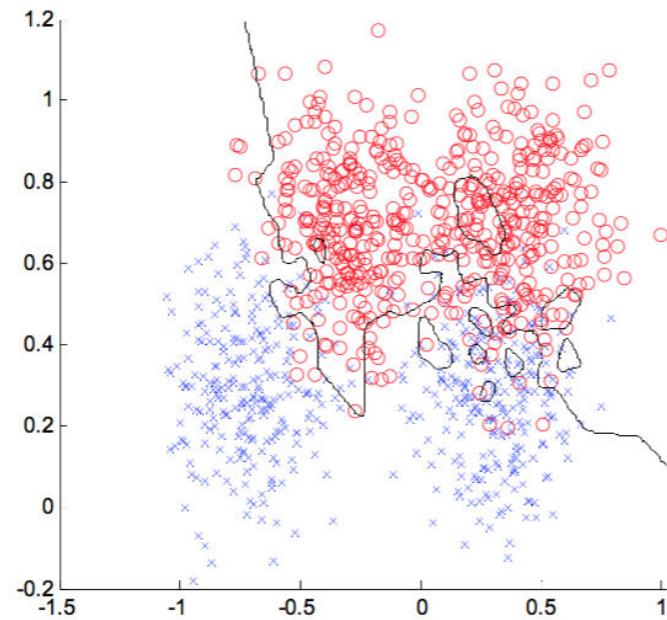
# Ce se întâmplă atunci când variem parametrul k?

Training data



error = 0.0

Testing data

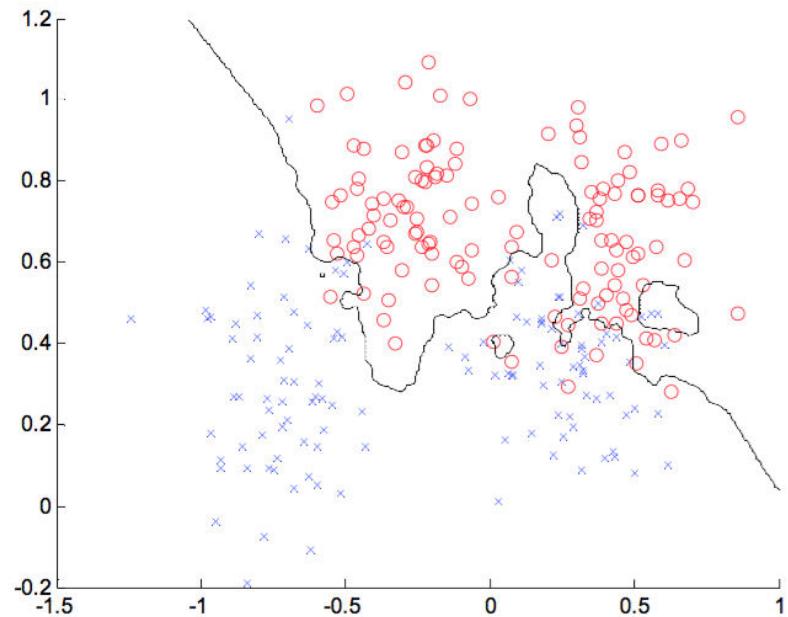


error = 0.15

- $k = 1$

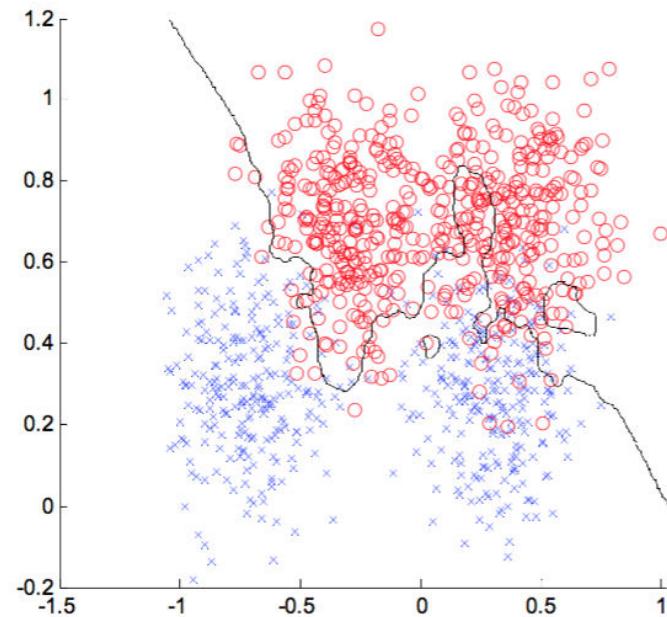
# Ce se întâmplă atunci când variem parametrul k?

Training data



error = 0.0760

Testing data

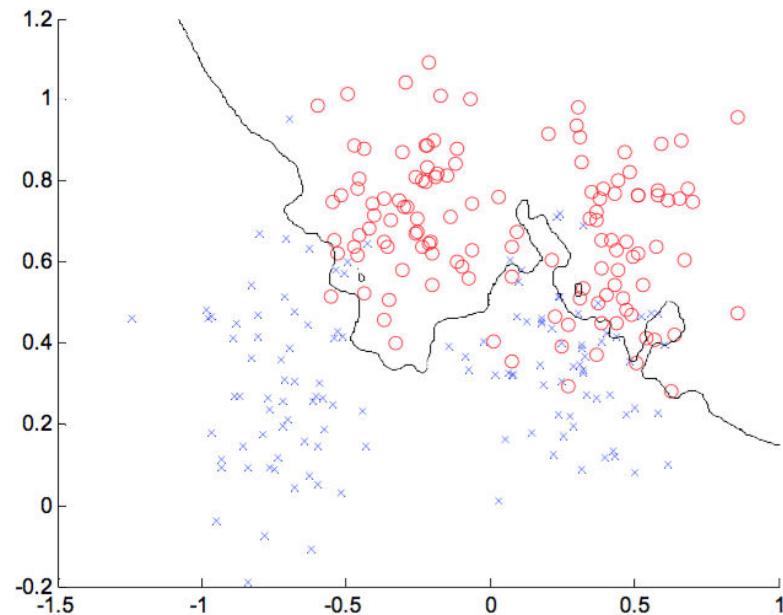


error = 0.1340

- $k = 3$

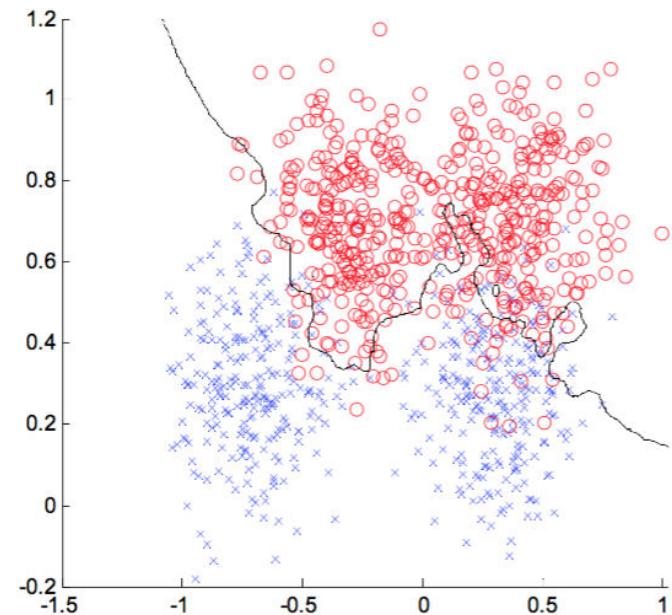
# Ce se întâmplă atunci când variem parametrul k?

Training data



error = 0.1320

Testing data

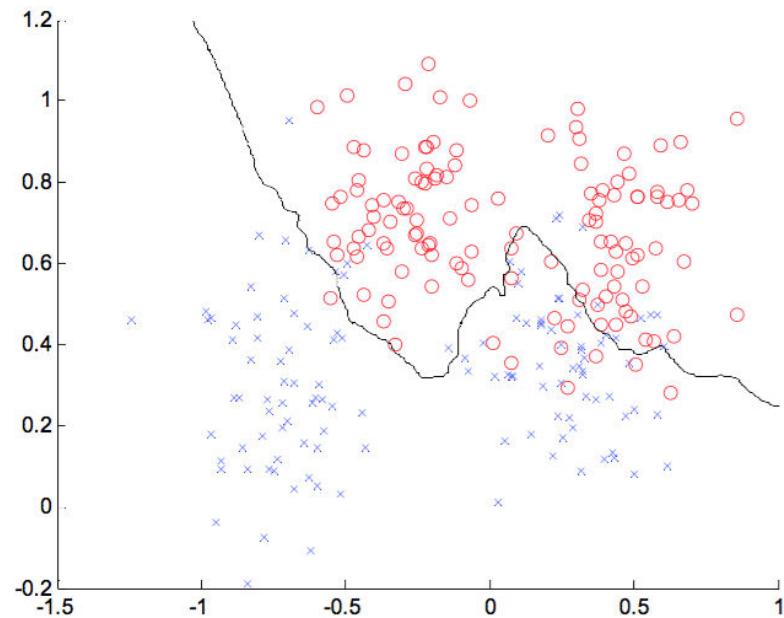


error = 0.1110

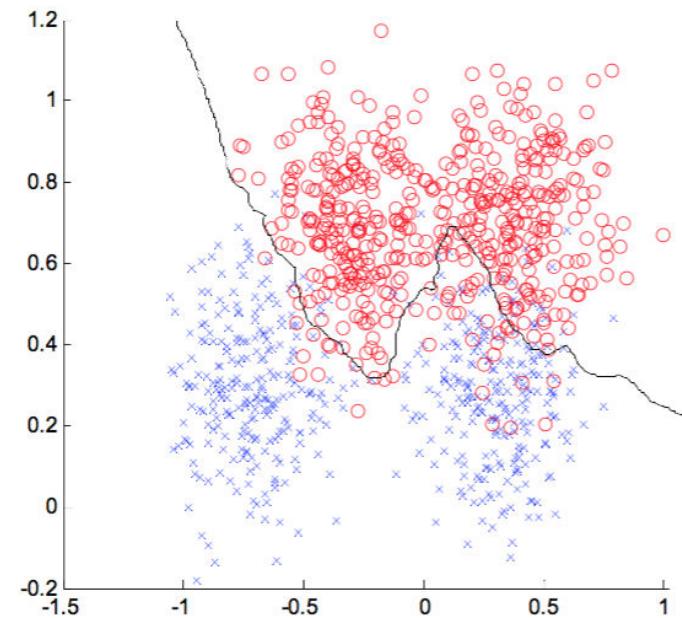
- $k = 7$

# Ce se întâmplă atunci când variem parametrul k?

Training data



Testing data



error = 0.1120

error = 0.0920

- $k = 21$

# Ce ne trebuie pentru un clasificator bazat pe memorie?

- O funcție de distanță
  - Distanța Euclidiană
  - Distanța Edit (Levenshtein)
  - Distanța Hamming
- Câți vecini să luăm în considerare?
- Cum să antrenăm modelul pe exemplele din vecinătate?

# În cazul 1-NN

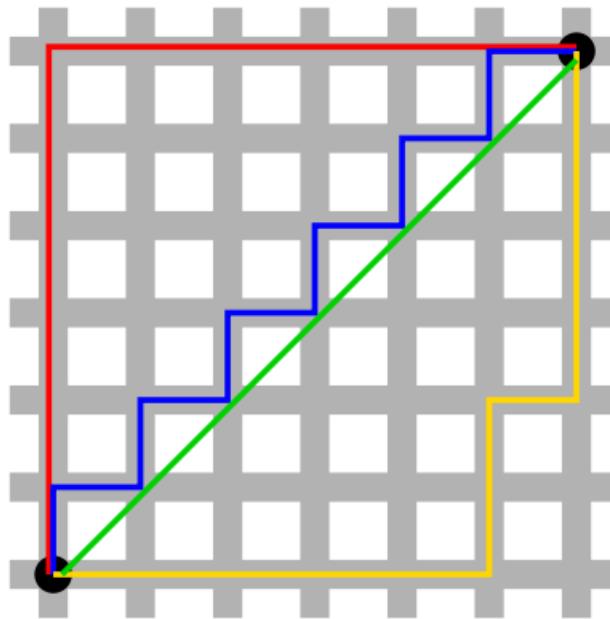
- O funcție de distanță
  - **De exemplu distanța Euclidiană**
- Câți vecini să luăm în considerare?
  - 1
- Cum să antrenăm modelul pe exemplele din vecinătate?
  - **Prezicem eticheta celui mai apropiat vecin**

# Distanța Euclidiană ( $L_2$ )

- Pentru vectorii  $x = (5, 1, 3, 0)$  și  $y = (2, 1, 4, 1)$  avem:

$$\begin{aligned}d_{L_2}(x, y) &= \sqrt{(x_1 - y_1)^2 + \cdots + (x_n - y_n)^2} \\&= \sqrt{(5 - 2)^2 + (1 - 1)^2 + (3 - 4)^2 + (0 - 1)^2} \\&= \sqrt{9 + 1 + 1} = \sqrt{11} \\&\cong 3.32\end{aligned}$$

# Distanța Manhattan ( $L_1$ )



- Pentru vectorii  $x = (5, 1, 3, 0)$  și  $y = (2, 1, 4, 1)$  avem:

$$\begin{aligned}d_{L_1}(x, y) &= |x_1 - y_1| + \cdots + |x_n - y_n| \\&= |5 - 2| + |1 - 1| + |3 - 4| + |0 - 1| \\&= 3 + 1 + 1 = 5\end{aligned}$$

# Distanța Minkowski ( $L_p$ )

- Pentru vectorii  $x = (x_1, \dots, x_n)$  și  $y = (y_1, \dots, y_n)$  avem:

$$d_{L_p}(x, y) = \sqrt[p]{|x_1 - y_1|^p + \cdots + |x_n - y_n|^p}$$

- Distanța Minkowski este o generalizare pentru distanțele Euclidiană ( $p = 2$ ) și Manhattan ( $p = 1$ )
- Dacă  $p < 1$ , atunci nu mai este distanță. Nu respectă inegalitatea triunghiului pentru  $x = (0,0)$ ,  $y = (1,1)$  și  $z = (0,1)$ :

$$d_{L_{p<1}}(x, y) > d_{L_{p<1}}(x, z) + d_{L_{p<1}}(z, y)$$

# Distanța Hamming

- De exemplu, utilă pentru probleme de clasificare cu date categorice sau secvențe ADN
- Pentru vectorii  $x = (A, G, T, C)$  și  $y = (G, G, T, A)$  avem:
$$d_{Hamming}(x, y) = 1 + 0 + 0 + 1 = 2$$
- Câte trăsături (componente) diferă între cei doi vectori

# Distanța Edit (Levenshtein)

- De exemplu, utilă pentru probleme de clasificare cu siruri de caractere (documente text sau secvențe ADN), secvențe temporale (imagini video)
- Distanța este dată de câte modificări (inserare, ștergere, înlocuire) sunt necesare pentru a transforma un obiect în cel de-al doilea
- Pentru secvențe video, folosim Dynamic Time Warping (DTW)

# Aplicație: Clasificarea gesturilor

- Ce gest reprezintă mișcarea persoanei?



# Considerăm 10 clase de gesturi



# Problema recunoașterii gesturilor

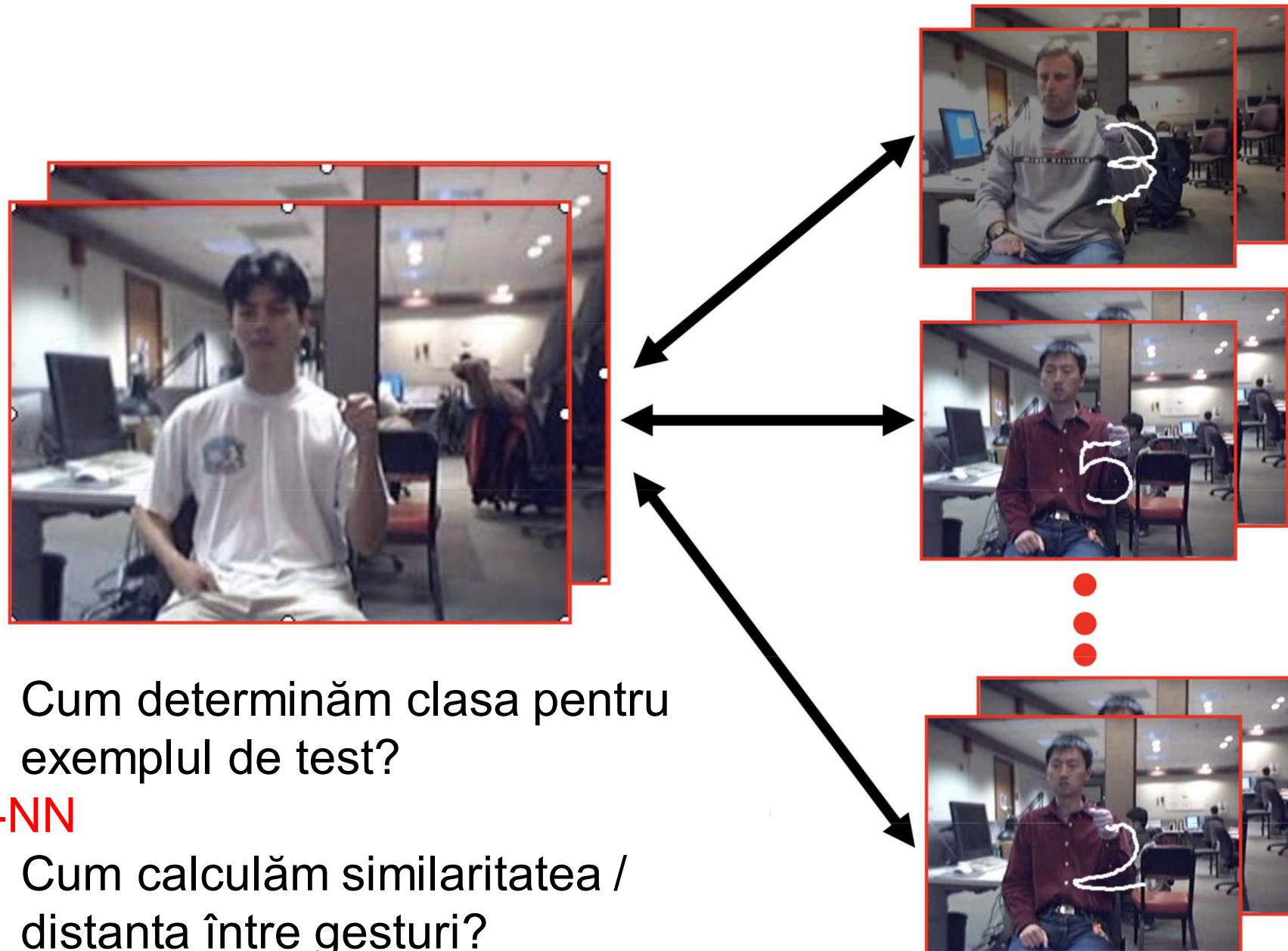
- Trebuie să recunoaștem 10 gesturi simple realizate de către o persoană
  - Fiecare gest corespunde unui număr de la 0 la 9
  - Contează doar traекторia urmată de mâna, nu și forma mâinii / poziția degetelor
- Acestă este doar o alegere valabilă pentru problema aleasă
- În multe situații (recunoașterea limbajului semnelor), poziția degetelor este utilă

# Descompunerea problemei

Avem nevoie de mai multe sisteme pentru:

- A determina / estima cum s-a mișcat persoana
  - Detectarea și urmărirea persoanei
  - Detectarea și urmărirea mâinii
- **A recunoaște ce reprezintă mișcarea**
  - Estimarea și recunoașterea mișcării sunt lucruri complet diferite:
    - Atunci când cineva comunică prin limbajul semnelor, vedem cum se mișcă, dar nu înțelegem ce reprezintă

# Recunoașterea gesturilor

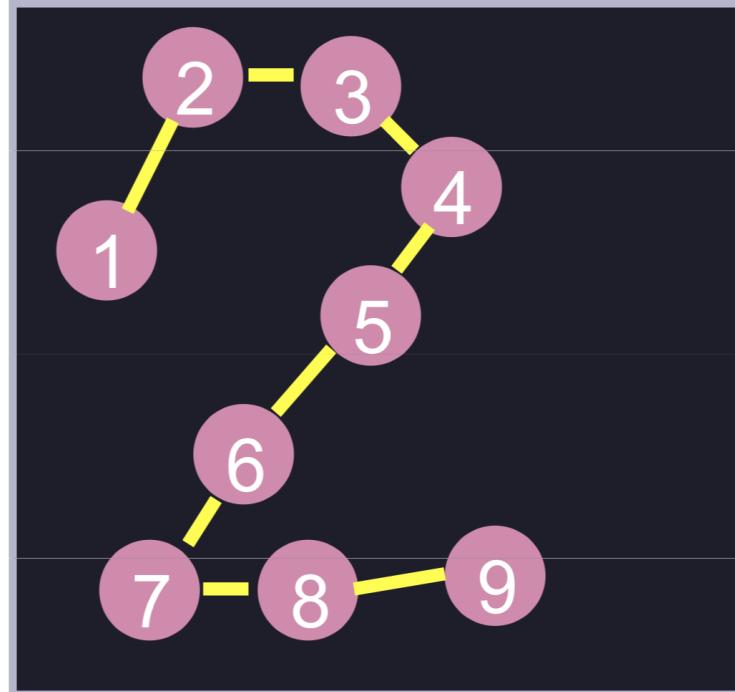
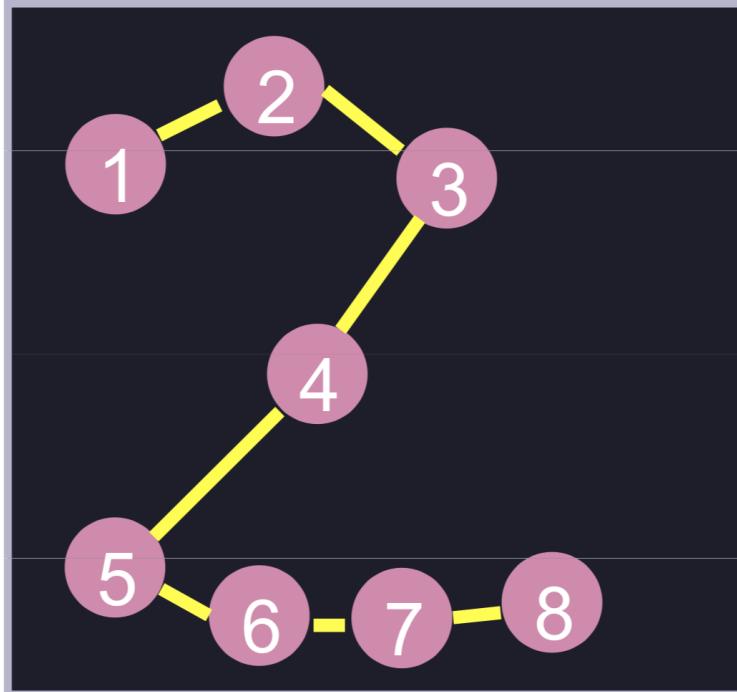


# Presupunem cunoscută locația mâinii



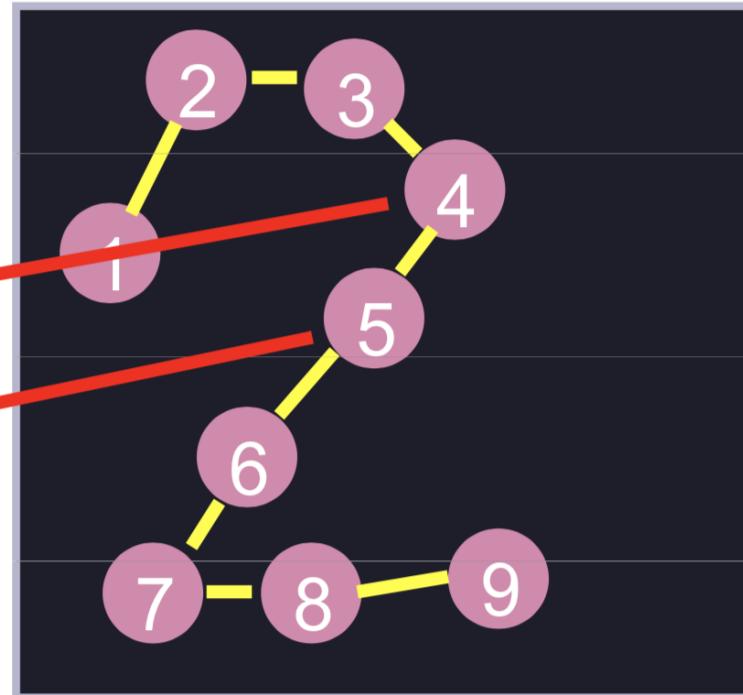
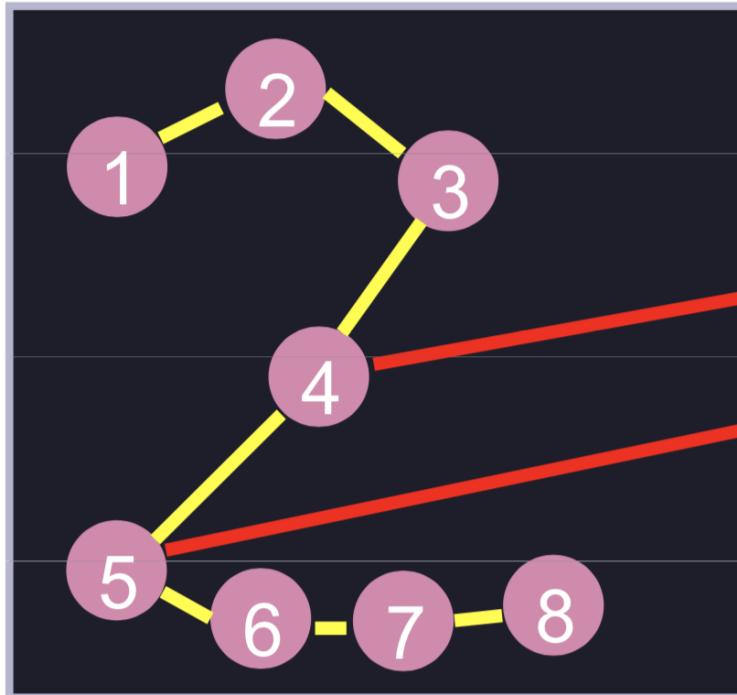
- Presupunem cunoscută poziția mâinii pentru toate exemplele
- Pentru video-urile din setul de antrenare adnotarea se poate realiza manual
- Pentru video-urile de test, avem nevoie de un detector (nu discutăm acest aspect)

# Compararea traiectoriilor



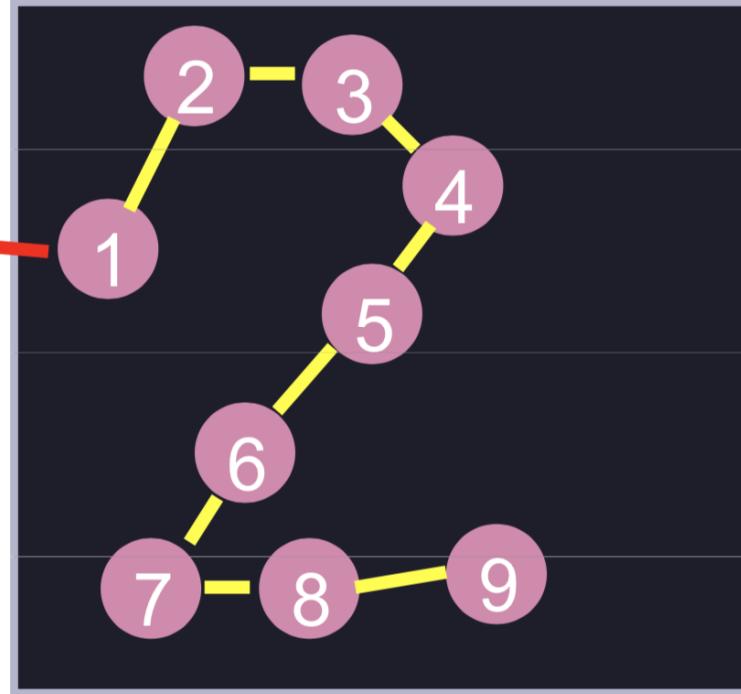
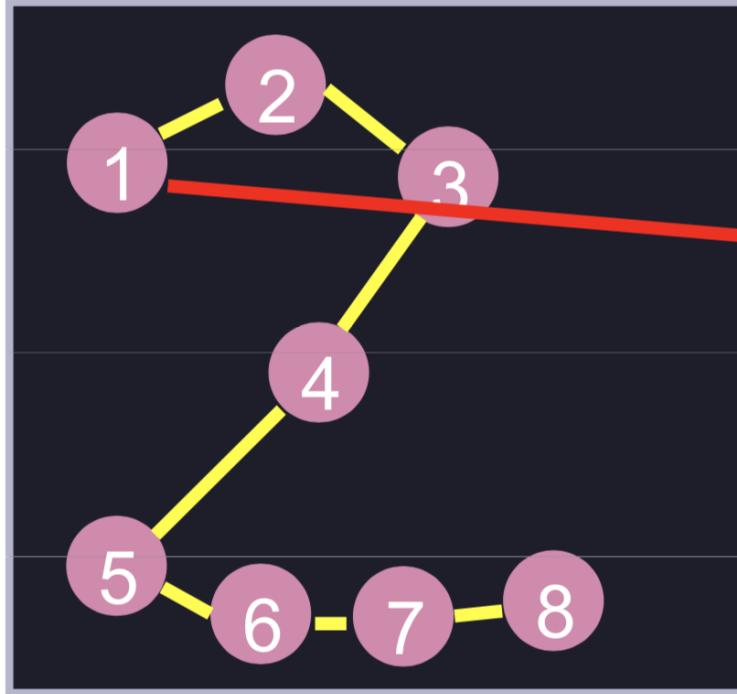
- Putem obține o traiectorie pentru fiecare gest, pe baza locației mâinii din fiecare cadru
- Cum comparăm traiectoriile?

# Alinierea traiectoriilor



- Dacă împerechem cadrul i din stânga cu cadrul i din dreapta, atunci ce facem cu cadrul 9 din dreapta?

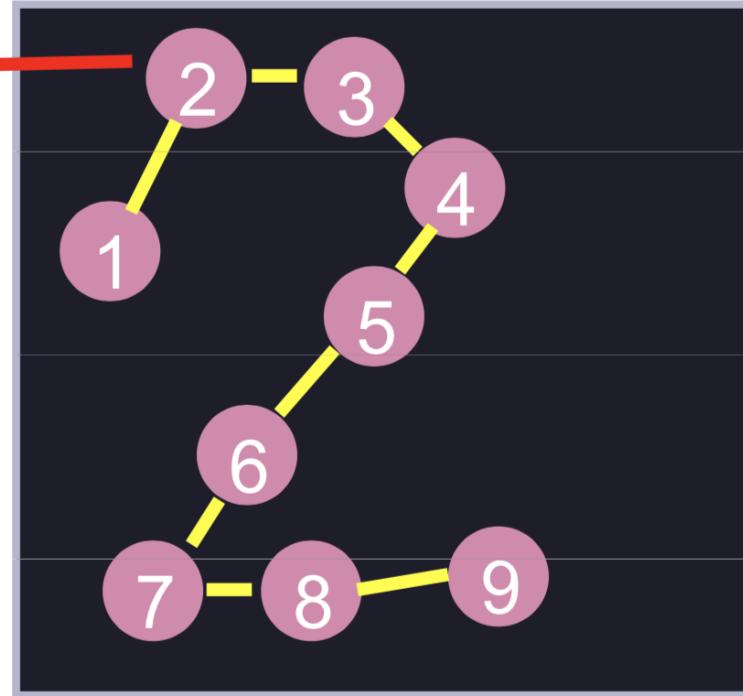
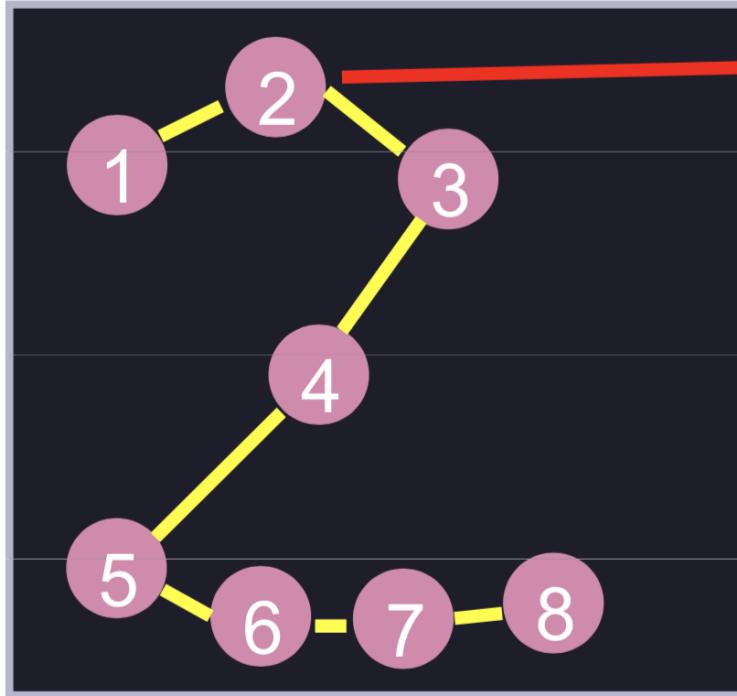
# Alinierea traiectoriilor



- Am putea să le aliniem:

((1, 1)

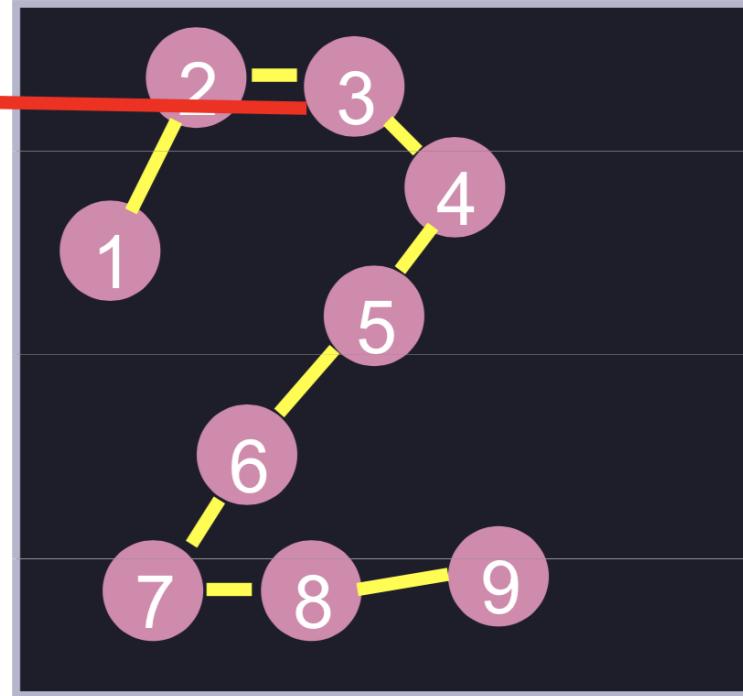
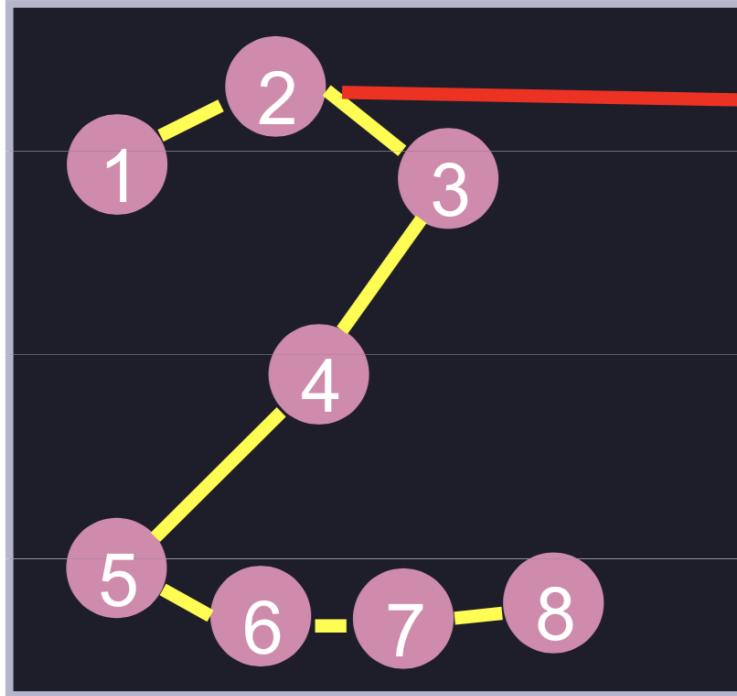
# Alinierea traiectoriilor



- Am putea să le aliniem:

((1, 1), (2, 2))

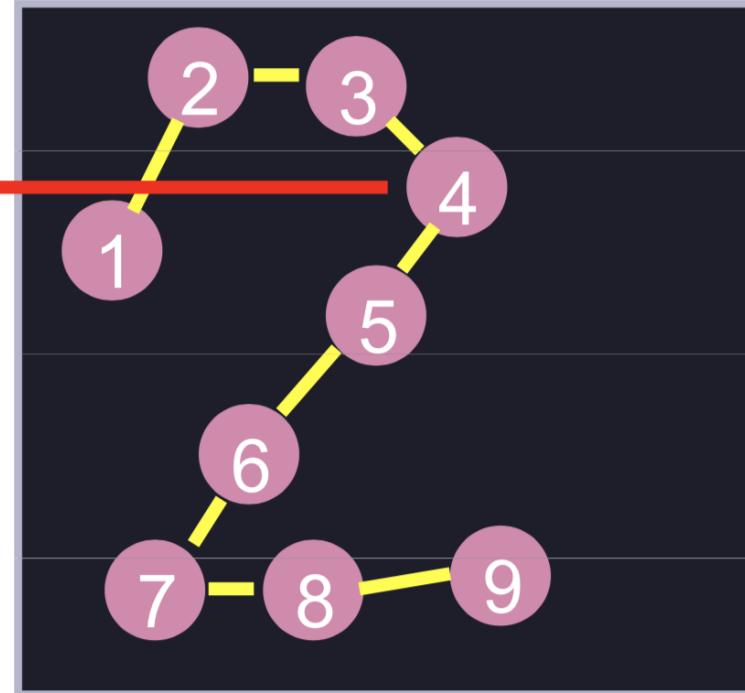
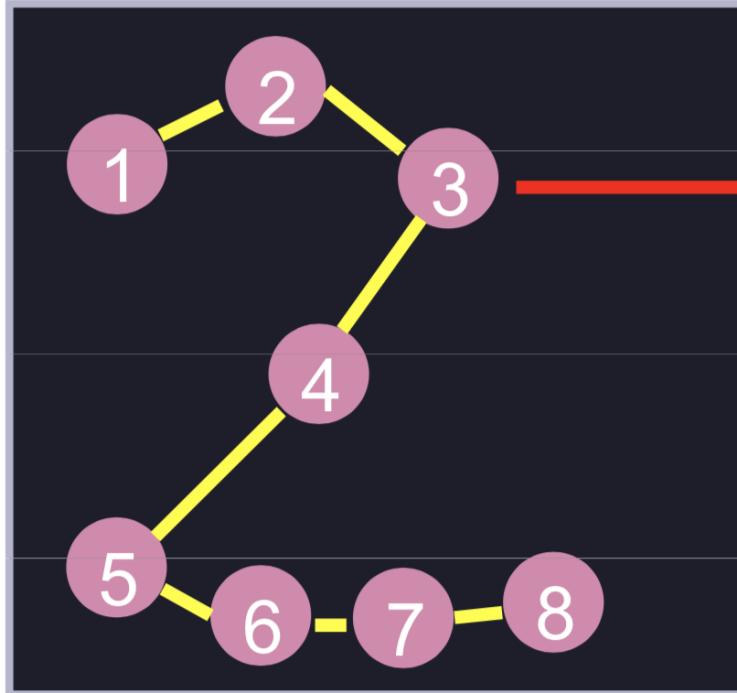
# Alinierea traiectoriilor



- Am putea să le aliniem:

$((1, 1), (2, 2), (2, 3)$

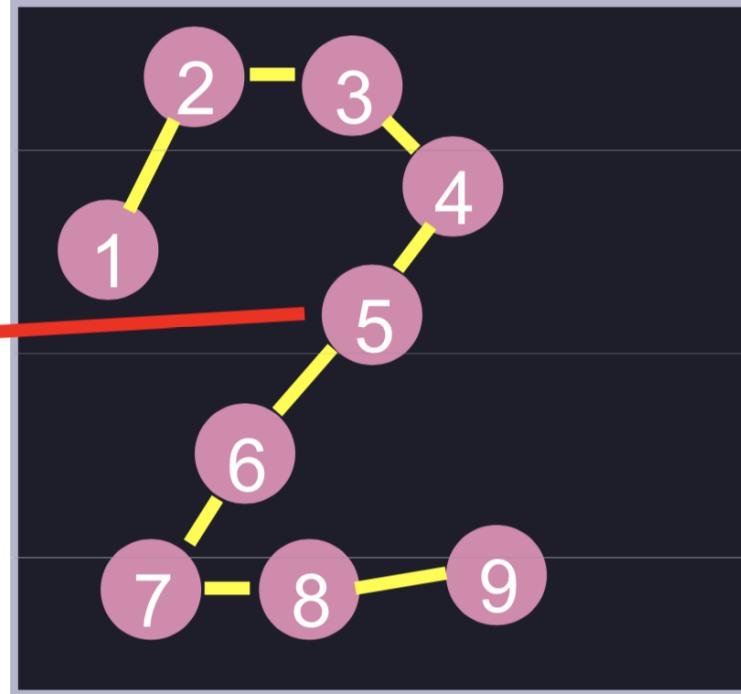
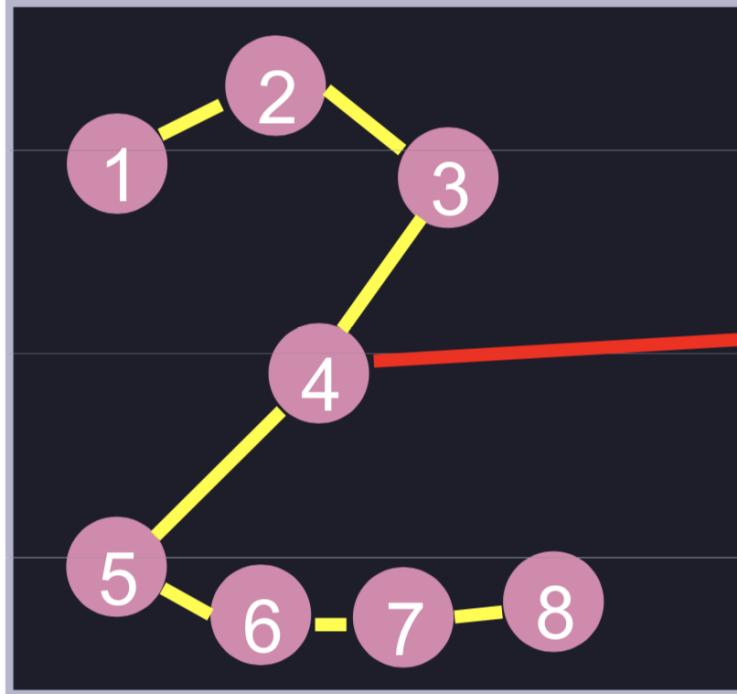
# Alinierea traiectoriilor



- Am putea să le aliniem:

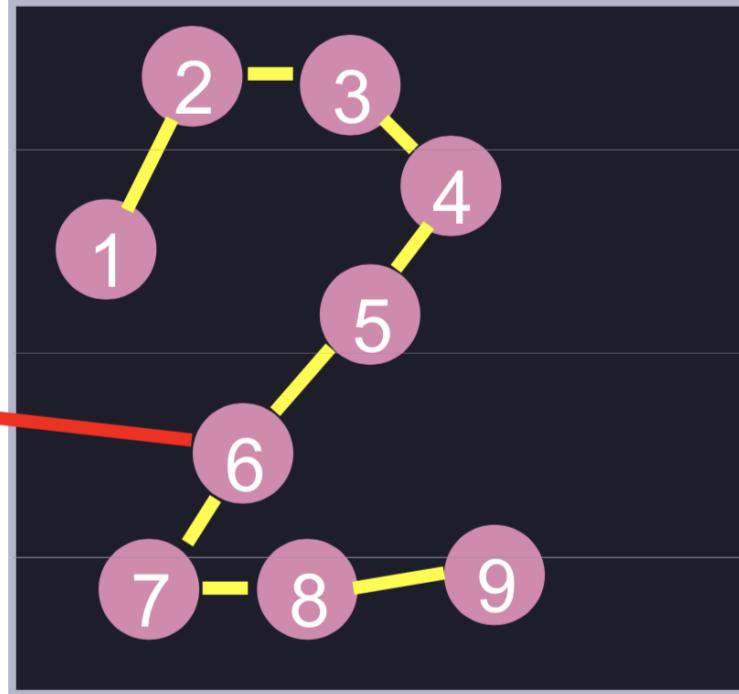
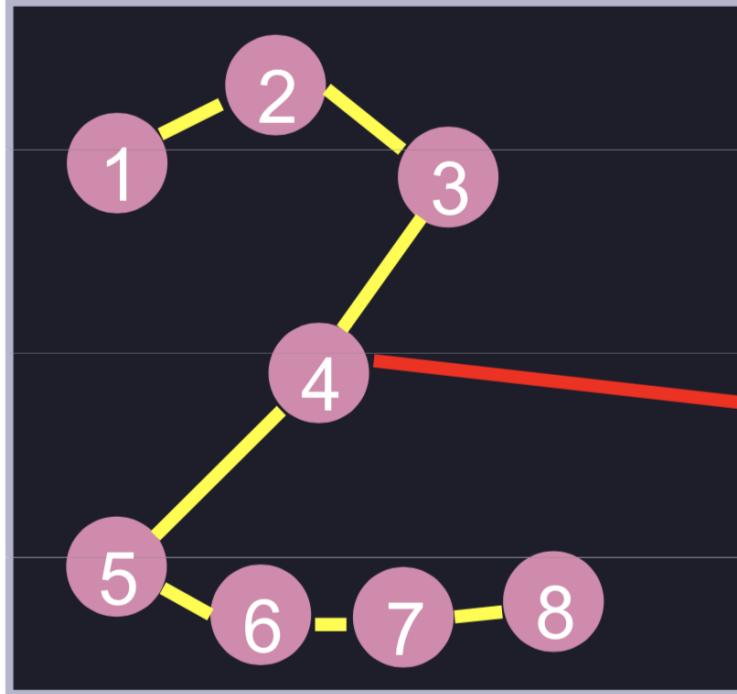
$((1, 1), (2, 2), (2, 3), (3, 4)$

# Alinierea traiectoriilor



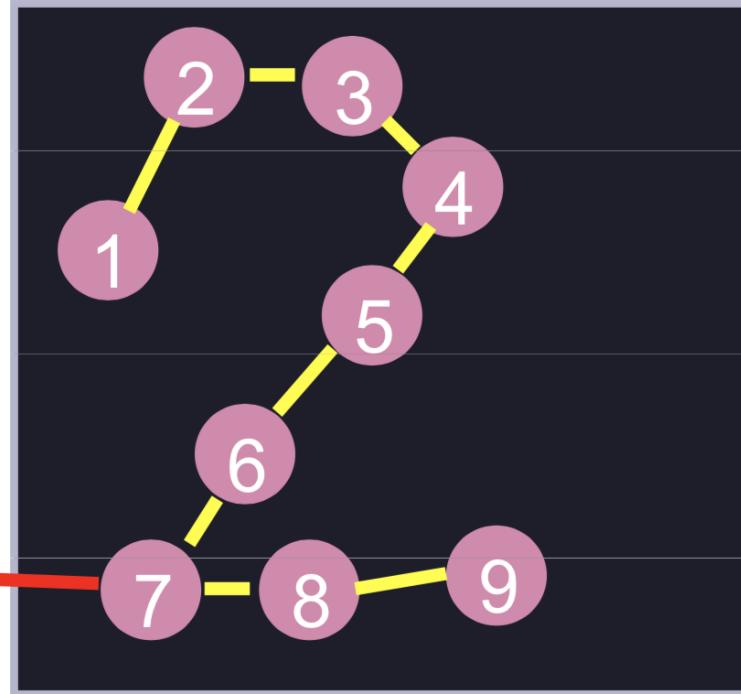
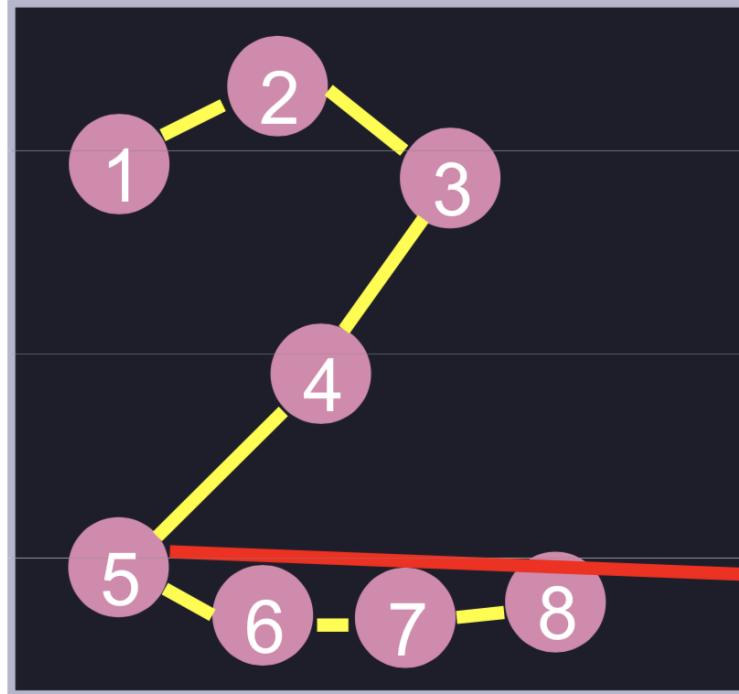
- Am putea să le aliniem:  
((1, 1), (2, 2), (2, 3), (3, 4), (4, 5))

# Alinierea traiectoriilor



- Am putea să le aliniem:  
((1, 1), (2, 2), (2, 3), (3, 4), (4, 5), **(4, 6)**)

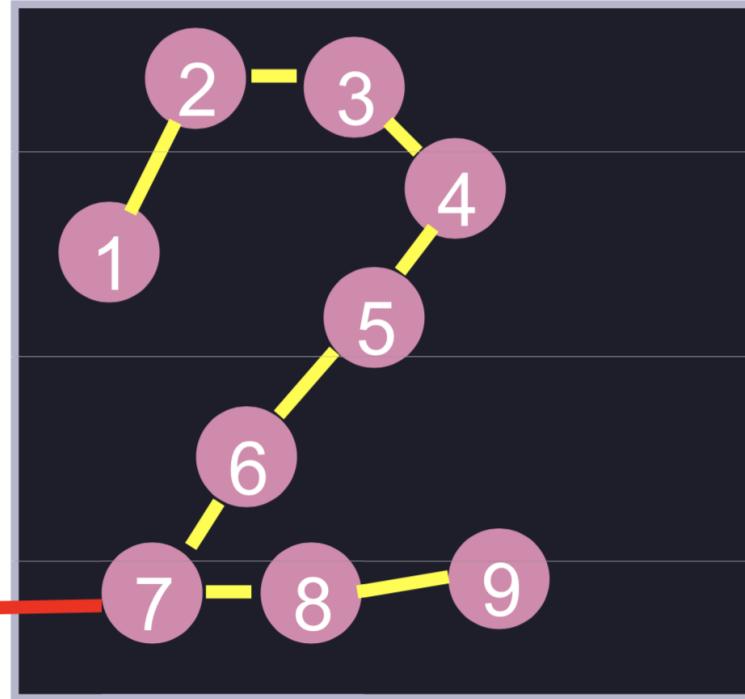
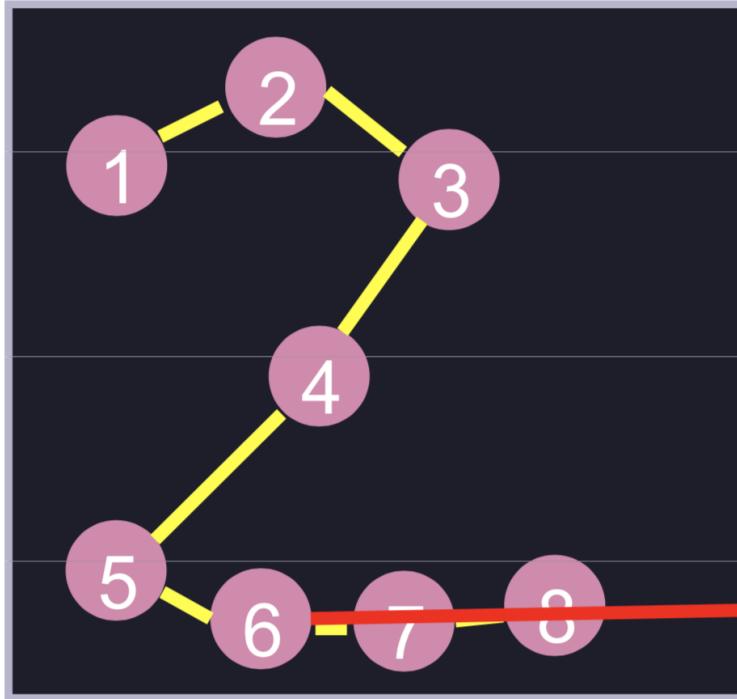
# Alinierea traiectoriilor



- Am putea să le aliniem:

$((1, 1), (2, 2), (2, 3), (3, 4), (4, 5), (4, 6), (5, 7)$

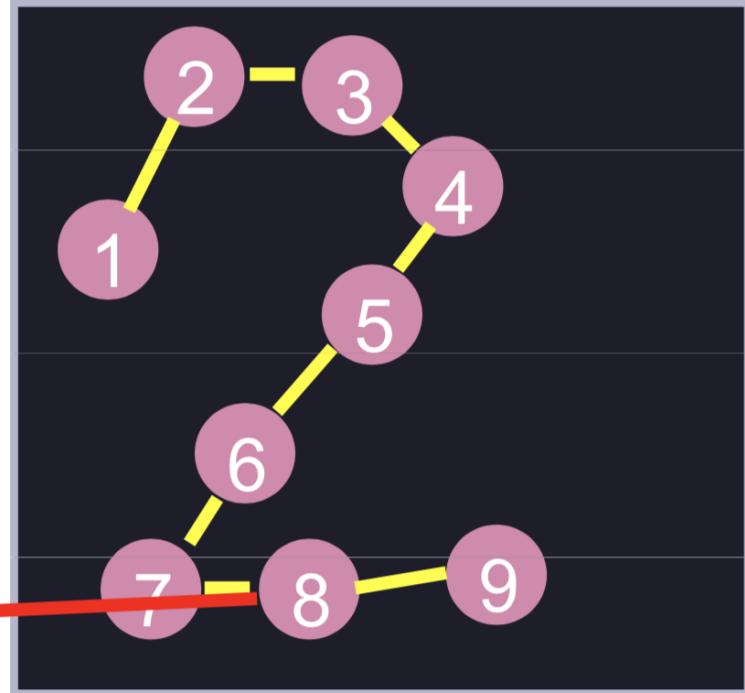
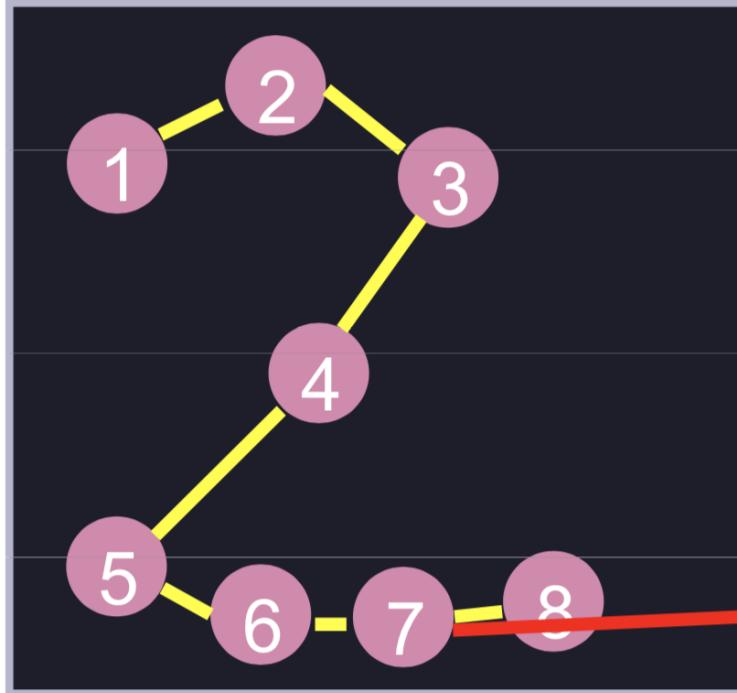
# Alinierea traiectoriilor



- Am putea să le aliniem:

$((1, 1), (2, 2), (2, 3), (3, 4), (4, 5), (4, 6), (5, 7), (6, 7))$

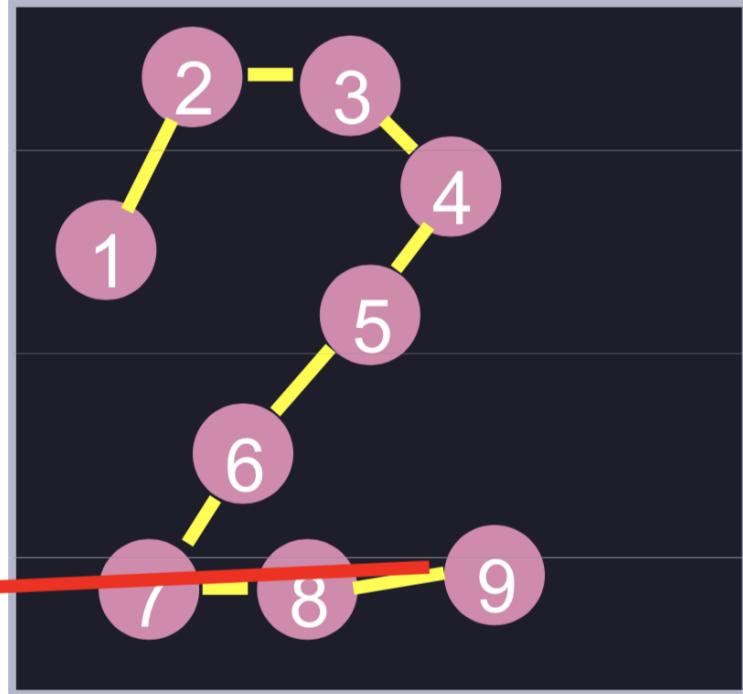
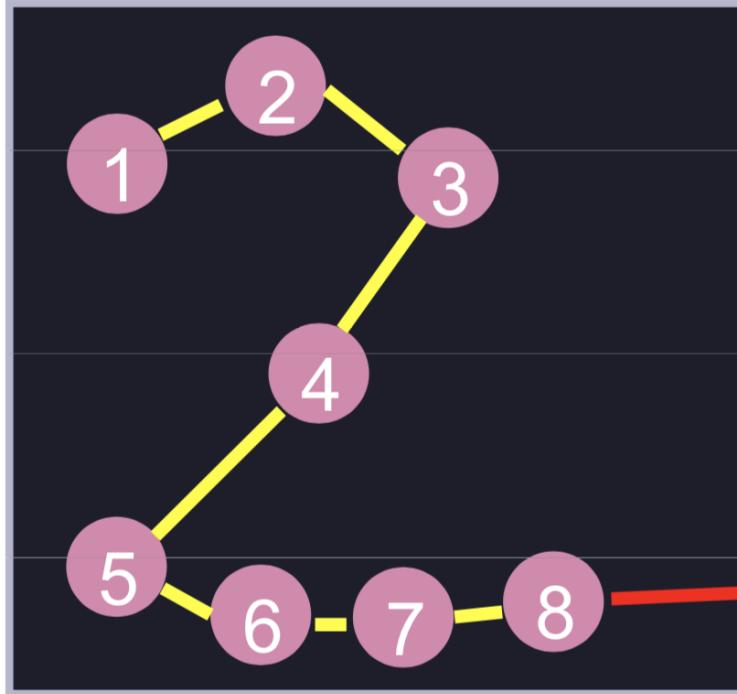
# Alinierea traiectoriilor



- Am putea să le aliniem:

$((1, 1), (2, 2), (2, 3), (3, 4), (4, 5), (4, 6), (5, 7), (6, 7), (7, 8)$

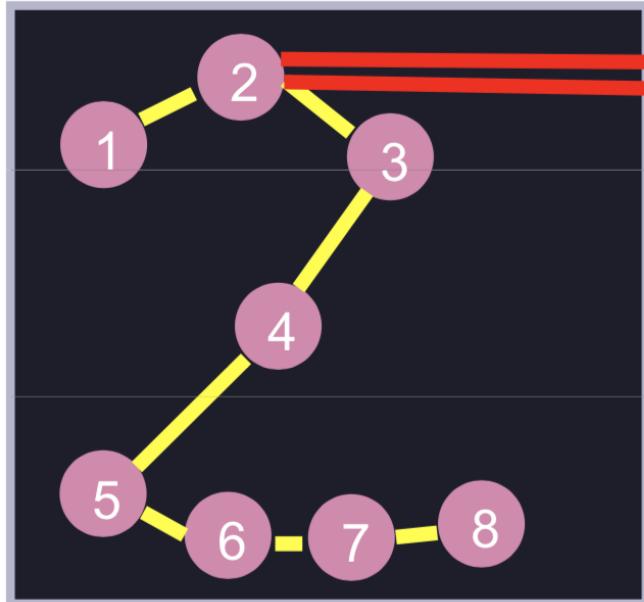
# Alinierea traiectoriilor



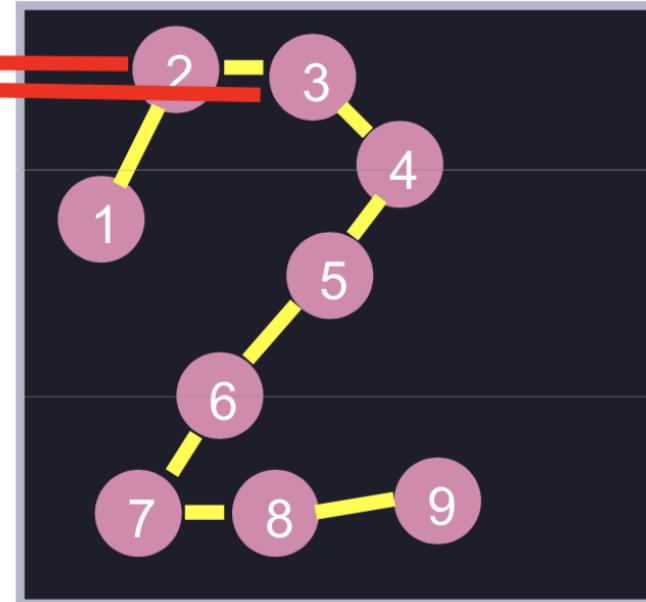
- Am putea să le aliniem:

$((1, 1), (2, 2), (2, 3), (3, 4), (4, 5), (4, 6), (5, 7), (6, 7), (7, 8), (8, 9))$

# Alinierea traiectoriilor



$$M = (M_1, M_2, \dots, M_8).$$



$$Q = (Q_1, Q_2, \dots, Q_9).$$

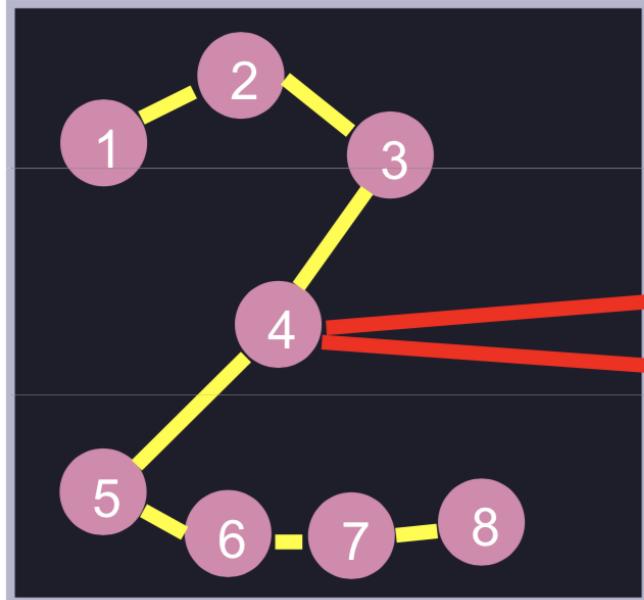
- Am putea să le aliniem:

$((1, 1), (2, 2), (2, 3), (3, 4), (4, 5), (4, 6), (5, 7), (6, 7), (7, 8), (8, 9))$

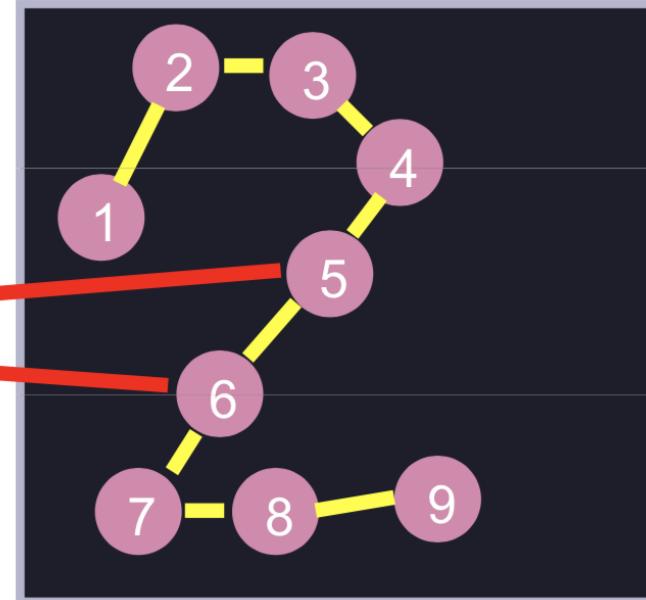
- Poate fi de tip many-to-many:

$M_2$  este împerecheat cu  $Q_2$  și  $Q_3$

# Alinierea traiectoriilor



$$M = (M_1, M_2, \dots, M_8).$$



$$Q = (Q_1, Q_2, \dots, Q_9).$$

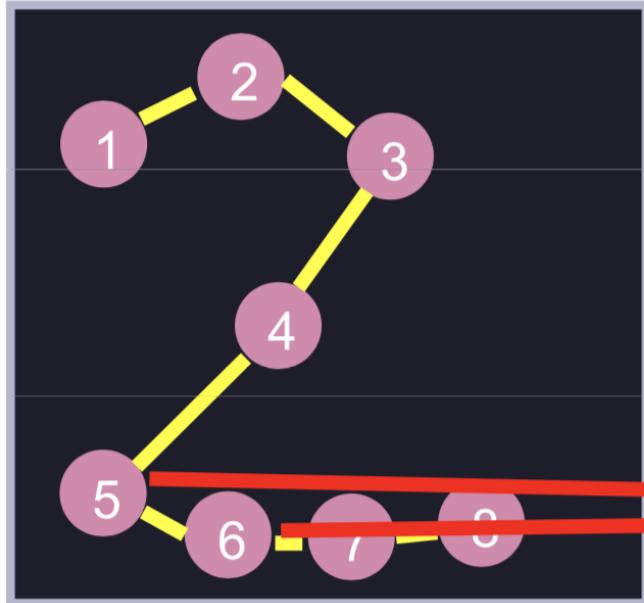
- Am putea să le aliniem:

$((1, 1), (2, 2), (2, 3), (3, 4), (4, 5), (4, 6), (5, 7), (6, 7), (7, 8), (8, 9))$

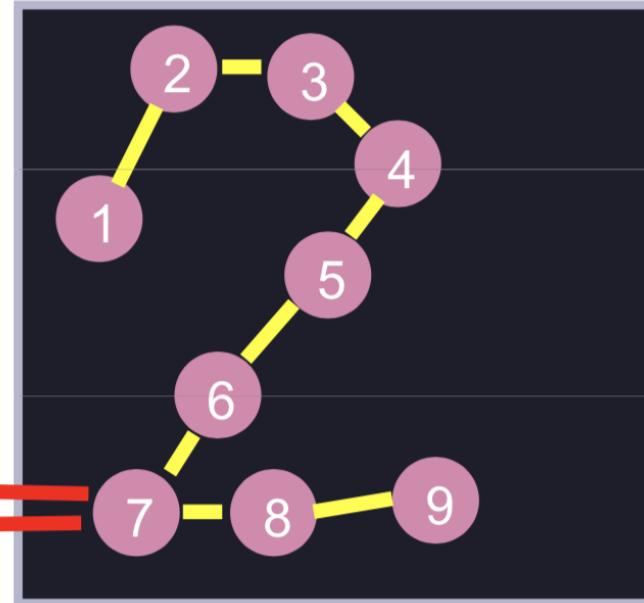
- Poate fi de tip many-to-many:

$M_4$  este împerecheat cu  $Q_5$  și  $Q_6$

# Alinierea traiectoriilor



$$M = (M_1, M_2, \dots, M_8).$$



$$Q = (Q_1, Q_2, \dots, Q_9).$$

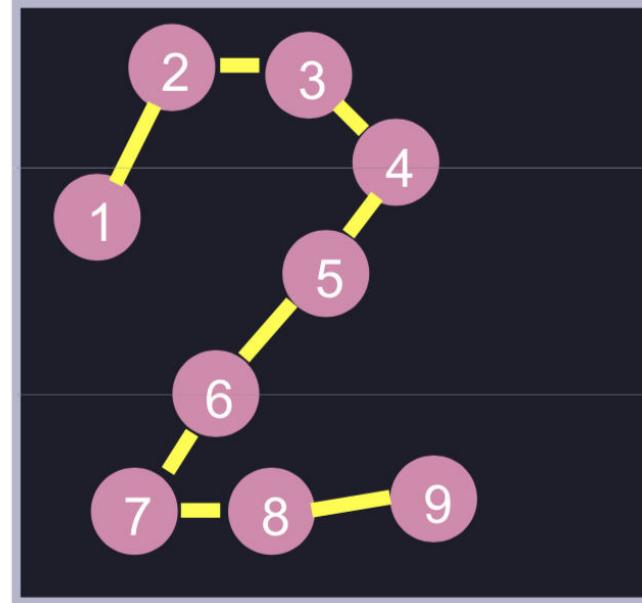
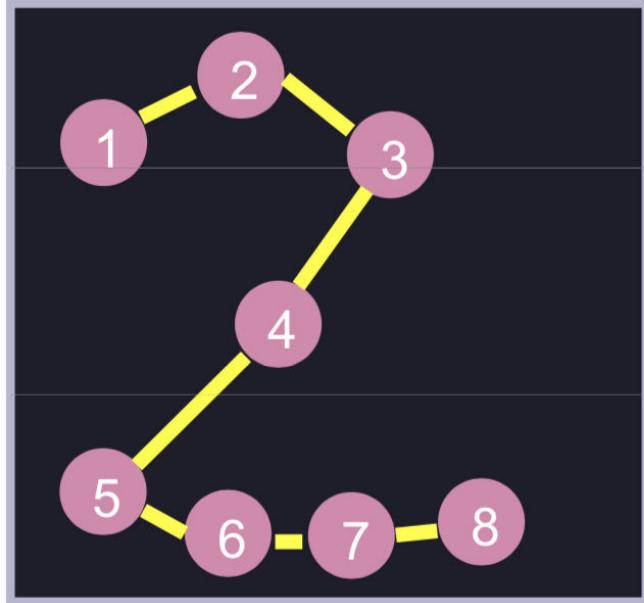
- Am putea să le aliniem:

$((1, 1), (2, 2), (2, 3), (3, 4), (4, 5), (4, 6), (5, 7), (6, 7), (7, 8), (8, 9))$

- Poate fi de tip many-to-many:

$M_5$  și  $M_6$  sunt împerecheate cu  $Q_7$

# Alinierea traiectoriilor

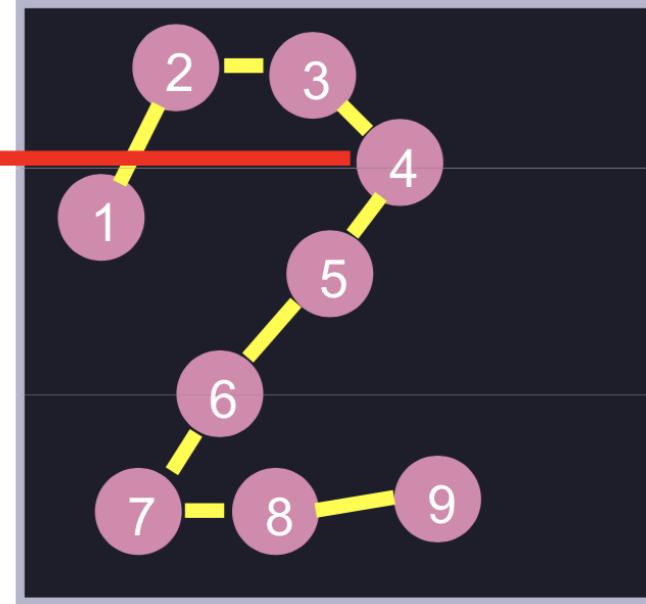
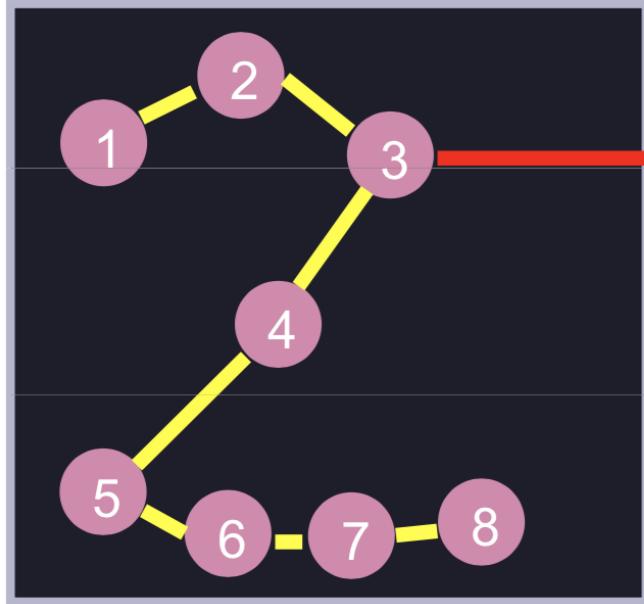


- Care este costul alinierii?

$$C = \text{cost}(s_1, t_2) + \text{cost}(s_2, t_2) + \dots + \text{cost}(s_m, t_n)$$

- Putem considera distanța Euclideană:  $\text{cost}(s_i, t_i) = d_{L_2}(s_i, t_i)$

# Alinierea traiectoriilor

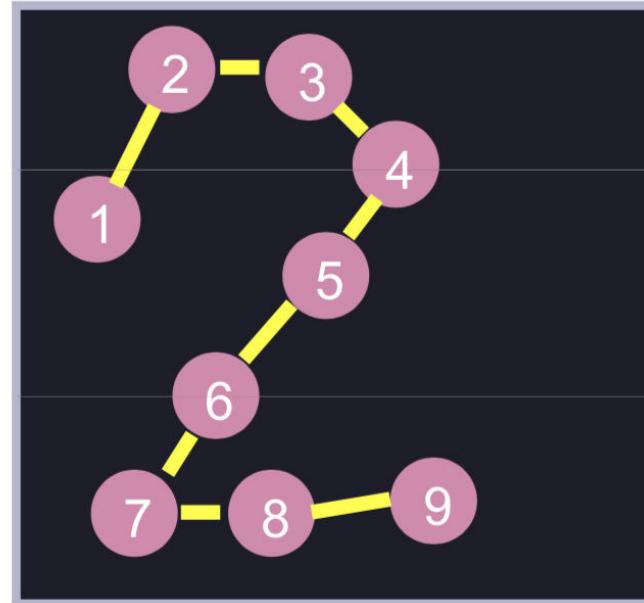
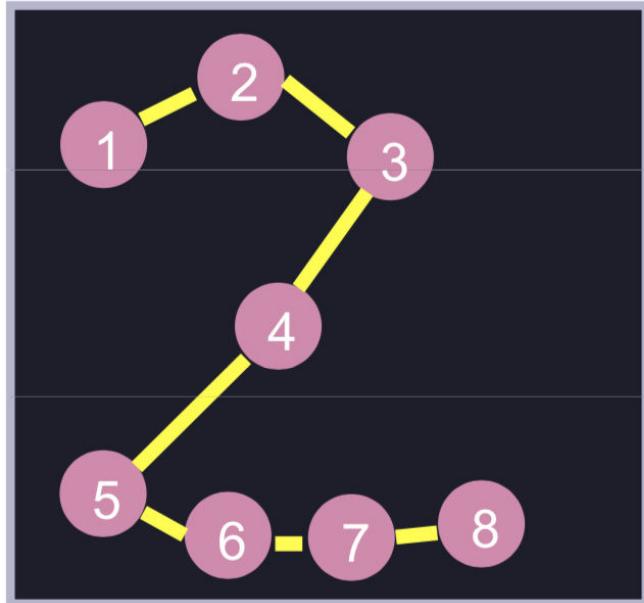


- Care este costul alinierii?

$$C = \text{cost}(s_1, t_1) + \text{cost}(s_2, t_2) + \dots + \text{cost}(s_m, t_n)$$

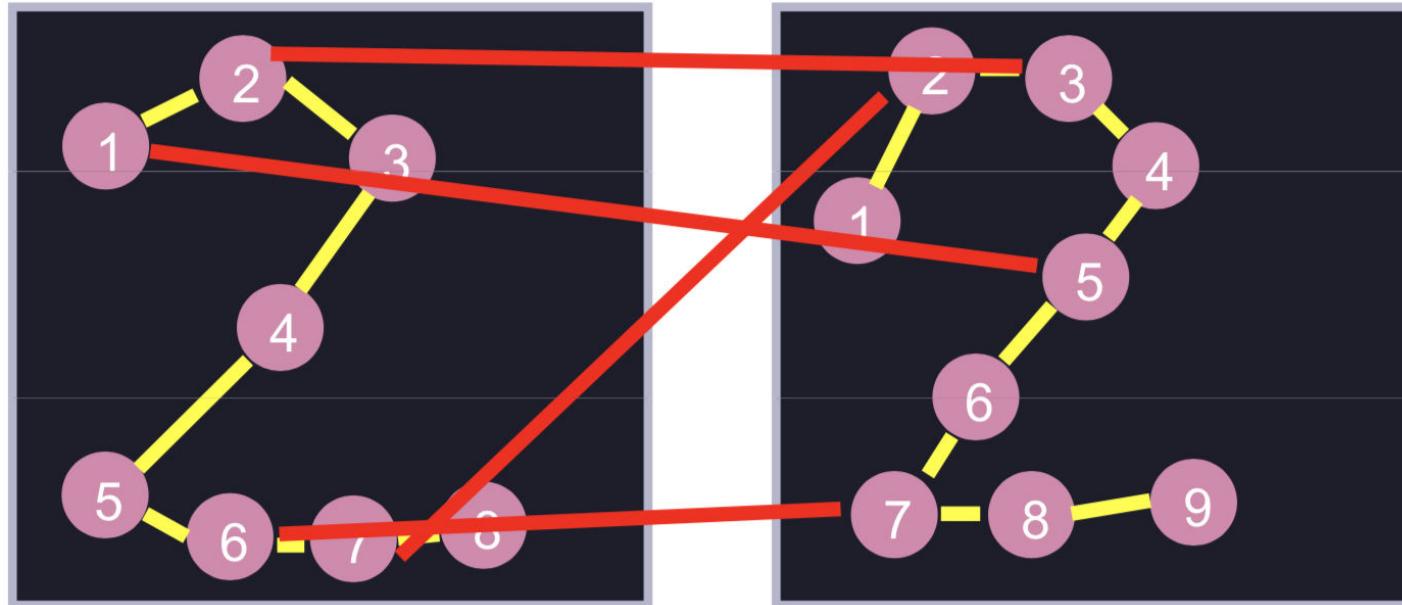
- Putem considera distanța Euclideană:  $\text{cost}(s_i, t_i) = d_{L_2}(s_i, t_i)$
- Exemplu:  $\text{cost}(3,4) = d_{L_2}(M_3, Q_4)$

# Alinierea traiectoriilor



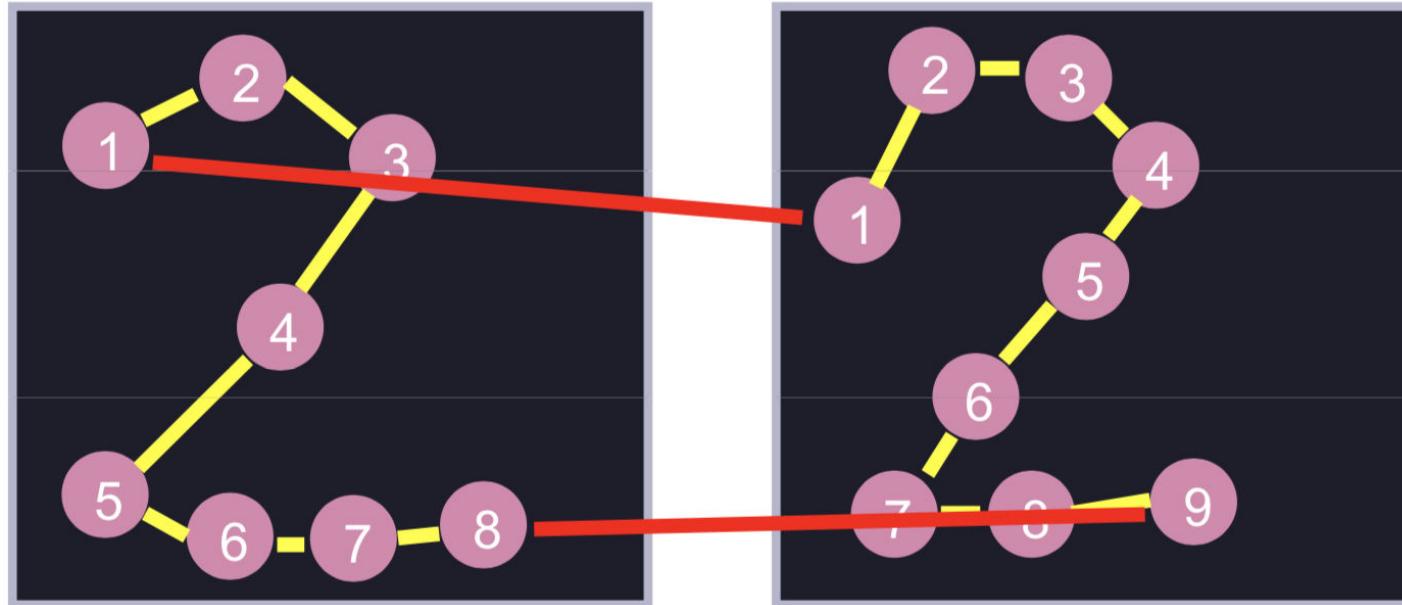
- Am putea să le aliniem:  
 $((1, 1), (2, 2), (2, 3), (3, 4), (4, 5), (4, 6), (5, 7), (6, 7), (7, 8), (8, 9))$
- Care sunt regulile alinierii?

# Alinierea traiectoriilor



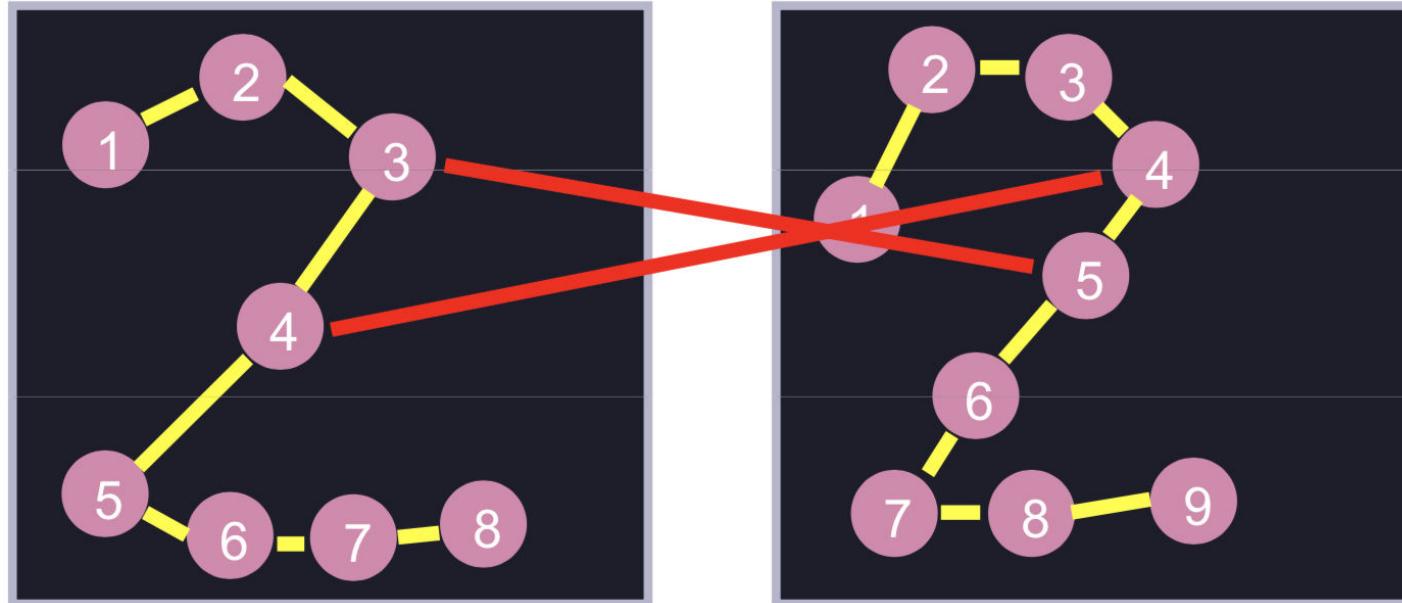
- Am putea să le aliniem:  
((1, 1), (2, 2), (2, 3), (3, 4), (4, 5), (4, 6), (5, 7), (6, 7), (7, 8), (8, 9))
- Care sunt regulile alinierii?
  - Este legală aliniera ((1, 5), (2, 3), (6, 7), (7, 1))?
  - Decizia depinde de aplicație

# Regulile alinierii pentru DTW



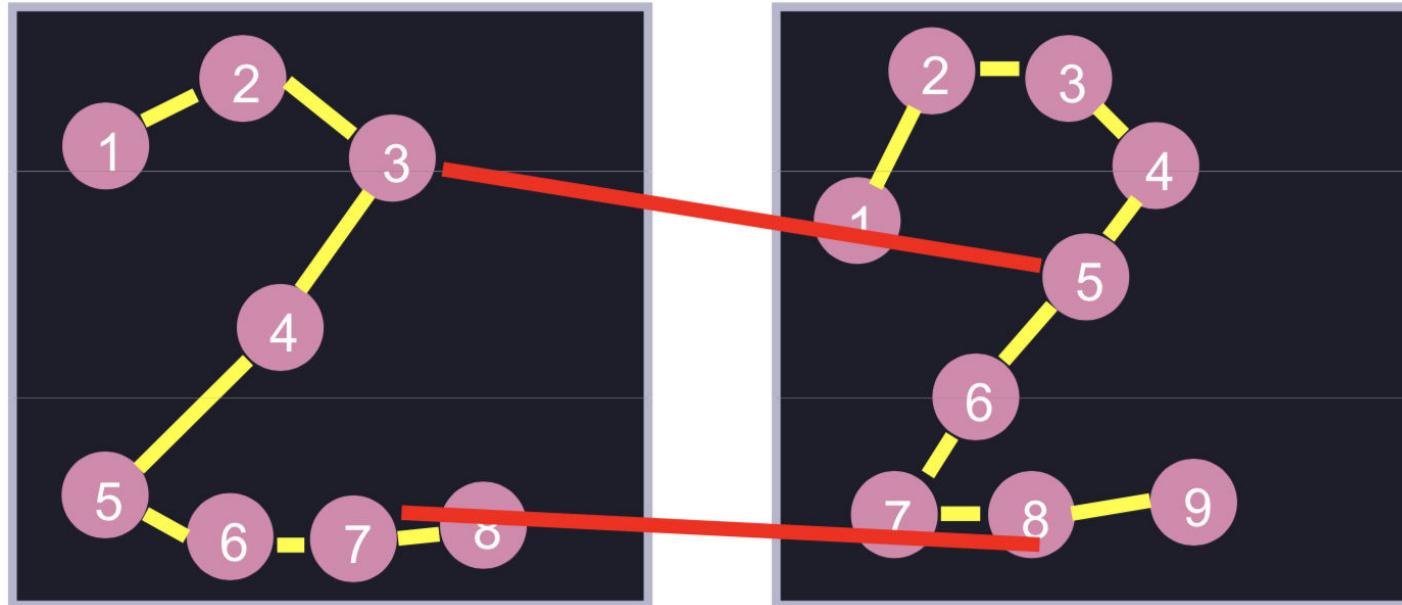
- Am putea să le aliniem:  
 $((1, 1), (2, 2), (2, 3), (3, 4), (4, 5), (4, 6), (5, 7), (6, 7), (7, 8), (8, 9))$
- Regulile pentru DTW: limitele
  - Primele elemente formează prima pereche:  $(s_1, t_1)$
  - Ultimele elemente formează ultima pereche:  $(s_m, t_n)$

# Regulile alinierii pentru DTW



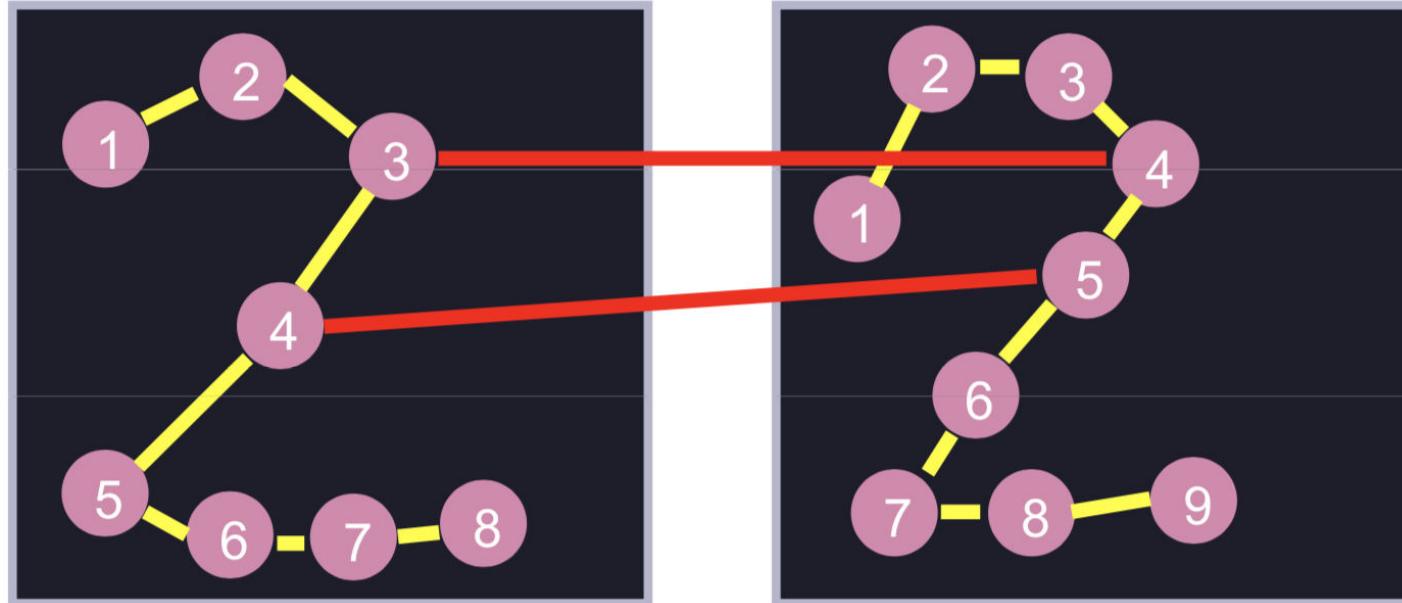
- Exemplu de aliniere care încalcă monotonia:  
 $(\dots, (3, 5), (4, 4), \dots)$
- Regulile pentru DTW: monotonie (nu ne putem întoarce)
  - $0 \leq (s_{i+1} - s_i)$
  - $0 \leq (t_{i+1} - t_i)$

# Regulile alinierii pentru DTW



- Exemplu de aliniere care încalcă continuitatea:  
 $(..., (3, 5), (7, 8), ...)$
- Regulile pentru DTW: continuitatea (nu putem sări elemente)
  - $(s_{i+1} - s_i) \leq 1$
  - $(t_{i+1} - t_i) \leq 1$

# Regulile alinierii pentru DTW



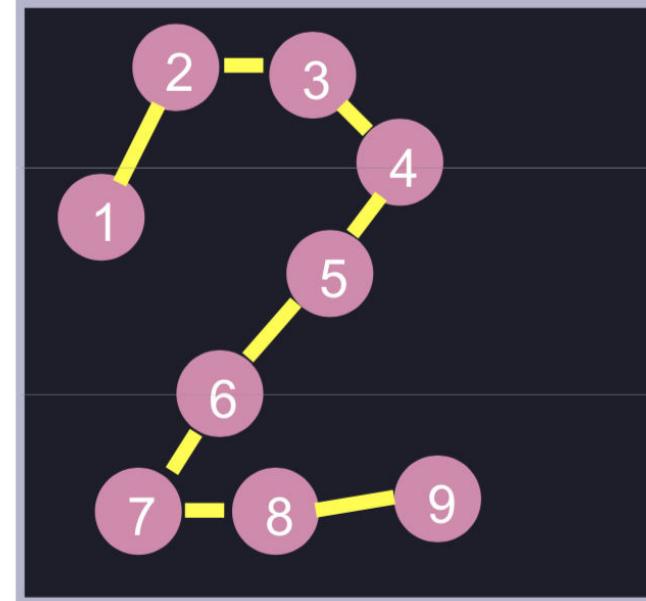
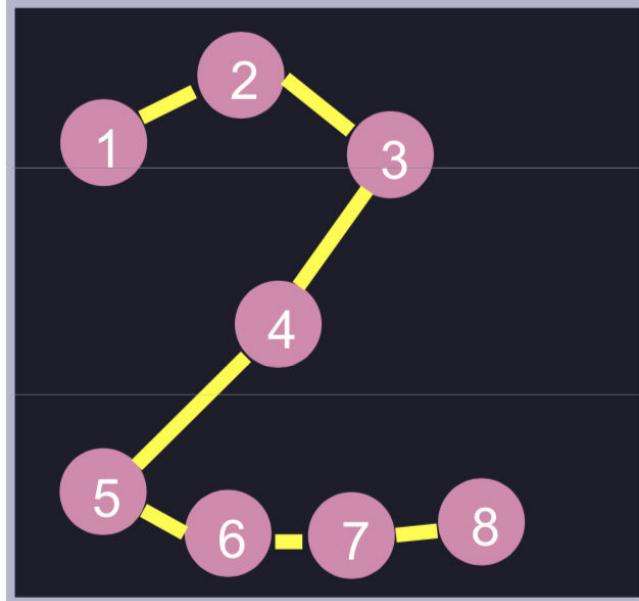
- Aliniere validă:

$((1, 1), (2, 2), (2, 3), (3, 4), (4, 5), (4, 6), (5, 7), (6, 7), (7, 8), (8, 9))$

- Regulile pentru DTW: monotonia și continuitatea

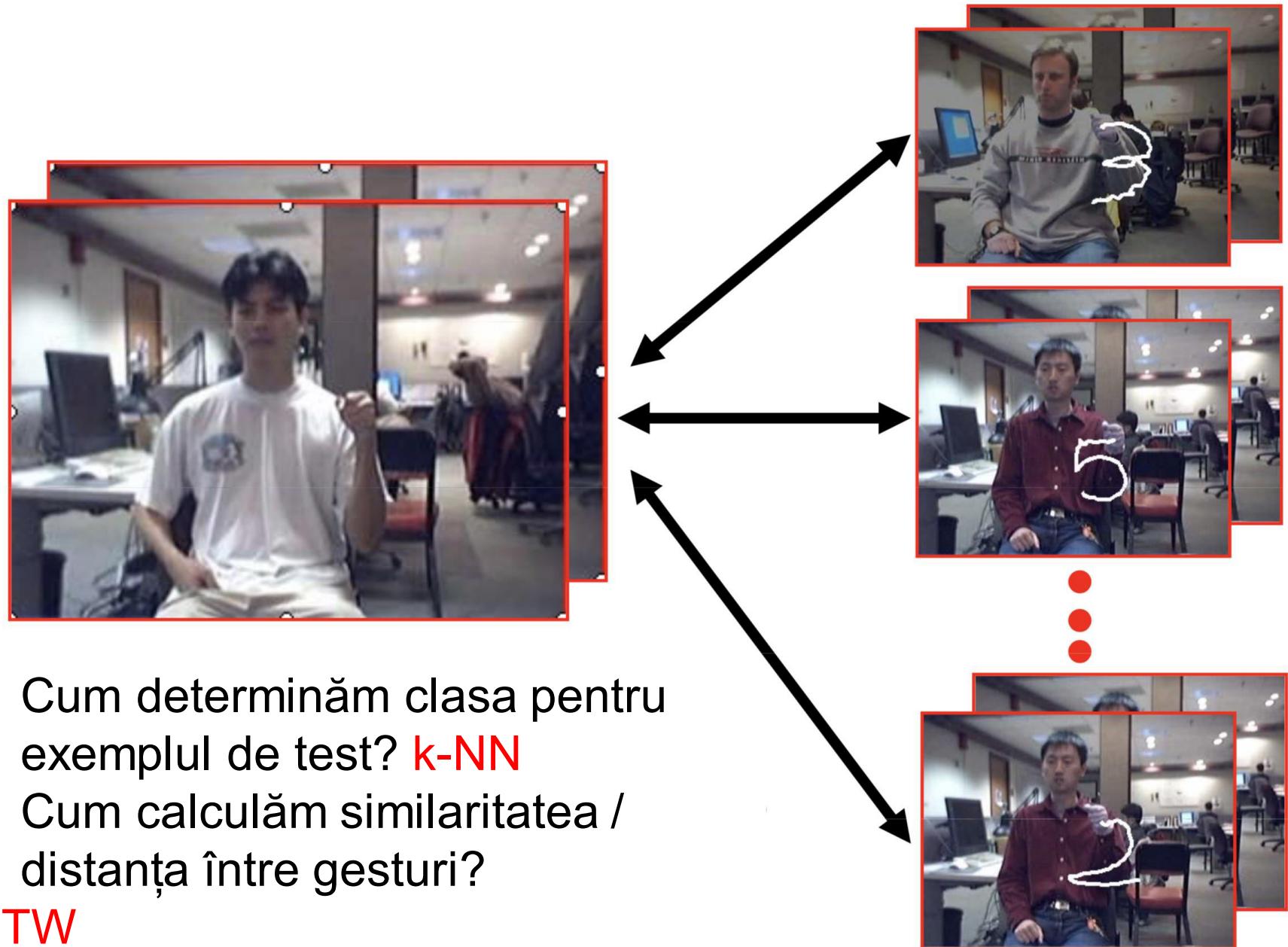
- $0 \leq (s_{i+1} - s_i) \leq 1$
- $0 \leq (t_{i+1} - t_i) \leq 1$

# Dynamic Time Warping

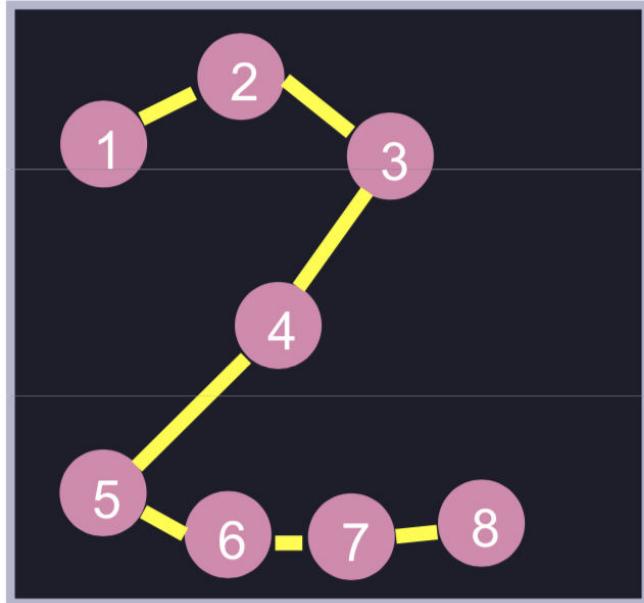


- DTW este o distanță între secvențe de puncte
- Distanța DTW între secvențe temporale este data de costul alinierii optime dintre cele două traекторii
- Alinierea trebuie să respecte regulile definite mai devreme

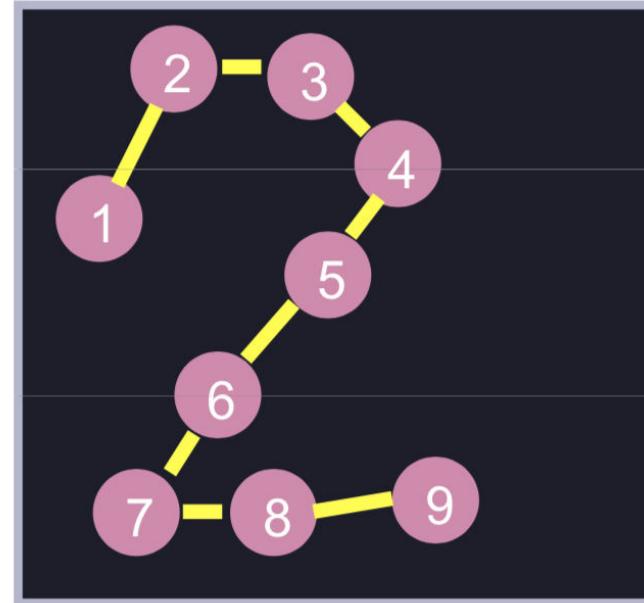
# Recunoașterea gesturilor



# Calcularea distanței DTW (Edit)



M



Q

- Exemplu de antrenare:  $M = (M_1, M_2, \dots, M_8)$
- Exemplu de testare:  $Q = (Q_1, Q_2, \dots, Q_9)$
- Fiecare  $M_i$  și  $Q_j$  poate fi, de exemplu, locația spațială (2D) a mâinii

# Calcularea distanței DTW (Edit)

- Exemplu de antrenare:  $M = (M_1, M_2, \dots, M_8)$
- Exemplu de testare:  $Q = (Q_1, Q_2, \dots, Q_9)$
- Dorim să obținem alinierea optimă dintre  $M$  și  $Q$
- Implementare bazată pe programare dinamică:
  - Împărțim problema într-o secvență de mai multe probleme mici  $P(i,j)$ , pe care le rezolvăm printr-o relație recursivă
  - Problema  $P(i,j)$ : găsirea alinierii optime dintre  $(M_1, M_2, \dots, M_i)$  și  $(Q_1, Q_2, \dots, Q_j)$

# Calcularea distanței DTW (Edit)

- Rezolvarea problemei  $P(1, j)$ :
  - Alinierea optimă este:  $((1, 1), (1, 2), \dots, (1, j))$
- Rezolvarea problemei  $P(i, 1)$ :
  - Alinierea optimă este:  $((1, 1), (2, 1), \dots, (i, 1))$
- Rezolvarea problemei  $P(i, j)$ :
  - Alegem cea mai bună soluție dintre:  
 $(i, j-1)$ ,  $(i-1, j)$ ,  $(i-1, j-1)$
  - Adăugăm la soluția aleasă perechea  $(i, j)$

# Algoritmul DTW

- Input:

- Exemplu de antrenare:  $M = [M_1, M_2, \dots, M_8]$
- Exemplu de testare:  $Q = [Q_1, Q_2, \dots, Q_9]$

- Algoritm:

$C = \text{zeros}(m, n)$

$C[1,1] = \text{cost}(M_1, Q_1)$

for  $i = 2$  to  $m$ :

$C[i,1] = C[i-1,1] + \text{cost}(M_i, Q_1)$

for  $j = 2$  to  $n$ :

$C[1,j] = C[1,j-1] + \text{cost}(M_1, Q_j)$

for  $i = 2$  to  $m$ :

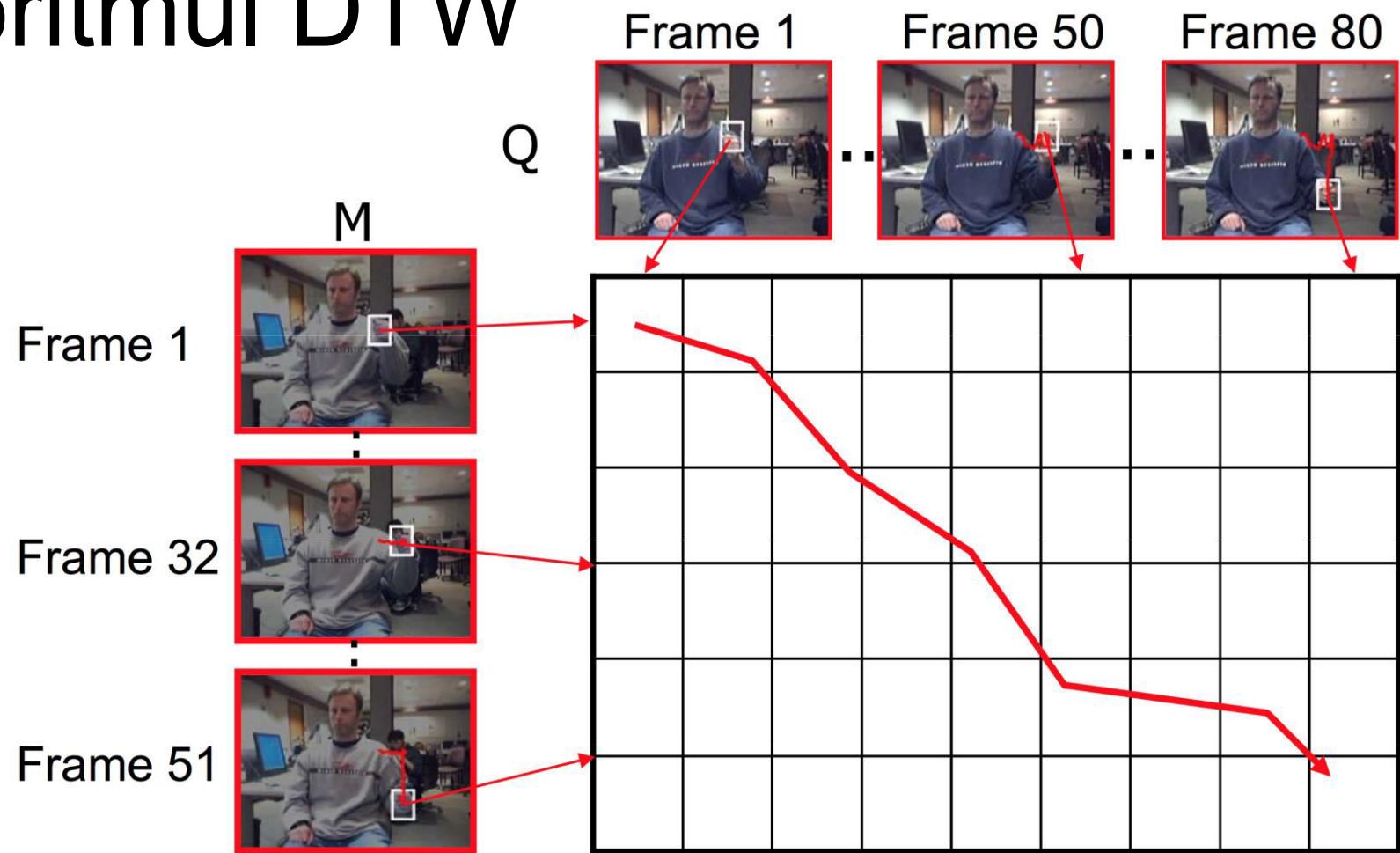
for  $j = 2$  to  $n$ :

$C[i,j] = \text{cost}(M_i, Q_j) + \min(C[i-1,j], C[i,j-1], C[i-1,j-1])$

- Returnează:

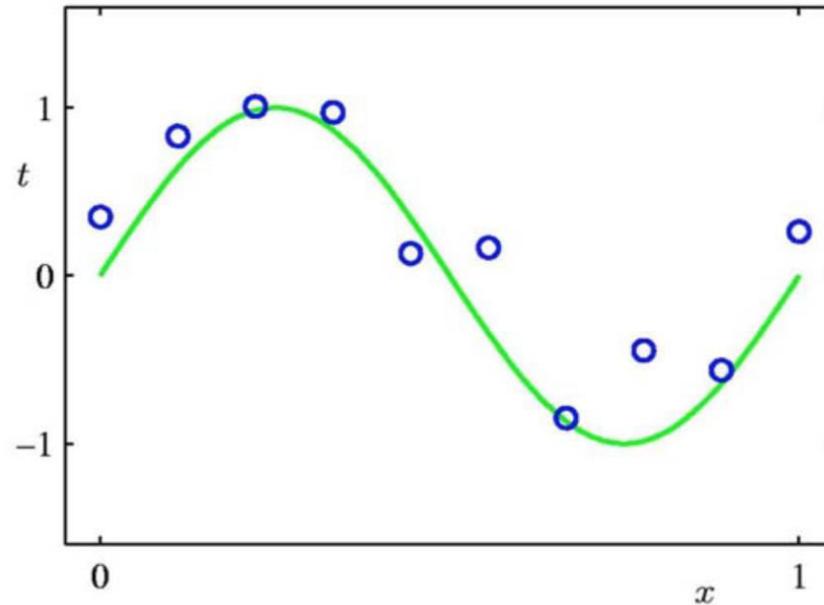
$C[m,n]$

# Algoritmul DTW



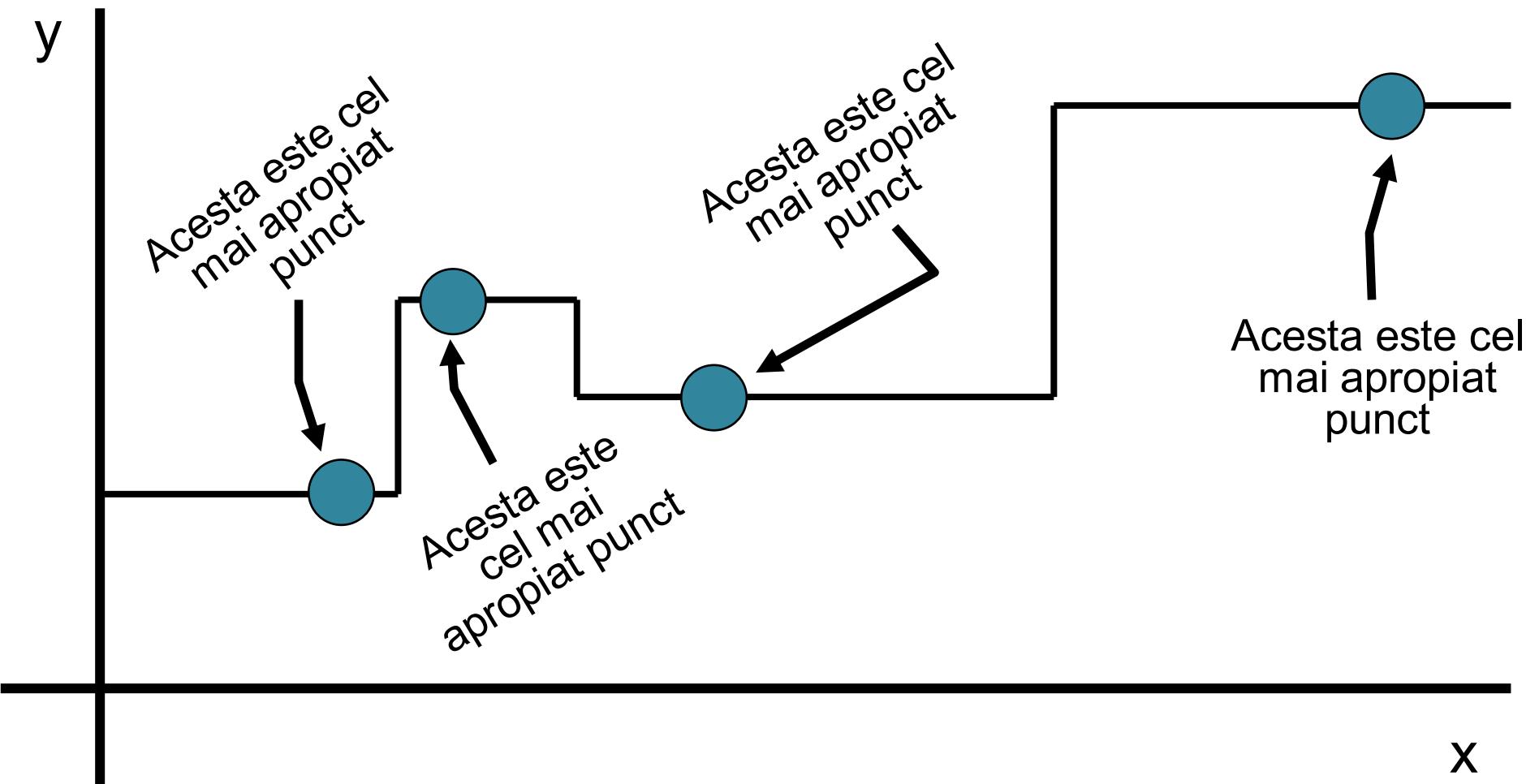
- Pentru fiecare celulă  $(i, j)$ :
  - Calculăm alinierea optimă dintre  $M[1:i]$  și  $Q[1:j]$
  - Răspunsul depinde doar de  $(i-1, j)$ ,  $(i, j-1)$ ,  $(i-1, j-1)$
  - Timpul: liniar cu mărimea matricii, pătratic cu lungimile secvențelor

# Regresie din exemple etichetate

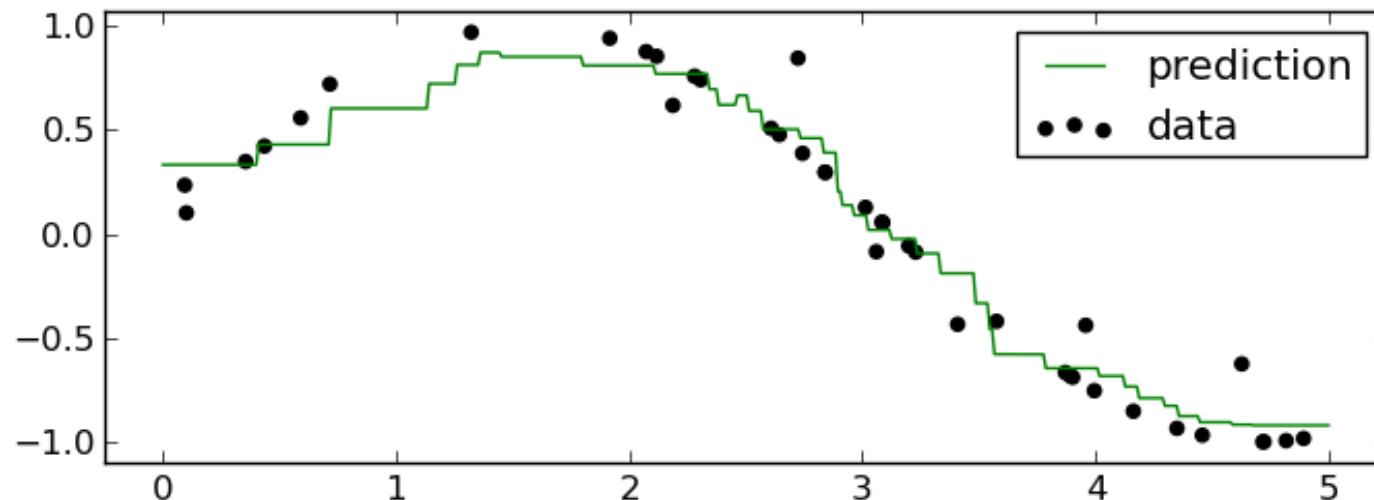


- Presupunem că avem un set de  $N$  exemple de antrenare:  
 $(x_1, \dots, x_N)$  and  $(y_1, \dots, y_N)$ ,  $x_i, y_i \in \mathbb{R}$
- Problema regresiei constă în estimarea funcției  $g(x)$  a.î.:  
$$g(x_i) = y_i$$

# 1-NN pentru probleme de regresie



# k-NN pentru probleme de regresie



- Algoritmul de regresie k-NN:
  - 1) Pentru fiecare exemplu de test  $x$ , găsim cei mai apropiati  $k$  vecini și etichetele lor
  - 2) Output-ul este media etichetelor celor  $k$  vecini

$$f(x) = \frac{1}{K} \sum_{i=1}^K y_i$$

# Avantaje și proprietăți ale modelului k-NN

- Modelul k-NN este un model simplu
- Poate fi aplicat pentru probleme cu mai multe clase
- Suprafața de decizie este neliniară
- Calitatea rezultatelor crește atunci când avem mai multe date de antrenare
- Avem un singur parametru care trebuie ajustat ( $k$ )
- Eroarea de clasificare pe antrenare crește odată cu  $k$ , dar suprafața de decizie devine mai netedă:
  - Metodă de regularizare care crește capacitatea de generalizare

# Dezavantaje ale modelului k-NN

- Ce înseamnă cel mai apropiat? Trebuie să definim o distanță
- Este distanța Euclidiană cea mai bună alegere?
- Costul computațional este ridicat: trebuie să stocăm și să parcurgem întreg setul de antrenare în timpul testării
- Soluții alternative pentru evitarea costului ridicat:
  - Partiționarea spațiului folosind arbori k-d
  - Locality sensitive hashing
- Suferă de “curse of dimensionality”

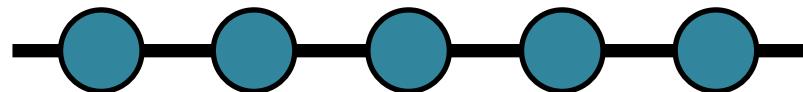
**“Blestemul dimensionalității”**  
**(Curse of dimensionality)**

# Blestemul dimensionalității

- În învățarea automată, folosim deseori date de dimensiuni mari
- Exemplu:
  - Dacă analizăm imagini cu tonuri de gri de dimensiune 200x200 de pixeli, atunci lucrăm într-un spațiu cu 40.000 de dimensiuni.
  - Dacă imaginile sunt color (reprezentate în spațiul RGB), dimensionalitatea spațiului crește la 120.000 de dimensiuni

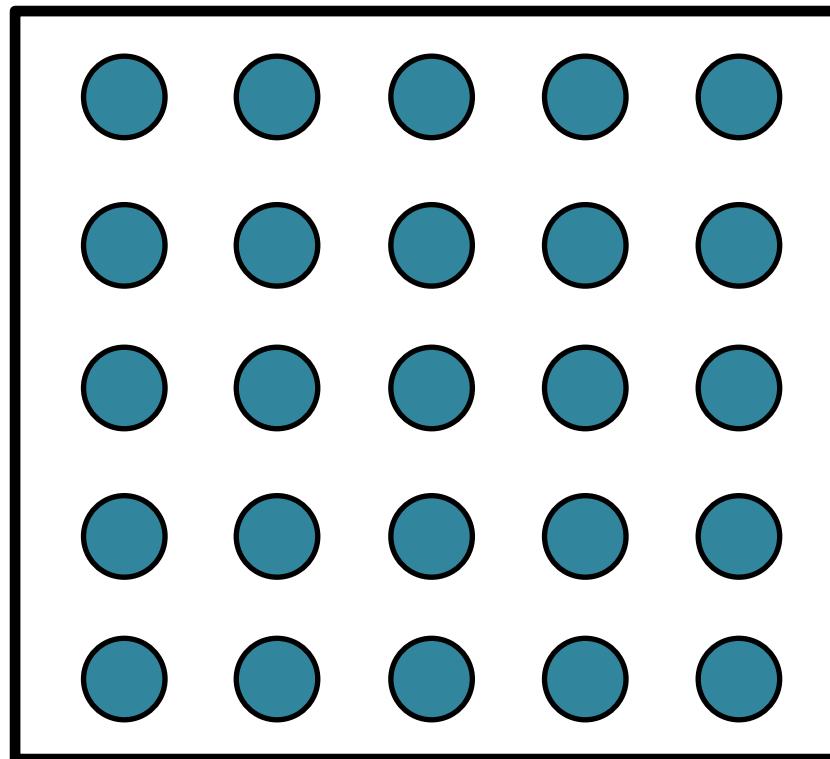
# Blestemul dimensionalității

- Pentru a “umple” un spațiu 1D (de exemplu  $\mathbb{R}^1$ ) avem nevoie de 5 puncte:



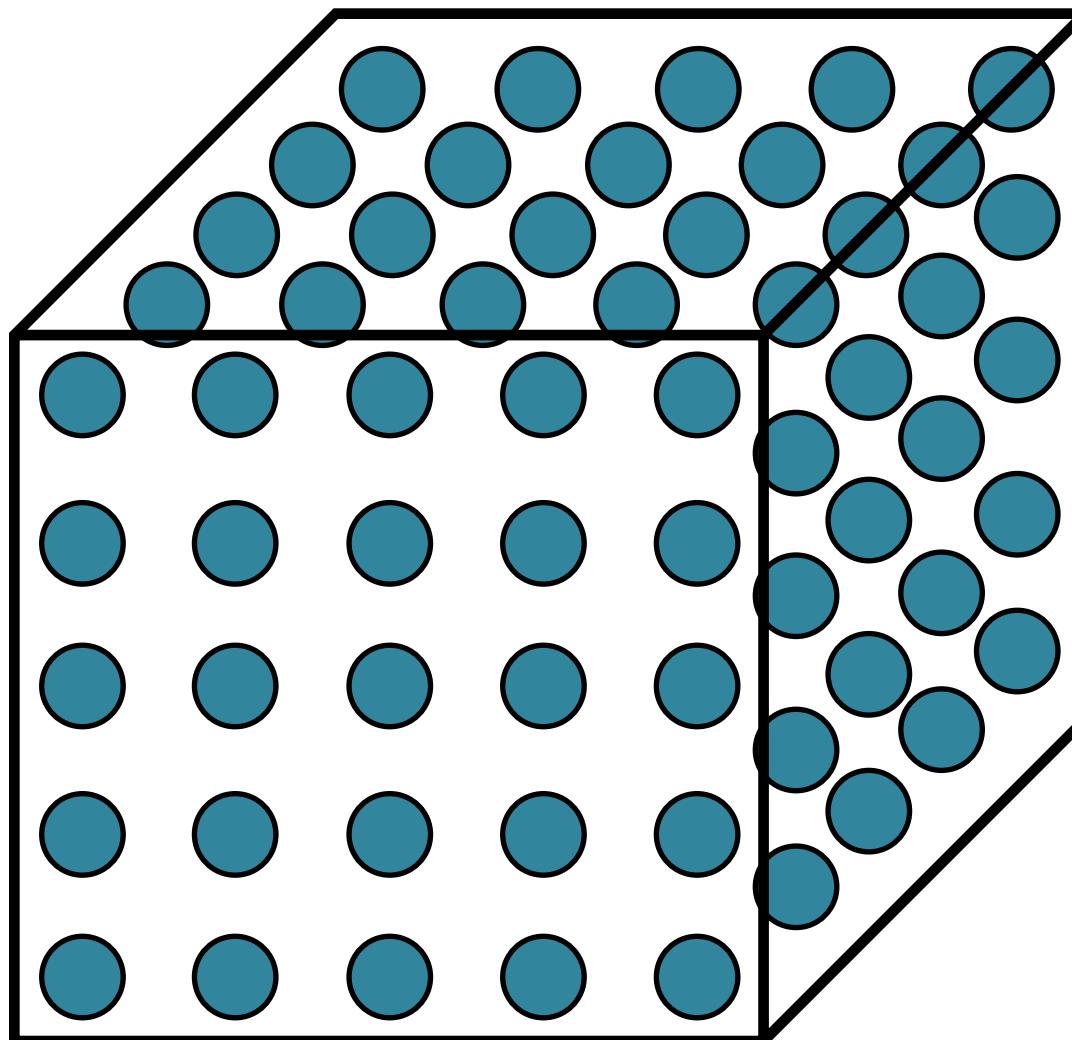
# Blestemul dimensionalității

- Pentru a “umple” un spațiu 2D (de exemplu  $\mathbb{R}^2$ ) avem nevoie de 25 puncte:



# Blestemul dimensionalității

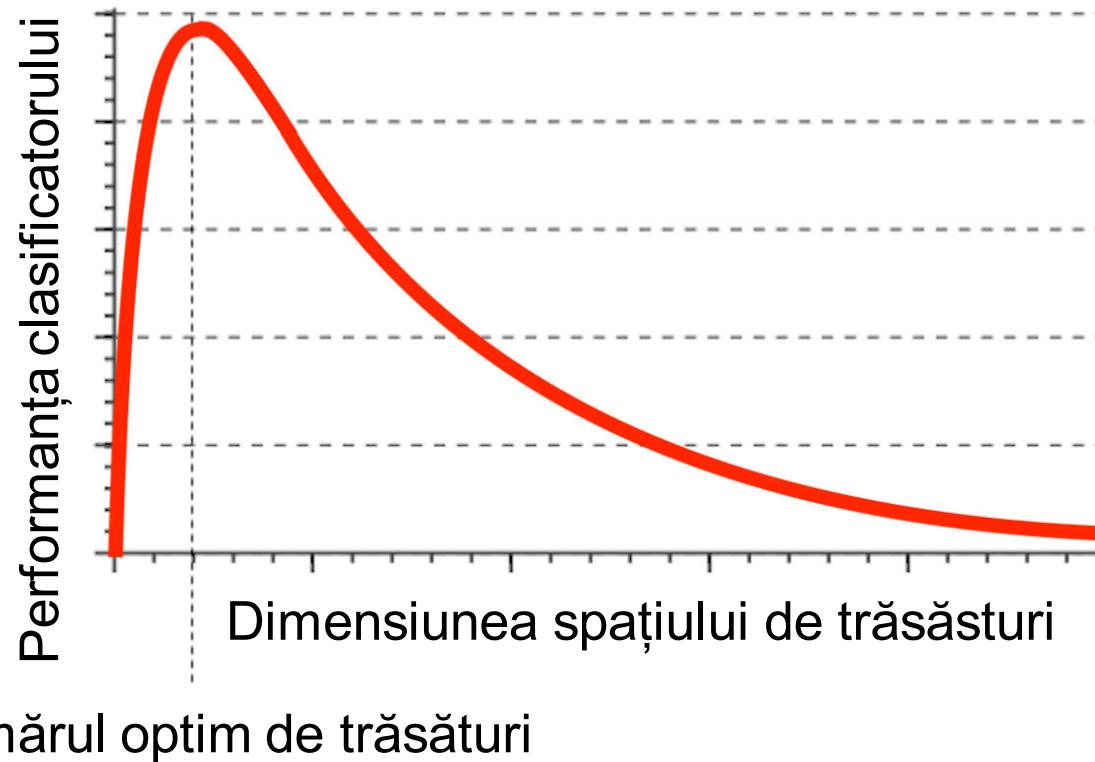
- Pentru a “umple” un spațiu 3D (de exemplu  $\mathbb{R}^3$ ) avem nevoie de 125 puncte:



# Blestemul dimensionalității

- Pentru a “umple” un spațiu nD (de exemplu  $\mathbb{R}^n$ ) avem nevoie de un număr exponențial de puncte
- Dacă avem un număr mare de caracteristici care descriu datele, atunci sistemul are nevoie de foarte multe exemple de antrenare pentru învăța un model care să generalizeze
- De cele mai multe ori aceste date nu sunt disponibile în practică

# Fenomenul Hughes



- Fenomenul Hughes arată că, pe măsură ce numărul de caracteristici crește, performanța clasificatorului crește până când ajungem la numărul optim de trăsături
- Adăugarea mai multor caracteristici păstrând dimensiunea setului de antrenare degradează performanța clasificatorului

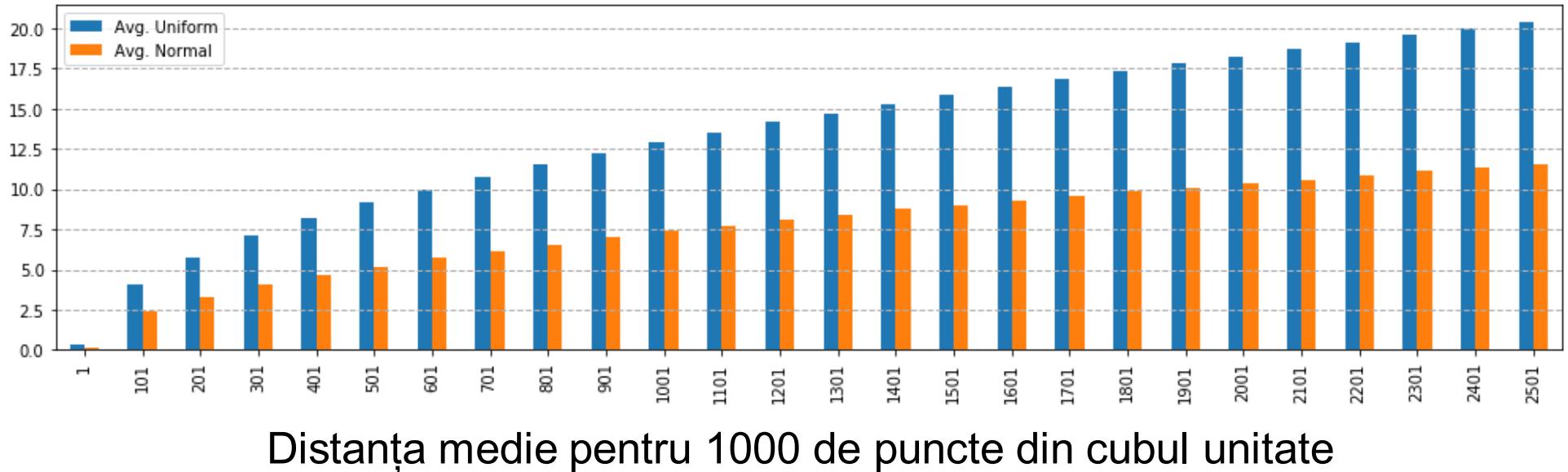
# Blestemul dimensionalității

- Creșterea numărului de dimensiuni al unui spațiu de caracteristici Euclidian, implică adăugarea de termeni pozitivi în calculul distanței Euclidiene:

$$(x, y) = \sqrt{(x_1 - y_1)^2 + \cdots + (x_n - y_n)^2}$$

- Cu alte cuvinte, deoarece numărul de trăsături crește pentru un număr fix de exemple, spațiul de caracteristici devine din ce în ce mai rar (mai puțin dens)

# Blestemul dimensionalității



- Figura arată că, odată cu creșterea dimensiunilor, distanța medie crește rapid
- Prin urmare, cu cât sunt mai multe dimensiuni, cu atât sunt necesare mai multe date pentru a depăși blestemul dimensionalității!
- Atunci când distanța dintre observații crește, învățarea automată devine mult mai dificilă, deoarece scade probabilitatea de a găsi exemple de antrenare cu adevărat similare cu cele de test

# Metode kernel. Regresia Ridge. Mașini cu Vectori Suport.

Prof. Dr. Radu Ionescu

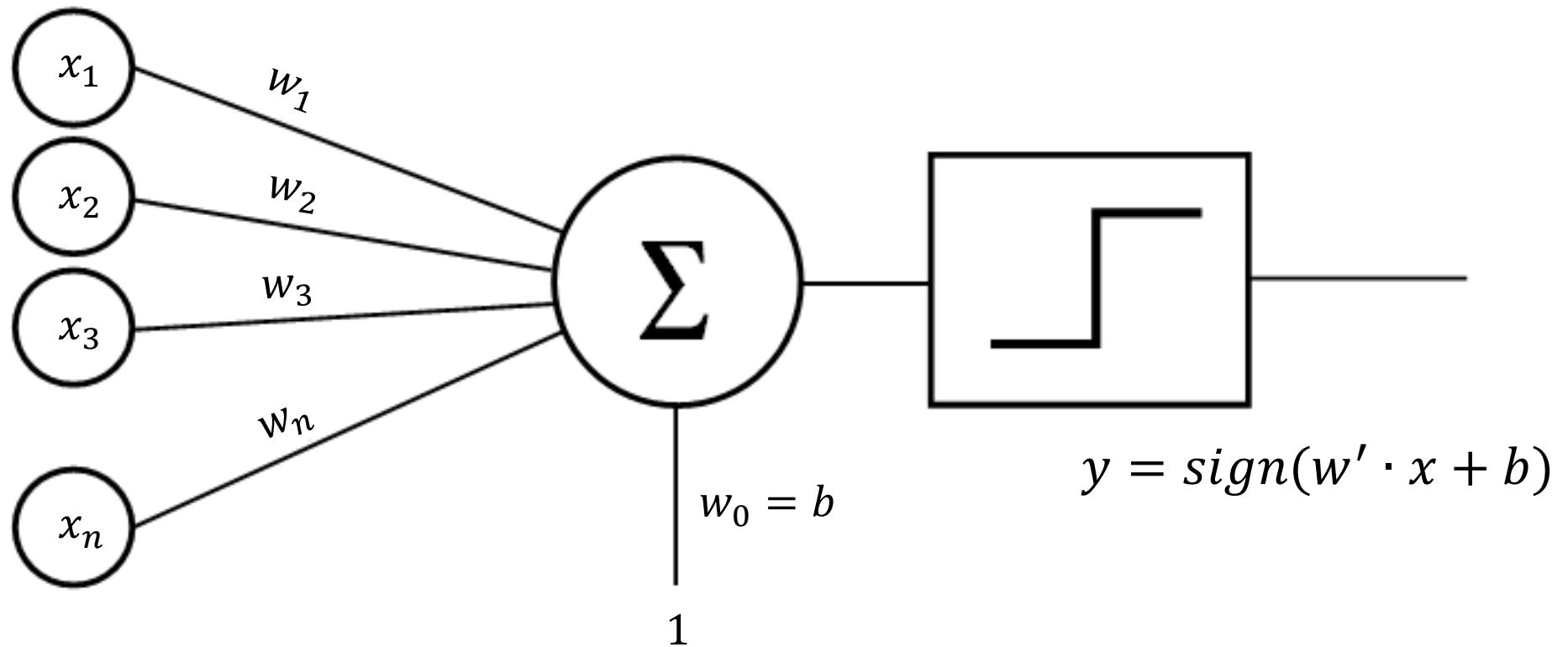
[raducu.ionescu@gmail.com](mailto:raducu.ionescu@gmail.com)

Facultatea de Matematică și Informatică  
Universitatea din București

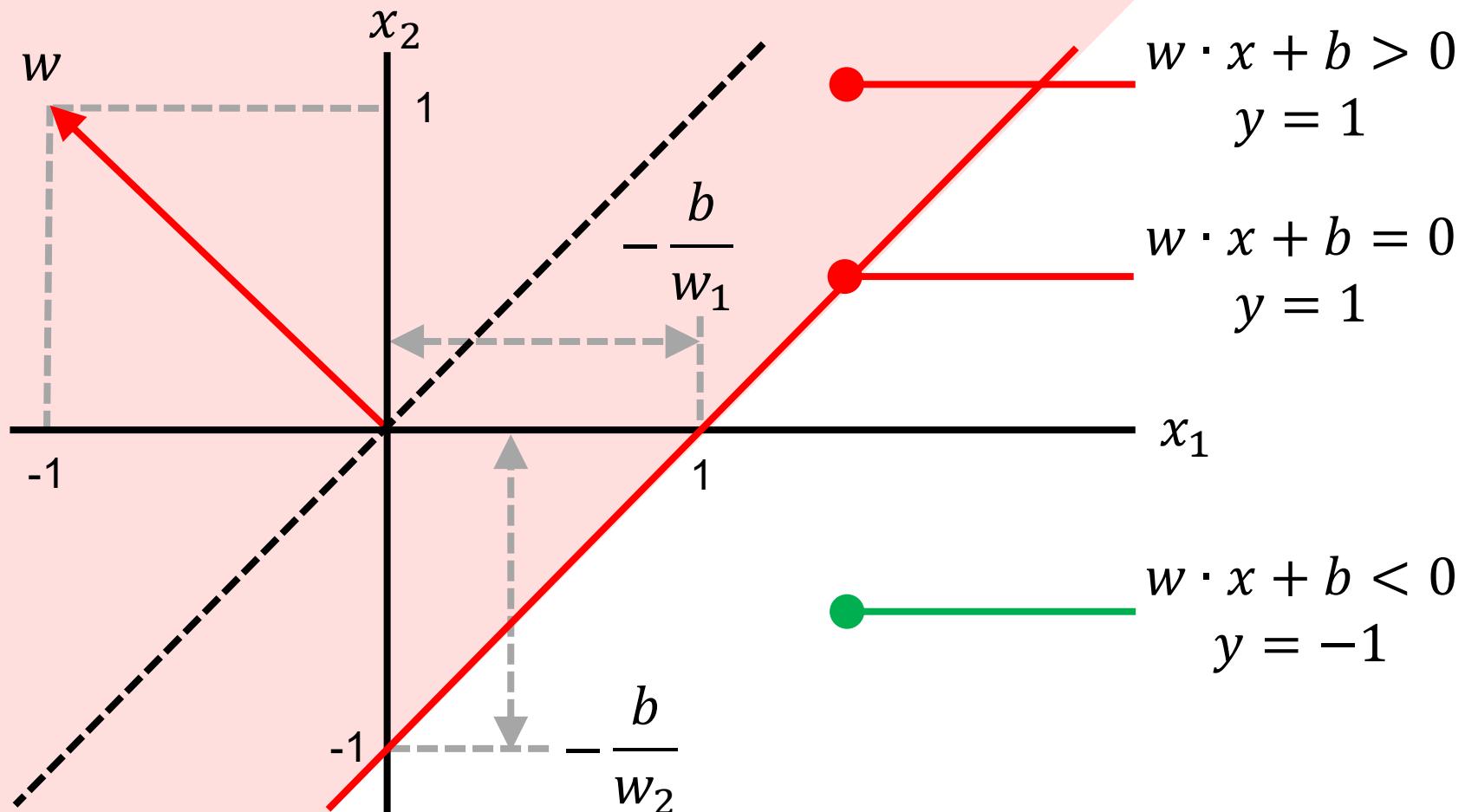
# Evoluția metodelor de învățare automată

- Anii 1950: este introdus perceptronul (Rosenblatt, 1957)
- Anii 1980: este introdus algoritmul de backpropagare pentru rețelele neuronale multistrat (Hinton, 1986)
- Anii 1990: apar metodele kernel (nucleu)

# Perceptronul



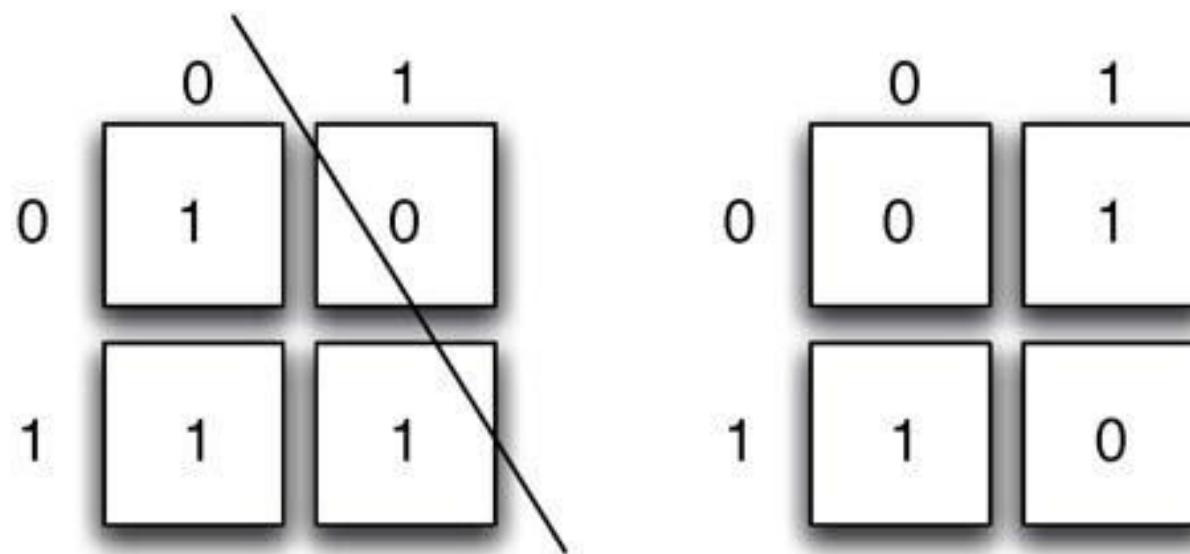
# Granita de separare liniară



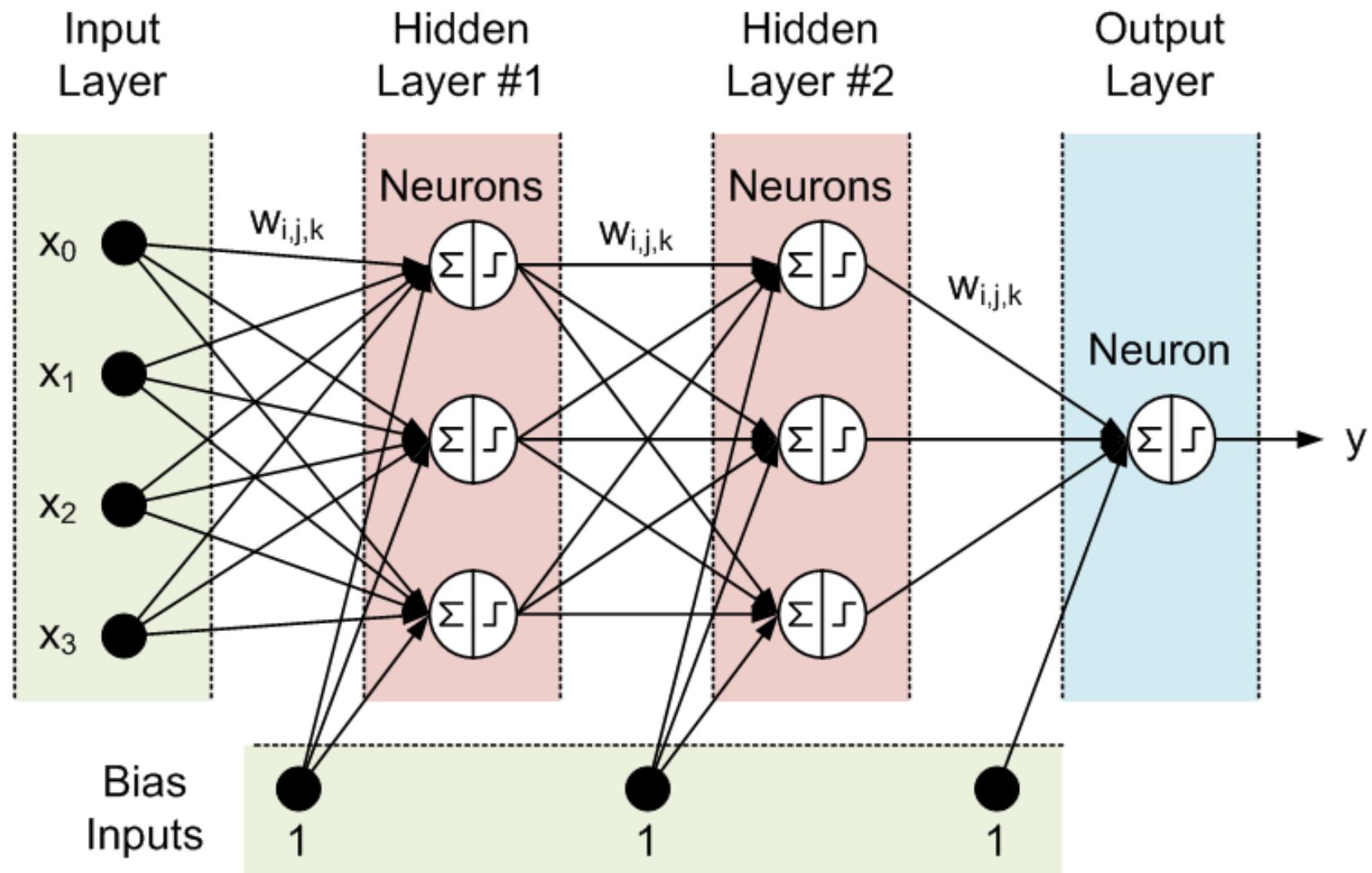
Where  $w_1 = -1, w_2 = 1, b = 1$

# XOR (Minsky și Papert, 1969)

- O metodă de clasificare liniară nu poate învăța funcția XOR

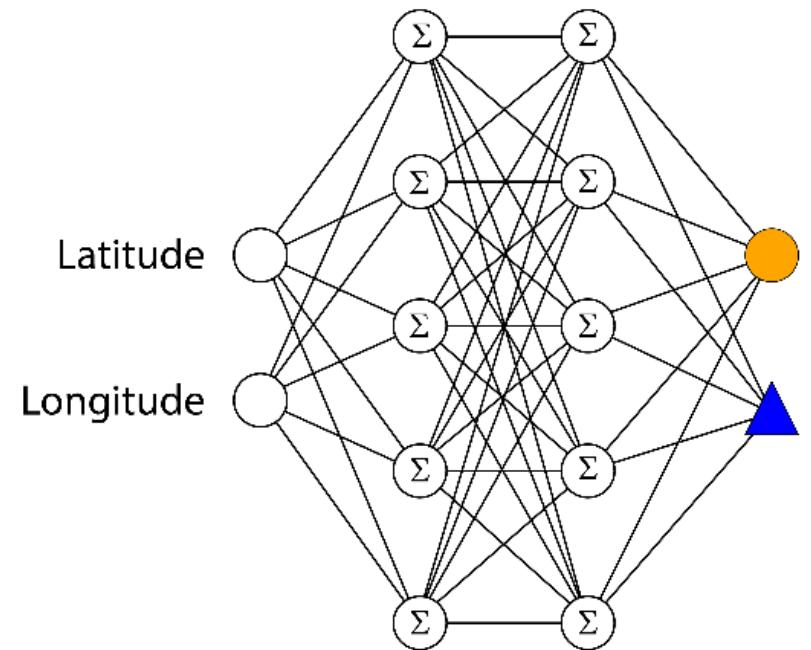
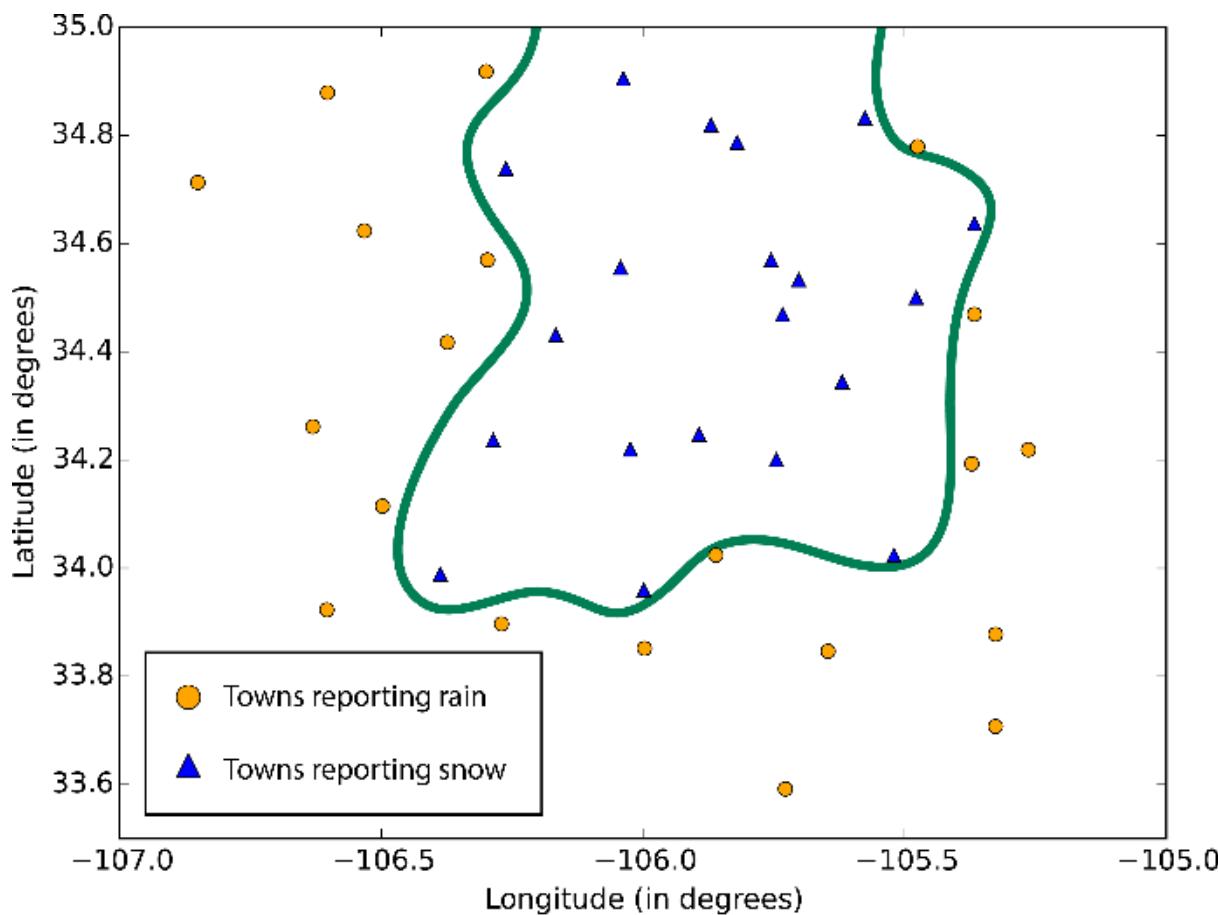


# Soluția 1: Rețele neuronale



# Soluția 1: Rețele neuronale

- Granița de decizie devine non-liniară

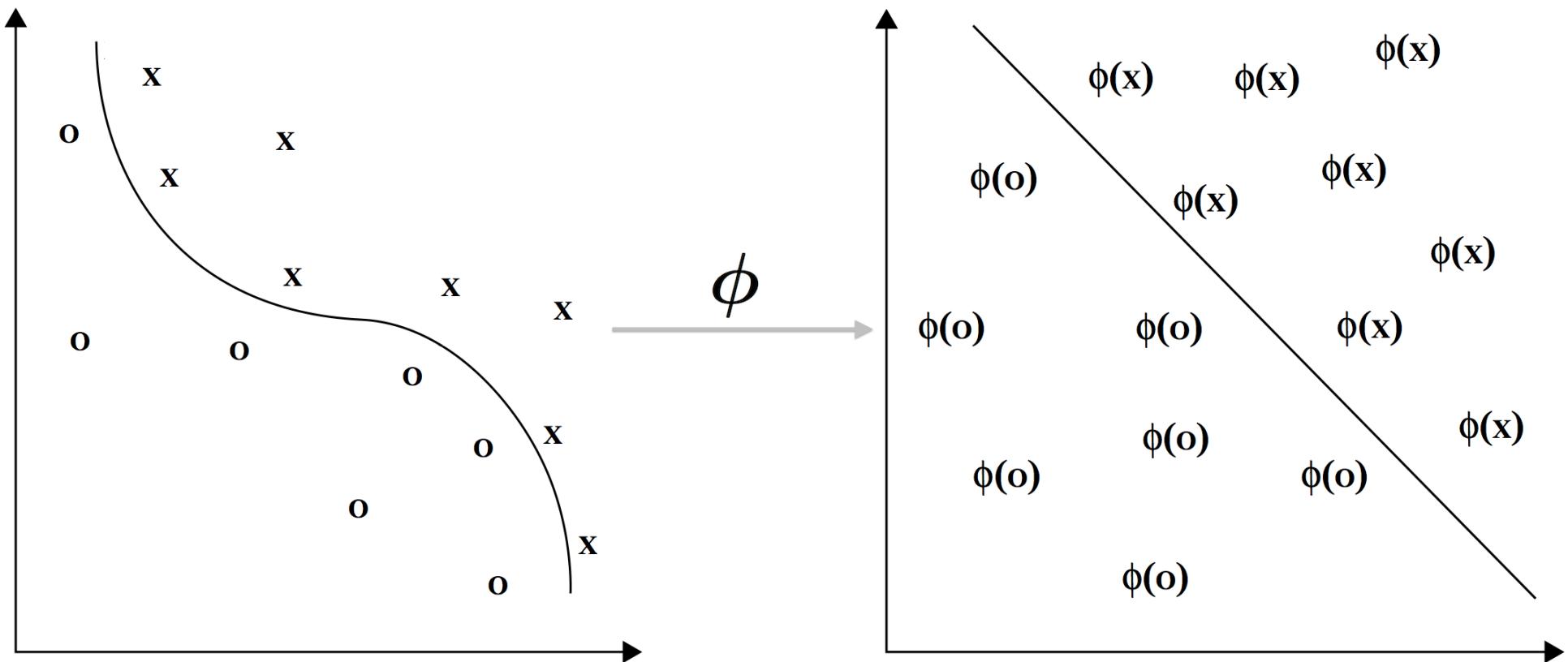


# Soluția 2: Metode kernel

- Metodele kernel funcționează prin următorii doi pași:
  - 1. Datele sunt scufundate într-un spațiu (Hilbert) cu mai multe dimensiuni
  - 2. Relațiile liniare sunt căutate în acest spațiu
- Scufundarea datelor se realizează implicit, prin specificarea produsului scalar între exemple

# Scufundarea datelor cu o funcție kernel

- Relațiile neliniare din spațiul original sunt transformate în relații liniare prin scufundare



# Metode kernel

- Algoritmii sunt implementați (în forma duală) astfel încât coordonatele punctelor scufundate nu sunt necesare, fiind suficientă specificarea produsului scalar între perechi de puncte
- “Kernel trick”: Produsul scalar poate fi înlocuit cu orice funcție de similaritate, numită și funcție kernel (funcție nucleu)

## Forma primală

Features:  $f_1, f_2, f_3, f_4, f_5, f_6, f_7$

Train samples:  
 $x_1, x_2, x_3, x_4$

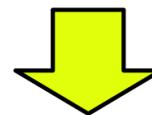
	$f_1$	$f_2$	$f_3$	$f_4$	$f_5$	$f_6$	$f_7$
$x_1$	4	0	2	5	3	0	1
$x_2$	0	0	1	3	4	0	2
$x_3$	2	1	0	0	1	2	5
$x_4$	1	3	0	1	0	1	2

$I_1$	1
$I_2$	1
$I_3$	-1
$I_4$	-1

= X

= L

Linear classifier:  $C = (w_1, w_2, w_3, w_4, w_5, w_6, w_7, b)$  such that  $\text{sign}(X * W' + b) = L$



Test samples:  
 $y_1, y_2, y_3$

	$f_1$	$f_2$	$f_3$	$f_4$	$f_5$	$f_6$	$f_7$
$y_1$	1	0	2	4	2	0	2
$y_2$	1	2	0	1	2	2	1
$y_3$	3	1	0	0	4	1	1

$p_1$	?
$p_2$	?
$p_3$	?

= Y

= P

Apply C to obtain predictions:  $P = \text{sign}(Y * W' + b)$

## Forma duală

Kernel type: linear

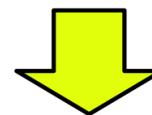
Train samples:  
 $x_1, x_2, x_3, x_4$

	$x_1$	$x_2$	$x_3$	$x_4$
$x_1$	55	31	16	11
$x_2$	31	30	14	7
$x_3$	16	14	35	17
$x_4$	11	7	17	16

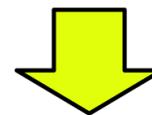
$$= X * X' = K_X$$

$I_1$	1
$I_2$	1
$I_3$	-1
$I_4$	-1

$$= L$$



Linear classifier:  $C = (\alpha_1, \alpha_2, \alpha_3, \alpha_4, b)$  such that  $\text{sign}(K_X * \alpha' + b) = L$



Test samples:  
 $y_1, y_2, y_3$

	$x_1$	$x_2$	$x_3$	$x_4$
$y_1$	36	26	14	9
$y_2$	16	13	15	12
$y_3$	25	18	18	9

$$= Y * X' = K_Y$$

$p_1$	?
$p_2$	?
$p_3$	?

$$= P$$

Apply C to obtain predictions:  $P = \text{sign}(K_Y * \alpha' + b)$

# Regresia liniară

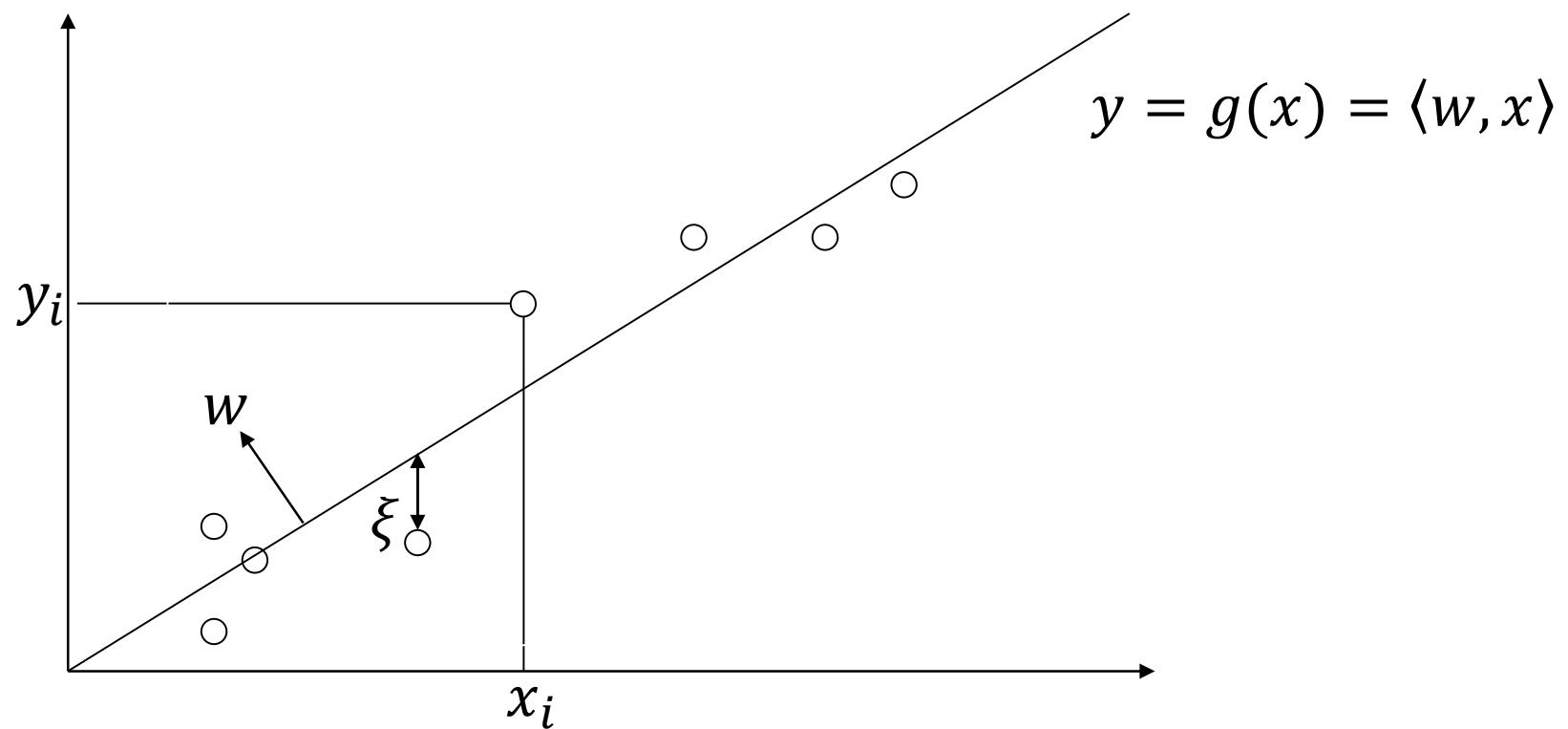
- Problema găsirii funcției  $g$  de forma:

$$g(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle = \mathbf{w}' \mathbf{x} = \sum_{i=1}^n w_i x_i$$

- care interpolează cel mai bine o mulțime de exemple:

$$S = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_\ell, y_\ell)\}$$

# Regresia liniară



# Regresia liniară

- Eroarea funcției liniare pe un exemplu particular:

$$\xi = (y - g(\mathbf{x}))$$

- Funcția de pierdere pe toate exemplele:

$$\begin{aligned}\mathcal{L}(g, S) &= \mathcal{L}(\mathbf{w}, S) = \sum_{i=1}^{\ell} (y_i - g(\mathbf{x}_i))^2 = \\ &= \sum_{i=1}^{\ell} \xi^2 = \sum_{i=1}^{\ell} \mathcal{L}(g, (\mathbf{x}_i, y_i))\end{aligned}$$

# Regresia liniară

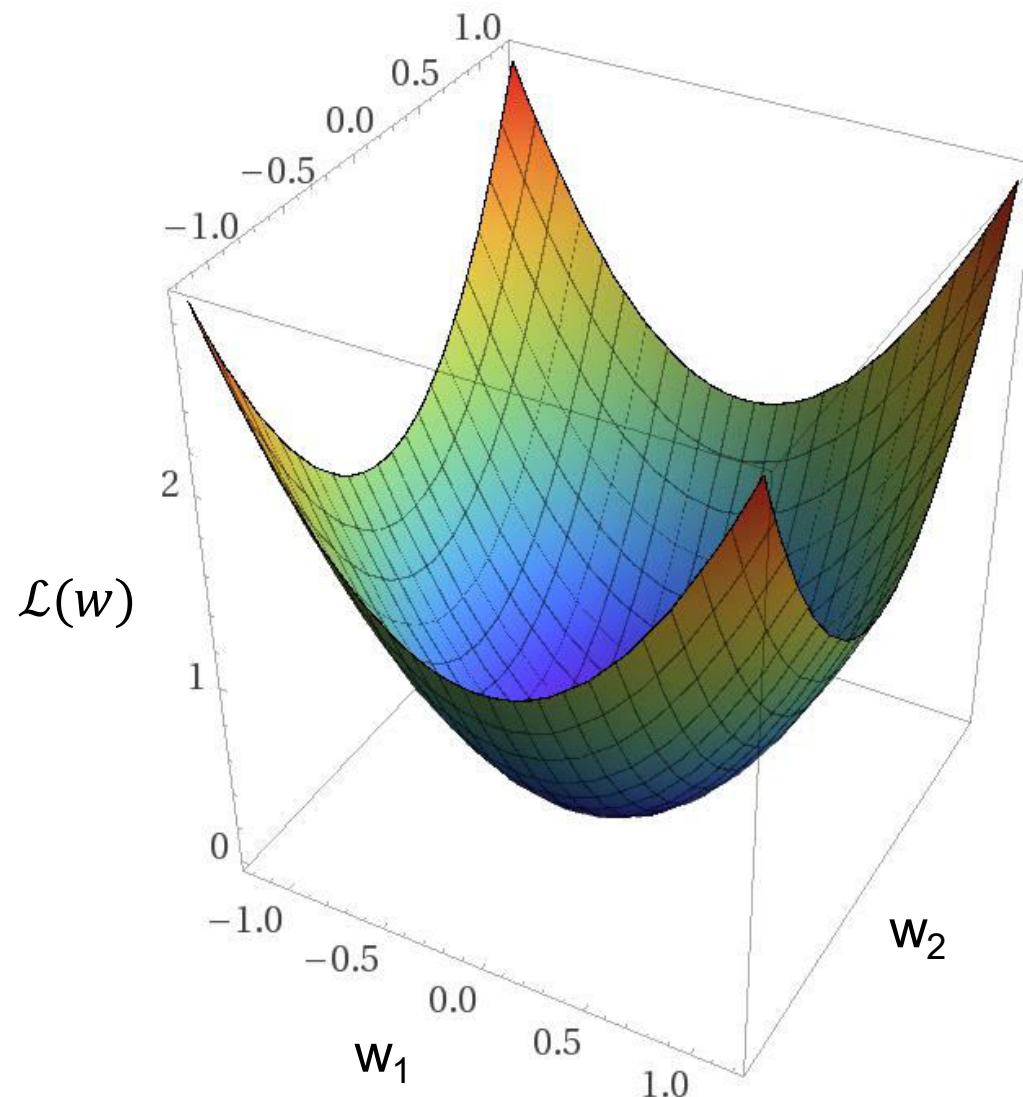
- Funcția de pierdere scrisă vectorial:

$$\xi = \mathbf{y} - \mathbf{X}\mathbf{w}$$

$$\mathcal{L}(\mathbf{w}, S) = \|\xi\|_2^2 = (\mathbf{y} - \mathbf{X}\mathbf{w})'(\mathbf{y} - \mathbf{X}\mathbf{w})$$

- Care este valoarea optimă pentru  $\mathbf{w}$ ?

# Regresia liniară



# Regresia liniară

- Valoarea optimă pentru  $w$ :

$$\frac{\partial \mathcal{L}(w, S)}{\partial w} = -2\mathbf{X}'\mathbf{y} + 2\mathbf{X}'\mathbf{X}w = \mathbf{0}$$

- Ecuată normală devine:

$$\mathbf{X}'\mathbf{X}w = \mathbf{X}'\mathbf{y}$$

- De unde îl putem scoate pe  $w$ , dacă există inversa:

$$w = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$$

# Regresia Ridge

- Dacă inversa nu există, problema este “prost-pusă” și trebuie să utilizăm regularizarea
- Criteriul de optimizare devine:

$$\min_{\mathbf{w}} \mathcal{L}_\lambda(\mathbf{w}, S) = \min_{\mathbf{w}} (\lambda \|\mathbf{w}\|^2 + \sum_{i=1}^{\ell} (y_i - g(\mathbf{x}_i))^2)$$

- Iar soluția optimă pentru  $\mathbf{w}$  este dată de:

$$\frac{\partial \mathcal{L}_\lambda(\mathbf{w}, S)}{\partial \mathbf{w}} = \frac{\partial (\lambda \|\mathbf{w}\|^2 + \sum_{i=1}^{\ell} (y_i - g(\mathbf{x}_i))^2)}{\partial \mathbf{w}} = \mathbf{0}$$

# Regresia Ridge

- Soluția optimă este:

$$\frac{\partial \mathcal{L}_\lambda(\mathbf{w}, S)}{\partial \mathbf{w}} = \frac{\partial (\lambda \|\mathbf{w}\|^2 + (\mathbf{y} - \mathbf{X}\mathbf{w})'(\mathbf{y} - \mathbf{X}\mathbf{w}))}{\partial \mathbf{w}} = \\ = 2\lambda \mathbf{w} - 2\mathbf{X}'\mathbf{y} + 2\mathbf{X}'\mathbf{X}\mathbf{w} = 0$$

$$\mathbf{X}'\mathbf{X}\mathbf{w} + \lambda \mathbf{w} = \mathbf{X}'\mathbf{y}$$

$$(\mathbf{X}'\mathbf{X} + \lambda \mathbf{I}_n)\mathbf{w} = \mathbf{X}'\mathbf{y}$$

$$\mathbf{w} = (\mathbf{X}'\mathbf{X} + \lambda \mathbf{I}_n)^{-1} \mathbf{X}'\mathbf{y}$$

# Regresia Ridge Duală

$$\mathbf{X}'\mathbf{X}\mathbf{w} + \lambda\mathbf{w} = \mathbf{X}'\mathbf{y}$$

$$\mathbf{w} = \lambda^{-1}(\mathbf{X}'\mathbf{y} - \mathbf{X}'\mathbf{X}\mathbf{w}) = \lambda^{-1}\mathbf{X}'(\mathbf{y} - \mathbf{X}\mathbf{w}) = \mathbf{X}'\boldsymbol{\alpha}$$

$$\lambda^{-1}\mathbf{X}'(\mathbf{y} - \mathbf{X}\mathbf{w}) = \mathbf{X}'\boldsymbol{\alpha}$$

$$\boldsymbol{\alpha} = \lambda^{-1}(\mathbf{y} - \mathbf{X}\mathbf{w})$$

- Dar:

$$\mathbf{w} = \mathbf{X}'\boldsymbol{\alpha}$$

- Astfel că:

$$\boldsymbol{\alpha} = \lambda^{-1}(\mathbf{y} - \mathbf{X}\mathbf{X}'\boldsymbol{\alpha})$$

# Regresia Ridge Duală

$$\alpha = \lambda^{-1}(\mathbf{y} - \mathbf{X}\mathbf{X}'\alpha)$$

$$\lambda\alpha = (\mathbf{y} - \mathbf{X}\mathbf{X}'\alpha)$$

$$\mathbf{X}\mathbf{X}'\alpha + \lambda\alpha = \mathbf{y}$$

$$(\mathbf{X}\mathbf{X}' + \lambda\mathbf{I}_\ell)\alpha = \mathbf{y}$$

$$\alpha = (\mathbf{G} + \lambda\mathbf{I}_\ell)^{-1}\mathbf{y}$$

- Unde:  
 $\mathbf{G} = \mathbf{X}\mathbf{X}'$
- este matricea Gram:  
 $\mathbf{G}_{ij} = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$

# Regresia Ridge Duală

- În forma duală, informația din exemplele de antrenare este dată prin matricea Gram ce conține produsul scalar între perechi de puncte:

$$\alpha = (\mathbf{G} + \lambda \mathbf{I}_\ell)^{-1} \mathbf{y}$$

- Funcția de predicție este dată de:

$$g(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle = \left\langle \sum_{i=1}^{\ell} \alpha_i \mathbf{x}_i, \mathbf{x} \right\rangle = \sum_{i=1}^{\ell} \alpha_i \langle \mathbf{x}_i, \mathbf{x} \rangle$$

## Forma primală

Features:  $f_1, f_2, f_3, f_4, f_5, f_6, f_7$

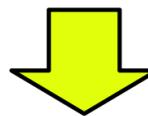
Train samples:  
 $x_1, x_2, x_3, x_4$

	$f_1$	$f_2$	$f_3$	$f_4$	$f_5$	$f_6$	$f_7$
$x_1$	4	0	2	5	3	0	1
$x_2$	0	0	1	3	4	0	2
$x_3$	2	1	0	0	1	2	5
$x_4$	1	3	0	1	0	1	2

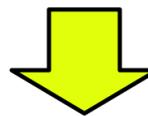
$I_1$	1
$I_2$	1
$I_3$	-1
$I_4$	-1

$$= X$$

$$= L$$



Linear classifier:  $C = (w_1, w_2, w_3, w_4, w_5, w_6, w_7, b)$  such that  $\text{sign}(X * W' + b) = L$



Test samples:  
 $y_1, y_2, y_3$

	$f_1$	$f_2$	$f_3$	$f_4$	$f_5$	$f_6$	$f_7$
$y_1$	1	0	2	4	2	0	2
$y_2$	1	2	0	1	2	2	1
$y_3$	3	1	0	0	4	1	1

$p_1$	?
$p_2$	?
$p_3$	?

$$= Y$$

$$= P$$

Apply C to obtain predictions:  $P = \text{sign}(Y * W' + b)$

## Forma duală

Kernel type: linear

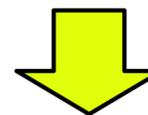
Train samples:  
 $x_1, x_2, x_3, x_4$

	$x_1$	$x_2$	$x_3$	$x_4$
$x_1$	55	31	16	11
$x_2$	31	30	14	7
$x_3$	16	14	35	17
$x_4$	11	7	17	16

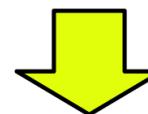
$$= X * X' = K_X$$

$I_1$	1
$I_2$	1
$I_3$	-1
$I_4$	-1

$$= L$$



Linear classifier:  $C = (\alpha_1, \alpha_2, \alpha_3, \alpha_4, b)$  such that  $\text{sign}(K_X * \alpha' + b) = L$



Test samples:  
 $y_1, y_2, y_3$

	$x_1$	$x_2$	$x_3$	$x_4$
$y_1$	36	26	14	9
$y_2$	16	13	15	12
$y_3$	25	18	18	9

$$= Y * X' = K_Y$$

$p_1$	?
$p_2$	?
$p_3$	?

$$= P$$

Apply C to obtain predictions:  $P = \text{sign}(K_Y * \alpha' + b)$

# Regresia Ridge Kernel

- Aplicăm “kernel trick”, înlocuind produsul scalar cu o funcție kernel:

$$\langle \cdot \rangle \mapsto k$$

$$\mathbf{G} = \begin{pmatrix} \langle \mathbf{x}_1, \mathbf{x}_1 \rangle & \langle \mathbf{x}_1, \mathbf{x}_2 \rangle & \cdots & \langle \mathbf{x}_1, \mathbf{x}_n \rangle \\ \langle \mathbf{x}_2, \mathbf{x}_1 \rangle & \langle \mathbf{x}_2, \mathbf{x}_2 \rangle & \cdots & \langle \mathbf{x}_2, \mathbf{x}_n \rangle \\ \vdots & \vdots & \vdots & \vdots \\ \langle \mathbf{x}_n, \mathbf{x}_1 \rangle & \langle \mathbf{x}_n, \mathbf{x}_2 \rangle & \cdots & \langle \mathbf{x}_n, \mathbf{x}_n \rangle \end{pmatrix} \mapsto \mathbf{K} = \begin{pmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & k(\mathbf{x}_1, \mathbf{x}_2) & \cdots & k(\mathbf{x}_1, \mathbf{x}_n) \\ k(\mathbf{x}_2, \mathbf{x}_1) & k(\mathbf{x}_2, \mathbf{x}_2) & \cdots & k(\mathbf{x}_2, \mathbf{x}_n) \\ \vdots & \vdots & \vdots & \vdots \\ k(\mathbf{x}_n, \mathbf{x}_1) & k(\mathbf{x}_n, \mathbf{x}_2) & \cdots & k(\mathbf{x}_n, \mathbf{x}_n) \end{pmatrix}$$

# Regresia Ridge Kernel

- Ponderile duale se calculează astfel:

$$\alpha = (\mathbf{G} + \lambda \mathbf{I}_\ell)^{-1} \mathbf{y} \rightarrow \alpha = (\mathbf{K} + \lambda \mathbf{I}_\ell)^{-1} \mathbf{y}$$

- Funcția de predicție devine:

$$g(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle = \left\langle \sum_{i=1}^{\ell} \alpha_i \mathbf{x}_i, \mathbf{x} \right\rangle = \sum_{i=1}^{\ell} \alpha_i \langle \mathbf{x}_i, \mathbf{x} \rangle$$



$$g(\mathbf{x}) = \sum_{i=1}^{\ell} \alpha_i k(\mathbf{x}_i, \mathbf{x})$$

# Regresia Ridge Kernel (Python)

```
# Parametrul de regularizare lambda:  
lmb = 10 ** -6  
  
# X_train - datele de antrenare (un exemplu pe linie)  
# T_train - clasele datelor de antrenare  
  
n = X_train.shape[0]  
K = np.matmul(X_train, X_train.T)  
  
# Antrenarea metodei:  
alpha = np.matmul(np.linalg.inv(K + lmb * np.eye(n)),  
                  T_train)  
  
# Prezicerea etichetelor pe datele de antrenare:  
Y_train = np.matmul(K, alpha)  
Y_train = np.sign(Y_train)  
  
acc_train = (T_train == Y_train).mean()  
print('Train accuracy: %.4f' % acc_train)
```

# Regresia Ridge Kernel (Python)

```
# X_test - datele de testare (un exemplu pe linie)
# T_test - clasele datelor de testare

K_test = np.matmul(X_test, X_train.T)

# Prezicerea etichetelor pe datele de test:
Y_test = np.matmul(K_test, alpha)
Y_test = np.sign(Y_test)

acc_test = (T_test == Y_test).mean()
print('Test accuracy: %.4f' % acc_test)
```

# Funcția kernel

- **Definiție:** O funcție kernel este o funcție

$$k : X \times X \longmapsto \mathbb{R}$$

pentru care există o funcție de scufundare din spațiul  $X$  în spațiul Hilbert  $F$

$$\phi : x \in \mathbb{R}^m \longmapsto \phi(x) \in F$$

a.î. pentru orice  $x, z \in X$

$$k(x, z) = \langle \phi(x), \phi(z) \rangle$$

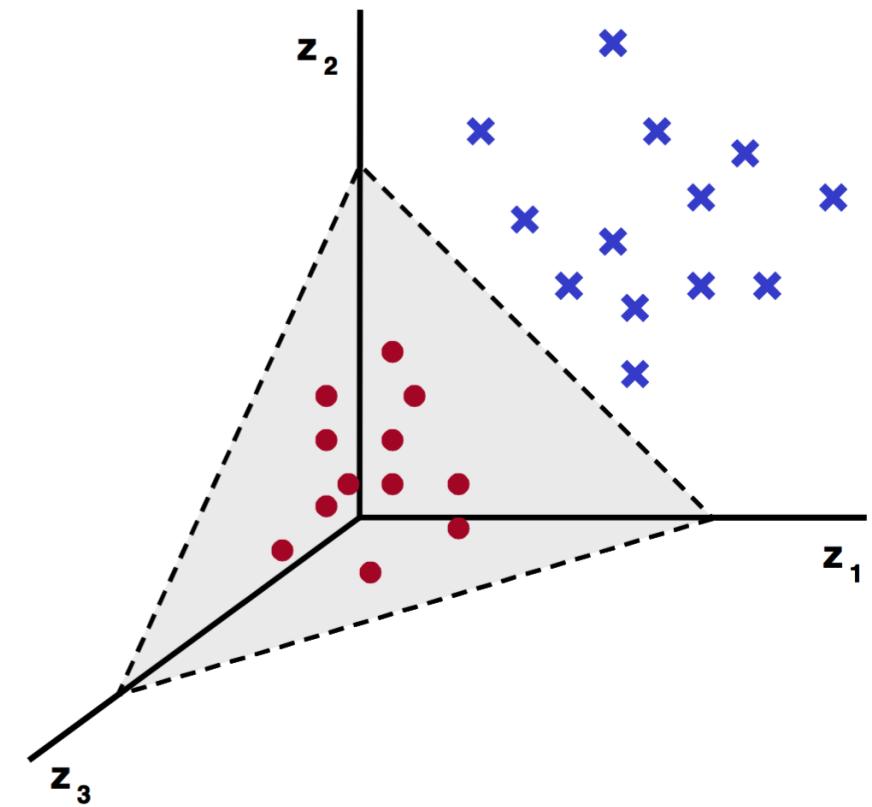
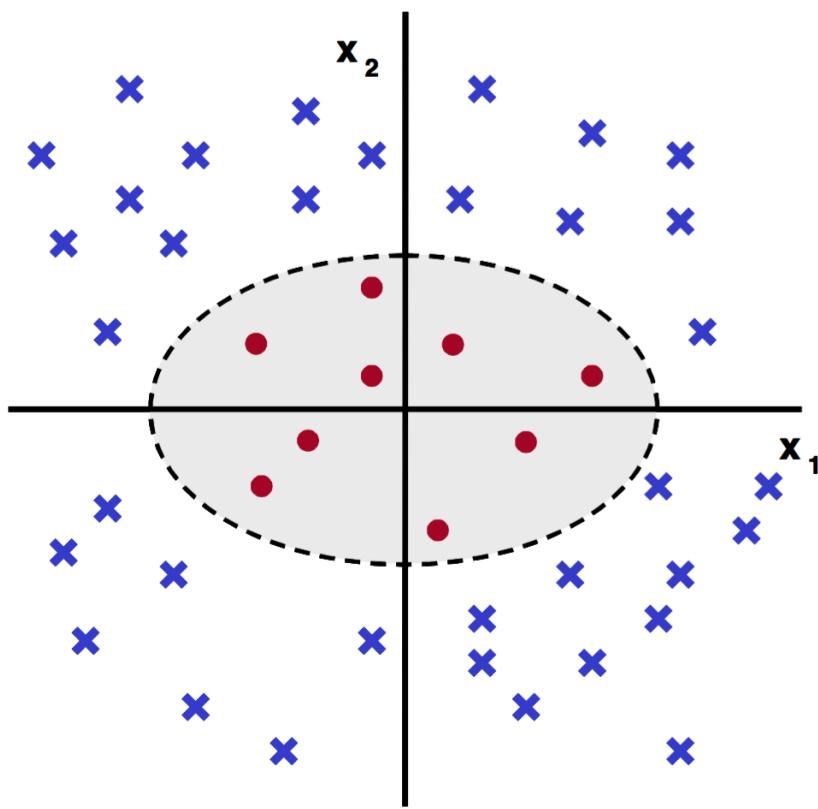
- **Teoremă:** O funcție  $k$  este funcție kernel doar dacă este finit pozitiv semi-definită

# Exemple de funcții kernel

- Prin definirea explicită a funcției de scufundare

$$\phi : \mathbb{R}^2 \rightarrow \mathbb{R}^3$$

$$(x_1, x_2) \mapsto (z_1, z_2, z_3) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)$$



# Exemple de funcții kernel

- Funcția kernel din exemplul anterior:

$$\langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle_F = \left\langle (x_1^2, x_2^2, \sqrt{2}x_1x_2), (z_1^2, z_2^2, \sqrt{2}z_1z_2) \right\rangle$$

$$\langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle_F = x_1^2 z_1^2 + x_2^2 z_2^2 + 2x_1 x_2 z_1 z_2$$

$$\langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle_F = (x_1 z_1 + x_2 z_2)^2$$

$$\langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle_F = \langle \mathbf{x}, \mathbf{z} \rangle^2$$

$$k(\mathbf{x}, \mathbf{z}) = \langle \mathbf{x}, \mathbf{z} \rangle^2$$

- Aceeași funcție kernel corespunde scufundării:

$$\phi: \mathbf{x} = (x_1, x_2) \mapsto \phi(\mathbf{x}) = (x_1^2, x_2^2, x_1 x_2, x_2 x_1)$$

# Functia kernel polinomială

- Pentru o constantă reală pozitivă  $c$  și un număr natural  $d$ :

$$k(\mathbf{x}, \mathbf{z}) = (\langle \mathbf{x}, \mathbf{z} \rangle + c)^d$$

- Constanta  $c$  permite controlul gradului de influență al polinoamelor de diverse grade

# Funcția kernel Gaussiană (RBF)

- Pentru  $x = (1, 2, 4, 1)$  și  $z = (5, 1, 2, 3)$  din  $\mathbb{R}^4$ :

$$k(x, z) = \exp\left(-\frac{\|x - z\|^2}{2\sigma^2}\right)$$

$$= \exp\left(-\frac{\sqrt{(1 - 5)^2 + (2 - 1)^2 + (4 - 2)^2 + (1 - 3)^2}}{2 \cdot 1^2}\right)$$

$$= \exp\left(-\frac{\sqrt{16 + 1 + 4 + 4}}{2}\right)$$

$$= \exp\left(-\frac{5}{2}\right)$$

$$\approx 0.0821.$$

# Functia kernel intersecție

- Pentru  $x = (1, 2, 4, 1)$  și  $z = (5, 1, 2, 3)$  din  $\mathbb{R}^4$ :

$$k(x, z) = \sum_i \min \{x_i, z_i\}$$

$$= \min \{1, 5\} + \min \{2, 1\} + \min \{4, 2\} + \min \{1, 3\}$$

$$= 1 + 1 + 2 + 1$$

$$= 5.$$

# Alte funcții kernel

- Funcția kernel Hellinger:

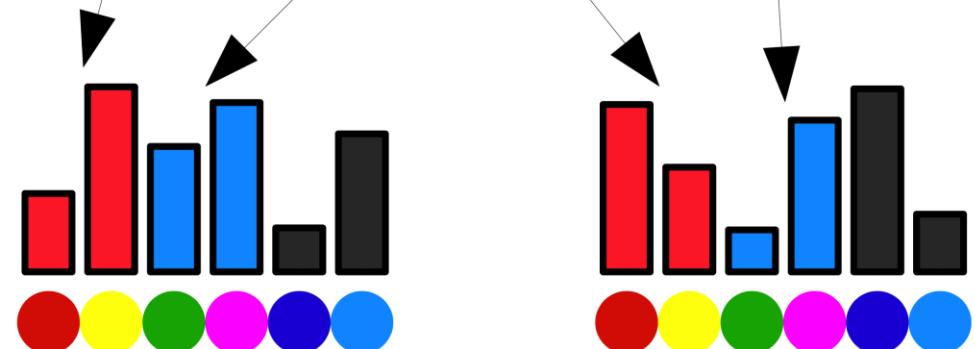
$$k(x, z) = \sum_i \sqrt{x_i \cdot z_i}$$

different order  
(discordant)

same order  
(concordant)

- Funcția kernel PQ:

$$k_{PQ}(X, Y) = 2(P - Q)$$



$$P = |\{(i, j) : 1 \leq i < j \leq n, (x_i - x_j)(y_i - y_j) > 0\}|$$

$$Q = |\{(i, j) : 1 \leq i < j \leq n, (x_i - x_j)(y_i - y_j) < 0\}|$$

# String kernels

- String kernels măsoară similaritatea între perechi de șiruri de caractere, prin numărarea subsecvențelor (n-grame) de caractere comune dintre cele două șiruri
- Textele pot fi interpretate ca șiruri de caractere
- Avantaje:
  - Nu trebuie să delimităm cuvintele
  - Metoda este independentă de limbă

# String kernels

- Exemplu:

Fiiind date  $s = \text{"pineapple pi"}$  și  $t = \text{"apple pie"}$  peste un alfabet  $\Sigma$ , și lungimea n-gramelor  $p = 2$ ,

construim tabele hash  $S$  and  $T$  care conțin perechi  $\langle \text{key} \rangle : \langle \text{value} \rangle$  de tipul

$\langle 2\text{-gram} \rangle : \langle \text{număr de apariții} \rangle$  în  $s$  și  $t$ :

- $S = \{\text{pi:2, in:1, ne:1, ea:1, ap:1, pp:1, pl:1, le:1, e\_\!:1, \_p:1}\}$ ,
- $T = \{\text{ap:1, pp:1, pl:1, le:1, e\_\!:1, \_p:1, pi:1, ie:1}\}$

# String kernel bazat pe biți de prezență

- Funcția string kernel bazată pe biți de prezență este definită astfel:

$$k_2^{0/1}(s, t) = \sum_{v \in \Sigma^p} S^{0/1}(v) \cdot T^{0/1}(v)$$

- Exemplu (continuare):

$S = \{pi:2, in:1, ne:1, ea:1, ap:1, pp:1, pl:1, le:1, e_:1, _p:1\}$ ,

$T = \{ap:1, pp:1, pl:1, le:1, e_:1, _p:1, pi:1, ie:1\}$

$$\begin{aligned} k_2^{0/1}(s, t) &= 1 \cdot 1 + 1 \cdot 1 \\ &= 1 + 1 + 1 + 1 + 1 + 1 + 1 \\ &= 7 \end{aligned}$$

# De ce metode kernel?

- Obțin rezultate state-of-the-art în anumite probleme:
  - Native Language Identification [Ionescu & Popescu, BEA17]
  - Arabic Dialect Identification [Butnaru & Ionescu, VarDial18]
  - Romanian Dialect Identification [Butnaru & Ionescu, ACL19]
- Utile pentru obținerea unei reprezentări mai compacte în cazul în care:  
numărul de exemple << numărul de trăsături
- Numărul de n-grame unice în setul de date TOEFL11:  
**4,662,520**
- ... versus numărul de exemple de antrenare:  
**11,000**

# De ce metode kernel?

- Generalizează mai bine decât cuvintele
- Exemple de transfer al limbii native din TOEFL11

German		French		Arabic		Hindi		Spanish		Chinese	
1	, that	1	indeed	1	alot	2	as compa	1	, is	2	t most
6	german	19	onnal	9	any	9	hence	2	difer	4	chin
11	. but	21	is to	13	them	16	then	13	, but	7	just
13	often	26	franc	16	thier	17	indi	15	, etc	8	still
207	special	28	to concl	19	his	21	towards	17	cesar	14	. take

Italian		Japanese		Korean		Telugu		Turkish			
1	ital	1	japan	1	korea	1	i concl	1	i agree.		
3	o beca	15	. if	24	e that	6	days	11	turk		
4	fact	19	i disa	27	. as	7	.the	21	. becau		
9	, for	27	. the	30	soci	11	where as	32	s about		
24	the life	38	. it	36	. also	13	e above	37	being		

# Exemple pe cazul French→English

- {onnal}

“...many academics subjects. **Additionnally**, people always have a subject...”

“I would not be in control of my **personnal** schedule during the trip.”

- {evelopp}

“...and who will have the curiosity to **dvelopp** research on the disease.”

“...be able to do so. **Underdevelopped** countries are a case in point.”

- {n France}

“...studied law in both *England* and **in France**, I have had the chance...”

“Numbers have actually shown that **in France** the number of new cars...”

- {to conc}

“...without a tour guide. **To conclude**, there are several advantages...”

“...job they will enjoy. **To conclude**, I think that the best solution is...”

- {exemple}

“...after using them. Onother **exemple** is my underwear that I bough...”

“Science is a great **exemple** of how successful people want to improve...”

# Noi funcții kernel din combinații

- Fiind date două funcții kernel  $k_1$  și  $k_2$ , o constantă reală pozitivă  $a$ , o funcție  $f$  cu valori reale și o matrice simetrică și pozitiv semi-definită  $B$ , următoarele funcții sunt kernel:

$$(i) \ k(x, z) = k_1(x, z) + k_2(x, z);$$

$$(ii) \ k(x, z) = ak_1(x, z);$$

$$(iii) \ k(x, z) = k_1(x, y) \cdot k_2(y, z);$$

$$(iv) \ k(x, z) = f(x) \cdot f(z);$$

$$(v) \ k(x, z) = x' B z.$$

## Forma primală

Features:  $f_1, f_2, f_3, f_4, f_5, f_6, f_7$

Train samples:  
 $x_1, x_2, x_3, x_4$

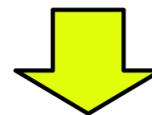
	$f_1$	$f_2$	$f_3$	$f_4$	$f_5$	$f_6$	$f_7$
$x_1$	4	0	2	5	3	0	1
$x_2$	0	0	1	3	4	0	2
$x_3$	2	1	0	0	1	2	5
$x_4$	1	3	0	1	0	1	2

$I_1$	1
$I_2$	1
$I_3$	-1
$I_4$	-1

= X

= L

Linear classifier:  $C = (w_1, w_2, w_3, w_4, w_5, w_6, w_7, b)$  such that  $\text{sign}(X * W' + b) = L$



Test samples:  
 $y_1, y_2, y_3$

	$f_1$	$f_2$	$f_3$	$f_4$	$f_5$	$f_6$	$f_7$
$y_1$	1	0	2	4	2	0	2
$y_2$	1	2	0	1	2	2	1
$y_3$	3	1	0	0	4	1	1

$p_1$	?
$p_2$	?
$p_3$	?

= Y

= P

Apply C to obtain predictions:  $P = \text{sign}(Y * W' + b)$

## Forma duală

Kernel type: linear

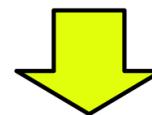
Train samples:  
 $x_1, x_2, x_3, x_4$

	$x_1$	$x_2$	$x_3$	$x_4$
$x_1$	55	31	16	11
$x_2$	31	30	14	7
$x_3$	16	14	35	17
$x_4$	11	7	17	16

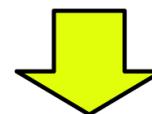
$$= X * X' = K_X$$

$I_1$	1
$I_2$	1
$I_3$	-1
$I_4$	-1

$$= L$$



Linear classifier:  $C = (\alpha_1, \alpha_2, \alpha_3, \alpha_4, b)$  such that  $\text{sign}(K_X * \alpha' + b) = L$



Test samples:  
 $y_1, y_2, y_3$

	$x_1$	$x_2$	$x_3$	$x_4$
$y_1$	36	26	14	9
$y_2$	16	13	15	12
$y_3$	25	18	18	9

$$= Y * X' = K_Y$$

$p_1$	?
$p_2$	?
$p_3$	?

$$= P$$

Apply C to obtain predictions:  $P = \text{sign}(K_Y * \alpha' + b)$

# Normalizarea datelor

- În forma primală:

$$x \longmapsto \phi(x) \longmapsto \frac{\phi(x)}{\|\phi(x)\|}$$

- În forma duală:

$$\hat{k}(x_i, x_j) = \frac{k(x_i, x_j)}{\sqrt{k(x_i, x_i) \cdot k(x_j, x_j)}}$$

- Direct pe matricea kernel:

$$\hat{K}_{ij} = \frac{K_{ij}}{\sqrt{K_{ii} \cdot K_{jj}}}$$

# Normalizarea datelor (Python)

% X - datele (un exemplu pe linie)

% Norma L2 în forma primală:

```
norms = np.linalg.norm(X, axis = 1, keepdims = True)  
X = X / norms
```

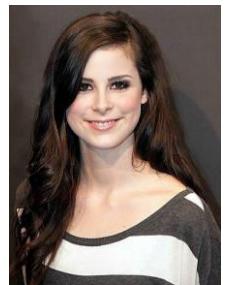
% Norma L2 în forma duală:

```
K = np.matmul(X, X.T)  
KNorm = np.sqrt(np.diag(K))  
KNorm = KNorm[np.newaxis]  
K = K / np.matmul(KNorm.T, KNorm)
```

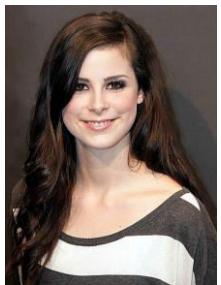
# Cum separăm optim aceste exemple?



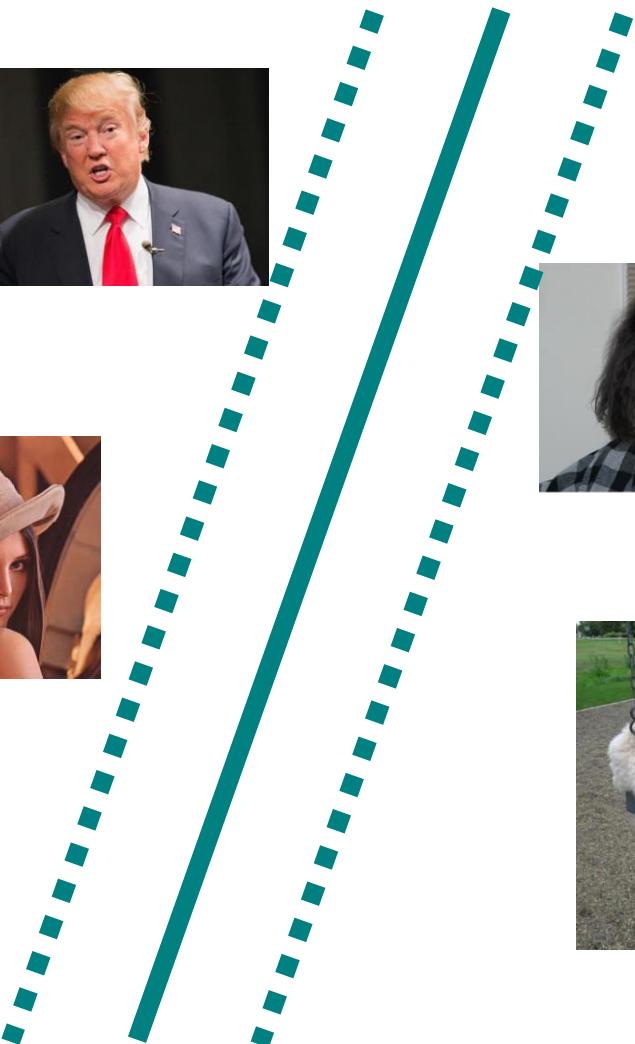
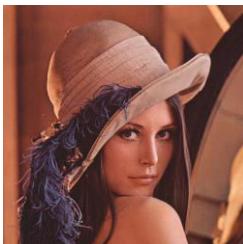
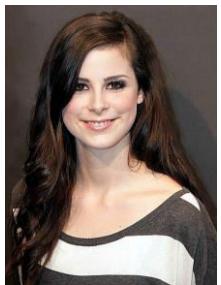
# Cum separăm optim aceste exemple?



# Cum separăm optim aceste exemple?

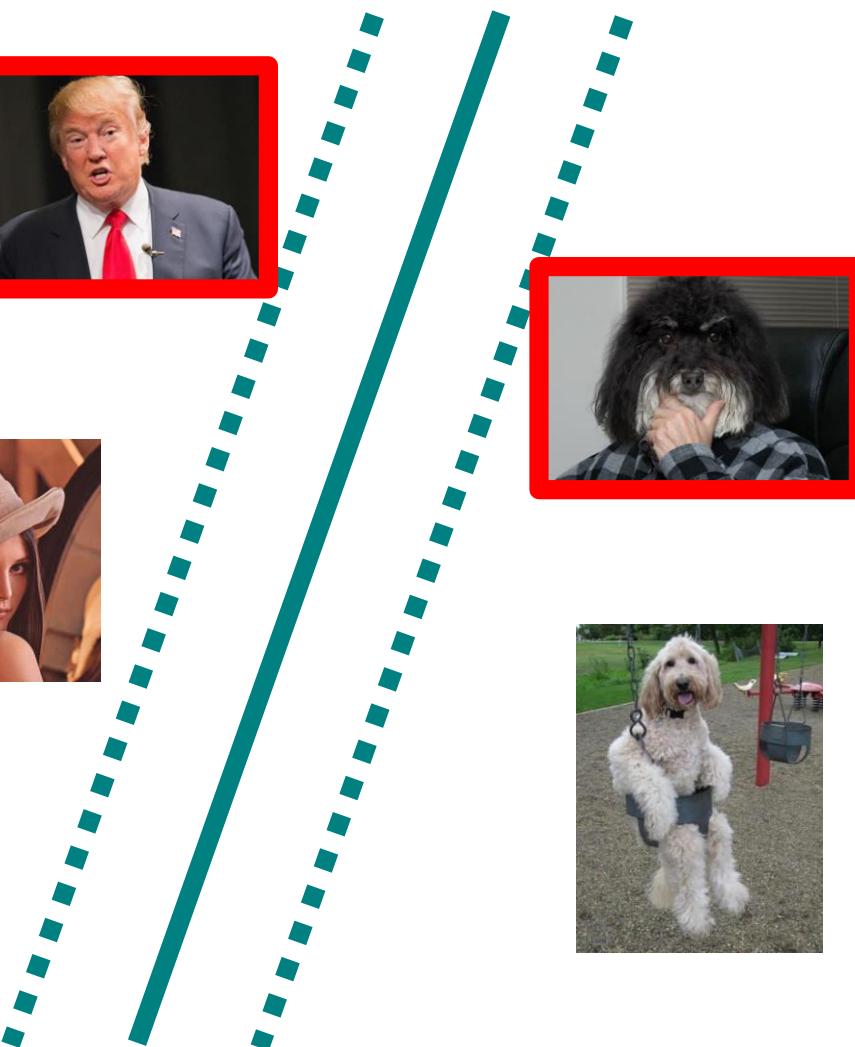


# Alegem hiperplanul de margine maximă



# Alegem hiperplanul de margine maximă

- Mașini cu **vectori suport** (SVM)



# SVM (Hard Margin)

$$S = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_\ell, y_\ell)\}$$

$$g(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b$$

$$\max_{\mathbf{w}, b, \gamma} \gamma$$

subject to

$$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq \gamma$$

$$i = 1, \dots, \ell$$

$$\|\mathbf{w}\|^2 = 1$$



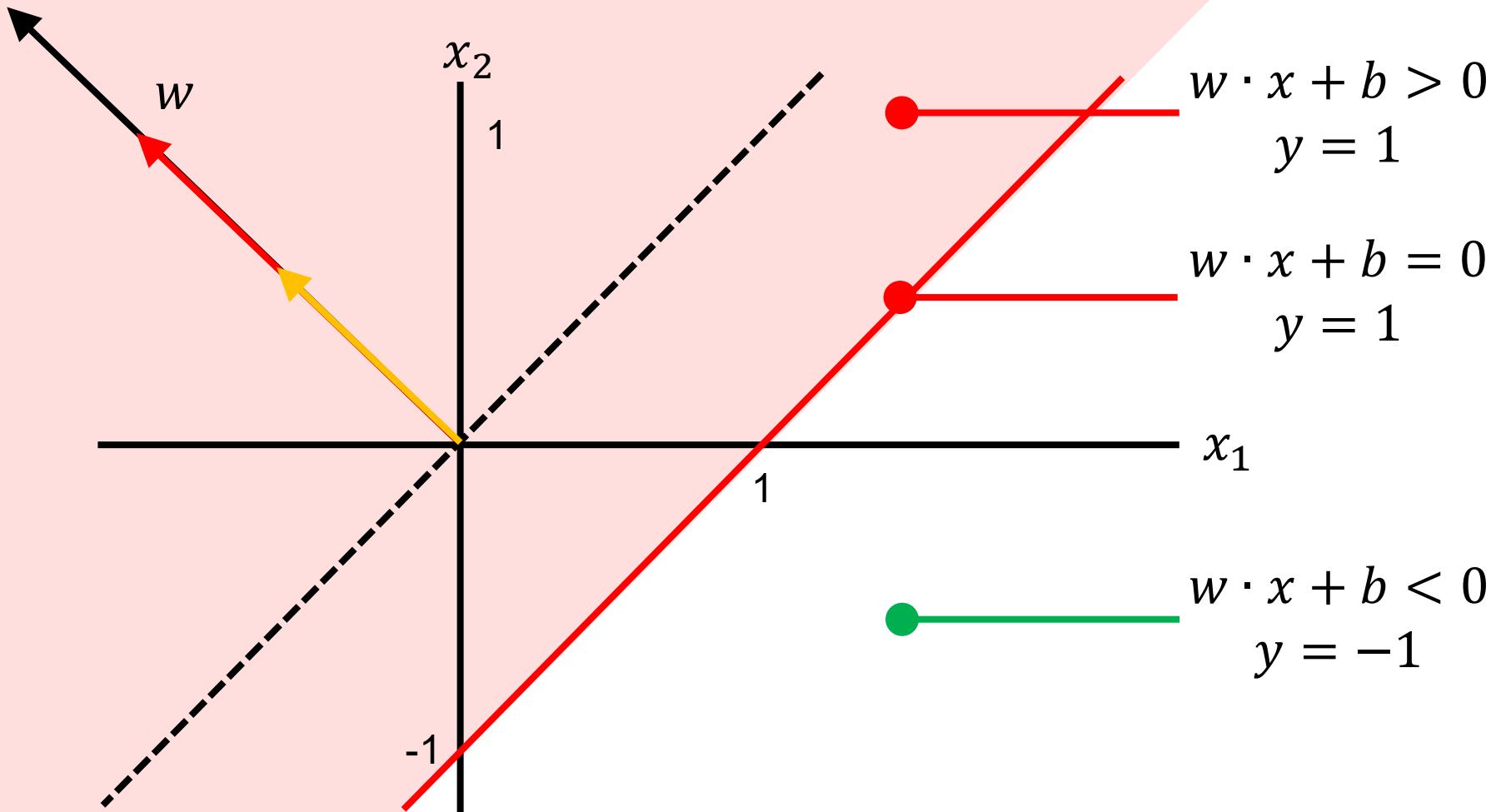
$$\min_{\mathbf{w}, b} \|\mathbf{w}\|^2$$

subject to

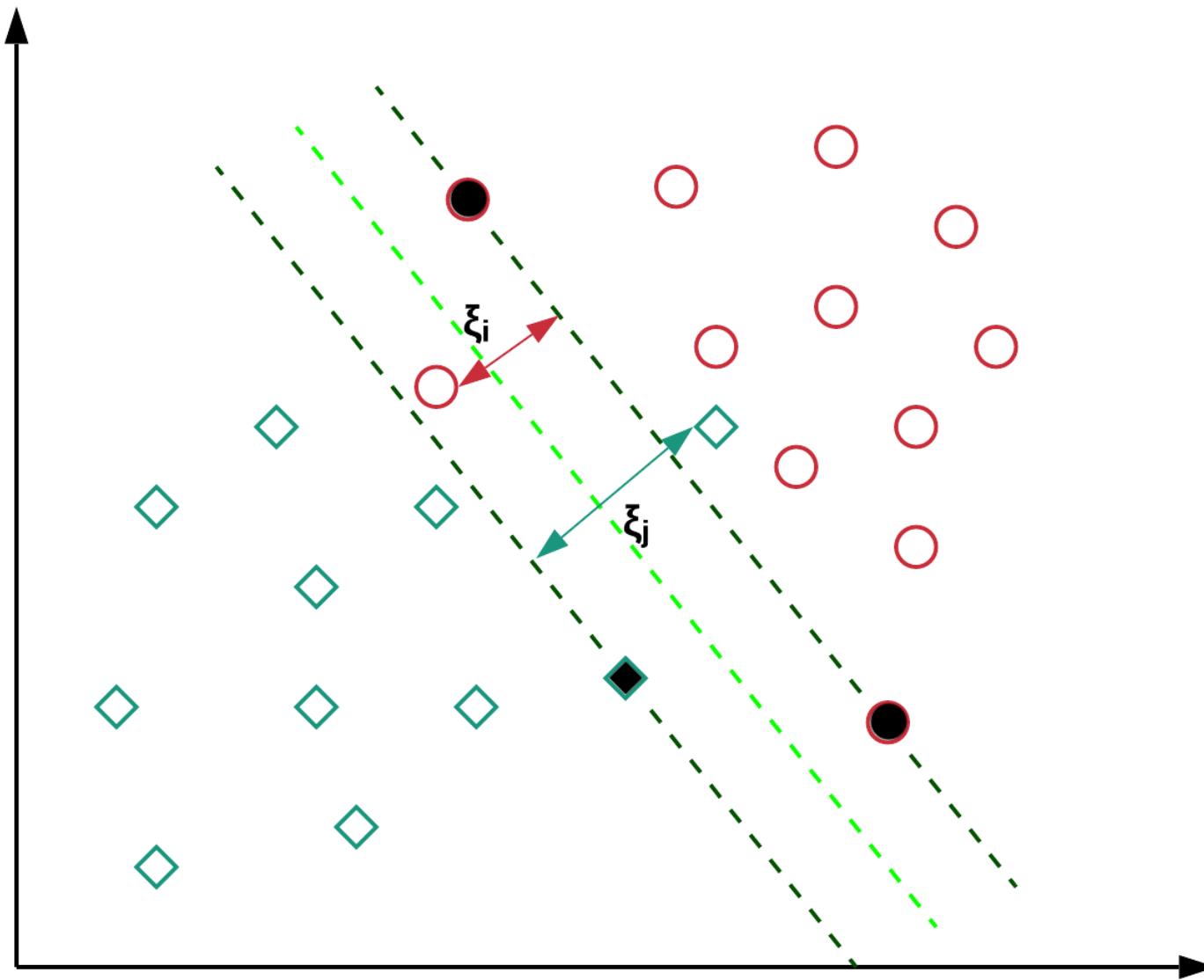
$$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1$$

$$i = 1, \dots, \ell$$

# Granita de separare liniară



# SVM (Soft Margin)



# SVM (Soft Margin)

- În cazul în care exemple nu sunt liniar separabile:

$$\min_{\mathbf{w}, b, \gamma, \xi} -\gamma + C \sum_{i=1}^{\ell} \xi_i$$

subject to

$$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq \gamma - \xi_i$$
$$\xi_i \geq 0 \quad i = 1, \dots, \ell$$
$$\|\mathbf{w}\|^2 = 1$$

$$\min_{\mathbf{w}, b, \xi} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^{\ell} \xi_i$$

subject to

$$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i$$
$$\xi_i \geq 0 \quad i = 1, \dots, \ell$$

# SVM (Python)

- Scikit-learn:

<https://scikit-learn.org/stable/modules/svm.html#svm-classification>

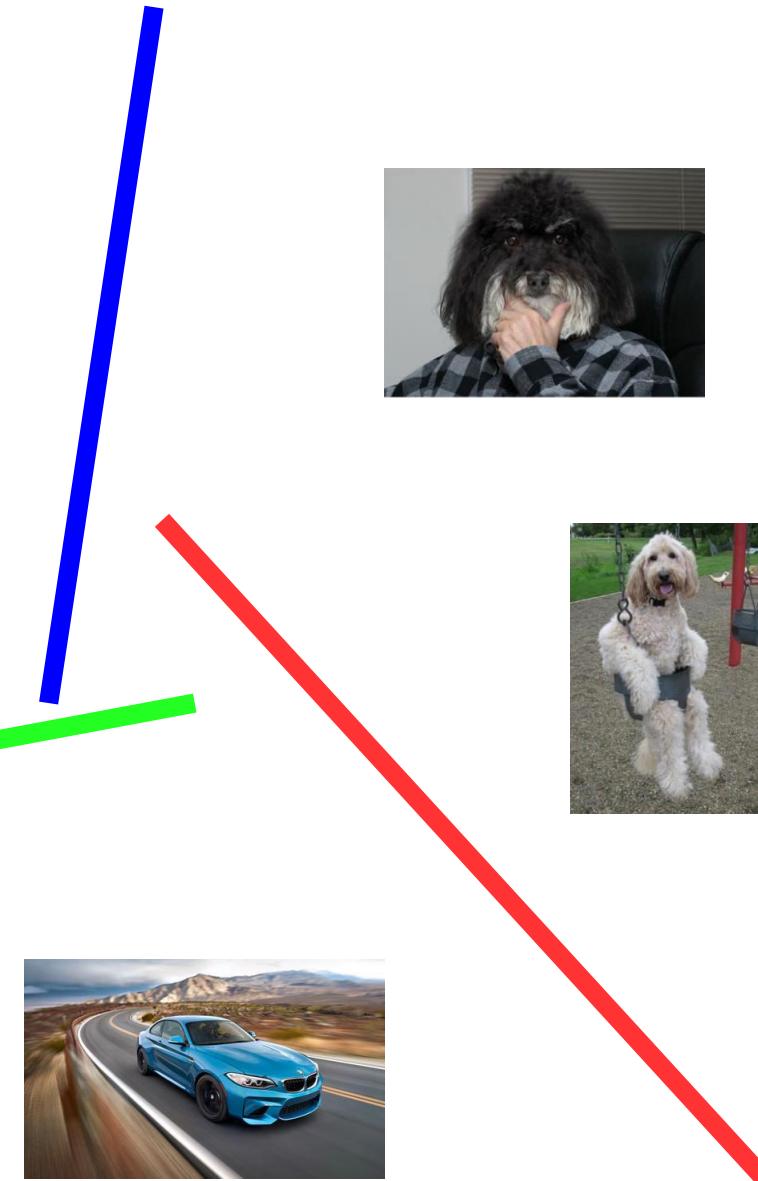
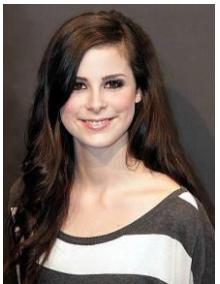
```
from sklearn import svm  
  
clf = svm.SVC(C = 1.0)  
  
clf.fit(X_train, T_train)  
  
Y_test = clf.predict(X_test)
```

- Plus mulți alți clasificatori

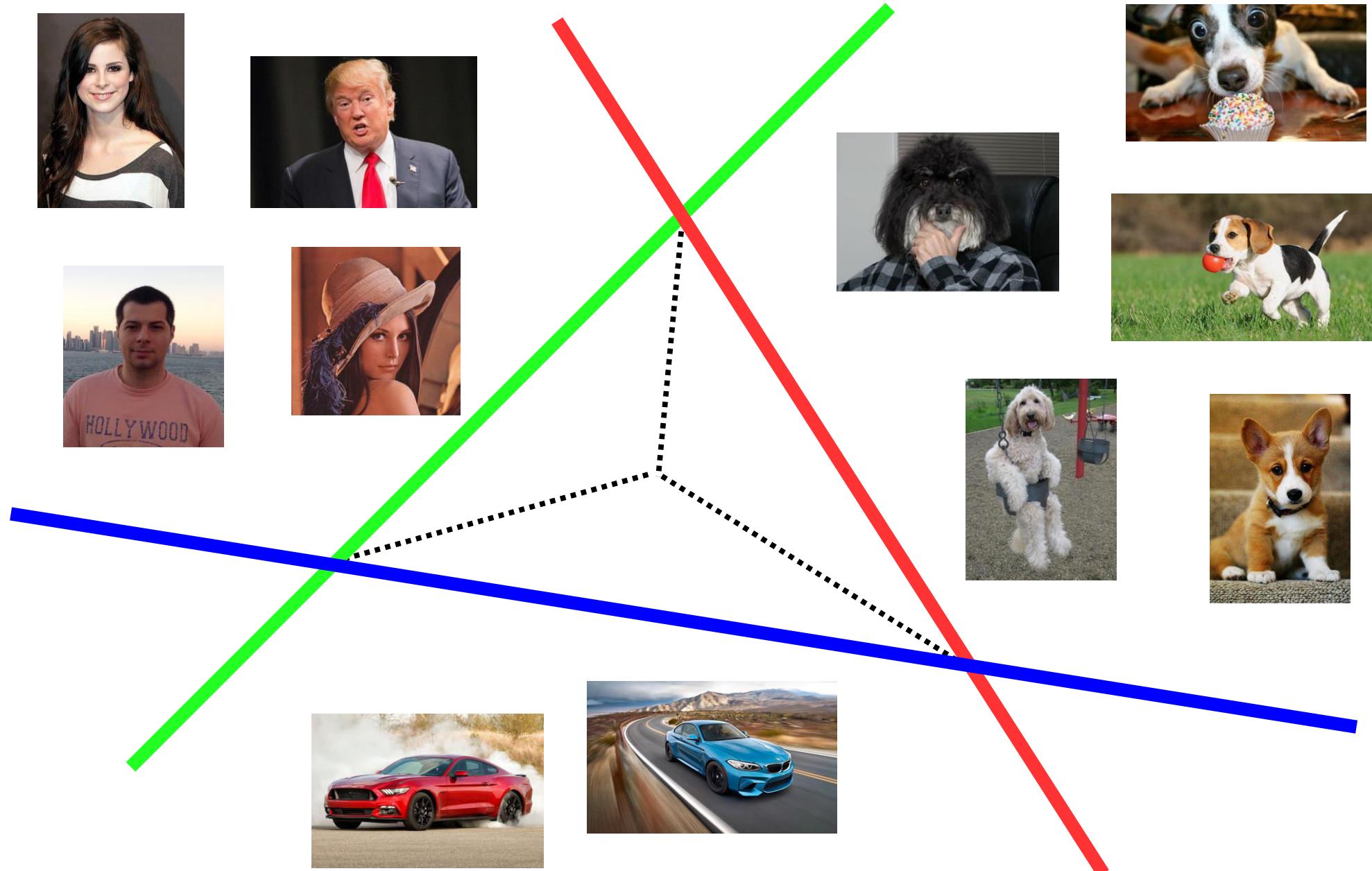
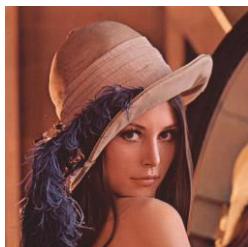
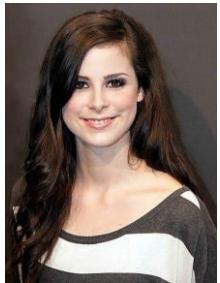
# Cum rezolvăm problemele cu mai multe clase?

- Scheme de combinare a mai multor clasificatori binari:
  - 1) One-versus-one
  - 2) One-versus-all

# One-versus-one



# One-versus-all



# Cum rezolvăm problemele cu mai multe clase?

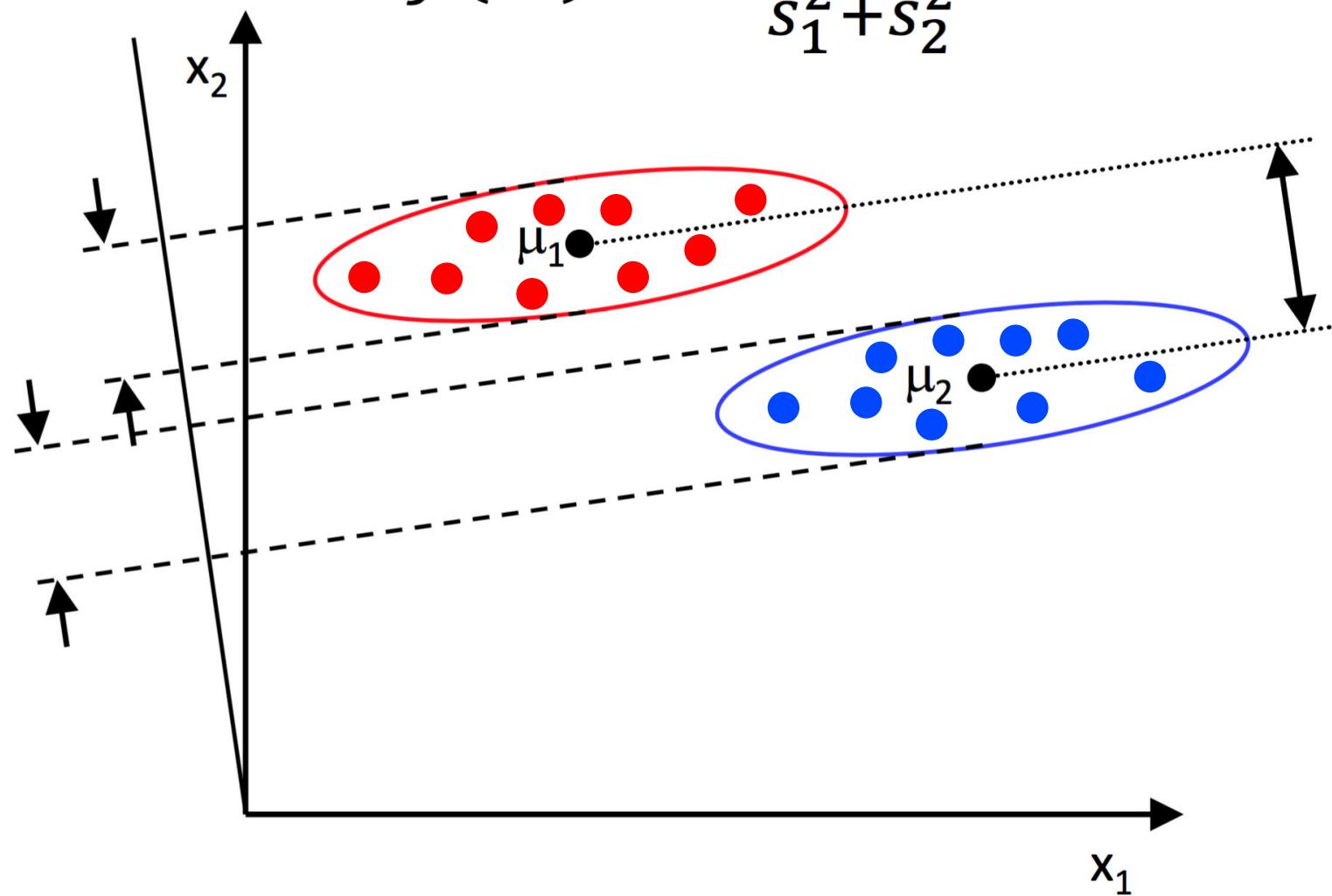
- Utilizarea unor metode de clasificare capabile să rezolve direct problema:
  - 1) Analiza liniar discriminantă (Fisher)
  - 2) Rețele neuronale (cursul următor)

# Analiza liniar discriminantă

- Fiecare clasă este aproximată cu o distribuție Gaussiană
- Algoritmul presupune găsirea unui hiperplan pe care se proiectează punctele a.î.:
  - distanța dintre mediile claselor este maximizată
  - dispersia fiecărei clase este minimizată

# Analiza liniar discriminantă

$$J(w) = \frac{|\mu_1 - \mu_2|^2}{s_1^2 + s_2^2}$$



# Analiza liniar discriminantă (Python)

- Scikit-learn:

[https://scikit-learn.org/stable/modules/lda\\_qda.html](https://scikit-learn.org/stable/modules/lda_qda.html)

```
from sklearn.discriminant_analysis  
      import LinearDiscriminantAnalysis  
  
clf = LinearDiscriminantAnalysis()  
  
clf.fit(X_train, T_train)  
  
Y_test = clf.predict(X_test)
```

# Ce metodă de clasificare este cea mai bună?

- **Teorema “No free lunch”:**

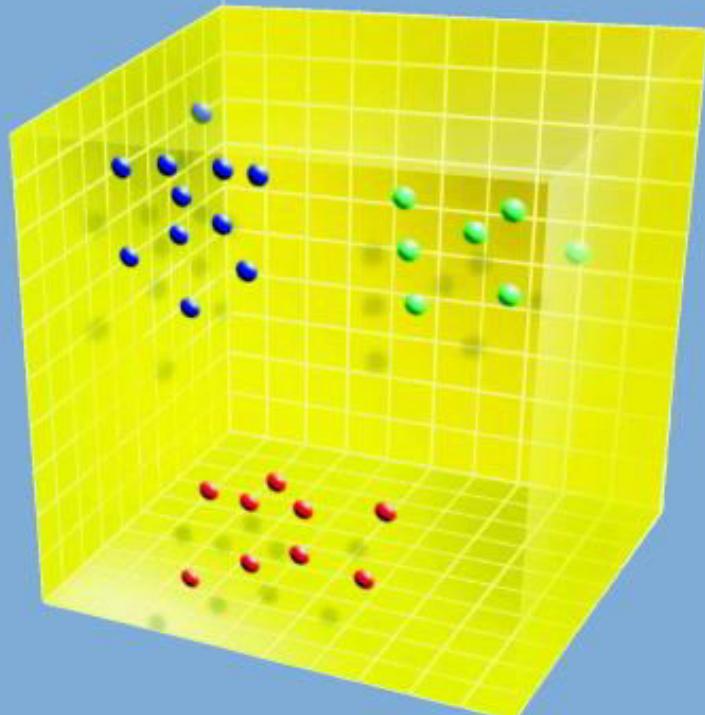
Orice doi algoritmi sunt echivalenți atunci când performanța lor este măsurată (în medie) pe toate problemele posibile

- Rezultă ca nu există nici o scurtătură în alegerea algoritmului potrivit pentru o anumită problemă
- Deobicei încercăm mai mulți algoritmi și vedem care obține rezultate mai bune

# Bibliografie

John Shawe-Taylor  
and Nello Cristianini

# Kernel Methods for Pattern Analysis



CAMBRIDGE

Advances in Computer Vision and Pattern Recognition



Radu Tudor Ionescu  
Marius Popescu

# Knowledge Transfer between Computer Vision and Text Mining

Similarity-based Learning Approaches

 Springer

# Optimizarea funcțiilor de pierdere. Algoritmul coborârii pe gradient.

Prof. Dr. Radu Ionescu

[raducu.ionescu@gmail.com](mailto:raducu.ionescu@gmail.com)

Facultatea de Matematică și Informatică

Universitatea din București

# Clasificator liniar pentru mai multe clase



trăsături parametri

$$f(\mathbf{x}, \mathbf{W})$$

N numere ce indică scorurile pentru fiecare clasă

Vector de trăsături

0.2	-0.5	0.1	2.0
1.5	1.3	2.1	0.0
0	0.25	0.2	-0.3

$$\begin{matrix} 56 \\ 231 \\ 24 \\ 2 \end{matrix}$$

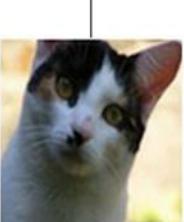
+

1.1
3.2
-1.2

-96.8
437.9
61.95

cat score  
dog score  
ship score

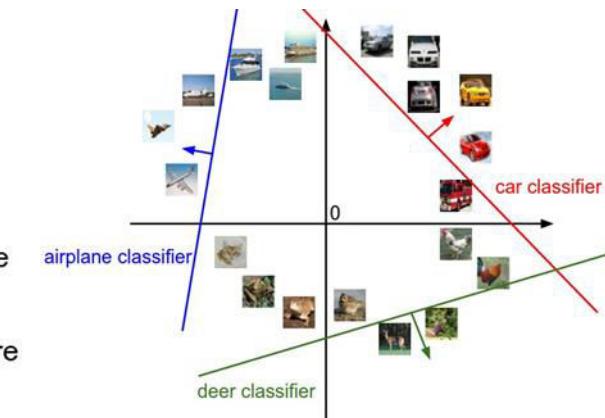
$$f(x_i; W, b)$$



input image

$W$

$x_i$



Pentru 3 exemple de antrenare, 3 clase, și ponderile W, obținem scorurile:  $f(x, W) = Wx$



pisică	<b>3.2</b>	1.3	2.2
mașină	<b>5.1</b>	<b>4.9</b>	2.5
broască	-1.7	2.0	<b>-3.1</b>

## Funcția de pierdere pentru SVM multi-clasă:

Fiind dat un exemplu  $(x_i, y_i)$  unde  $x_i$  este vectorul de trăsături și  $y_i$  este eticheta asociată (întreg), notând vectorul de scoruri cu:

$$s = f(x_i, W)$$

funcția de pierdere a clasificatorului SVM are forma:

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

Pentru 3 exemple de antrenare, 3 clase, și ponderile W, obținem scorurile:  $f(x, W) = Wx$



pisică	<b>3.2</b>	1.3	2.2
mașină	<b>5.1</b>	<b>4.9</b>	2.5
broască	<b>-1.7</b>	2.0	<b>-3.1</b>
Pierderile:	<b>2.9</b>		

## Funcția de pierdere pentru SVM multi-clasă:

Fiind dat un exemplu  $(x_i, y_i)$  unde  $x_i$  este vectorul de trăsături și  $y_i$  este eticheta asociată (întreg), notând vectorul de scoruri cu:

$$s = f(x_i, W)$$

funcția de pierdere a clasificatorului SVM are forma:

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

$$\begin{aligned}
 &= \max(0, 5.1 - 3.2 + 1) \\
 &\quad + \max(0, -1.7 - 3.2 + 1) \\
 &= \max(0, 2.9) + \max(0, -3.9) \\
 &= 2.9 + 0 \\
 &= 2.9
 \end{aligned}$$

Pentru 3 exemple de antrenare, 3 clase, și ponderile W, obținem scorurile:  $f(x, W) = Wx$



pisică	<b>3.2</b>	1.3	2.2
mașină	<b>5.1</b>	<b>4.9</b>	2.5
broască	-1.7	2.0	<b>-3.1</b>
Pierderile:	2.9	0	

## Funcția de pierdere pentru SVM multi-clasă:

Fiind dat un exemplu  $(x_i, y_i)$  unde  $x_i$  este vectorul de trăsături și  $y_i$  este eticheta asociată (întreg), notând vectorul de scoruri cu:

$$s = f(x_i, W)$$

funcția de pierdere a clasificatorului SVM are forma:

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

$$\begin{aligned}
 &= \max(0, 1.3 - 4.9 + 1) \\
 &\quad + \max(0, 2.0 - 4.9 + 1) \\
 &= \max(0, -2.6) + \max(0, -1.9) \\
 &= 0 + 0 \\
 &= 0
 \end{aligned}$$

Pentru 3 exemple de antrenare, 3 clase, și ponderile W, obținem scorurile:  $f(x, W) = Wx$



pisică	<b>3.2</b>	1.3	<b>2.2</b>
mașină	<b>5.1</b>	<b>4.9</b>	2.5
broască	-1.7	2.0	<b>-3.1</b>
Pierderile:	2.9	0	<b>12.9</b>

## Funcția de pierdere pentru SVM multi-clasă:

Fiind dat un exemplu  $(x_i, y_i)$  unde  $x_i$  este vectorul de trăsături și  $y_i$  este eticheta asociată (întreg), notând vectorul de scoruri cu:

$$s = f(x_i, W)$$

funcția de pierdere a clasificatorului SVM are forma:

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

$$\begin{aligned}
 &= \max(0, 2.2 - (-3.1) + 1) \\
 &\quad + \max(0, 2.5 - (-3.1) + 1) \\
 &= \max(0, 6.3) + \max(0, 6.6) \\
 &= 6.3 + 6.6 \\
 &= 12.9
 \end{aligned}$$

Pentru 3 exemple de antrenare, 3 clase, și ponderile W, obținem scorurile:  $f(x, W) = Wx$



pisică	<b>3.2</b>	1.3	2.2
mașină	<b>5.1</b>	<b>4.9</b>	2.5
broască	-1.7	2.0	<b>-3.1</b>
Pierderile:	<b>2.9</b>	0	<b>12.9</b>

## Funcția de pierdere pentru SVM multi-clasă:

Fiind dat un exemplu  $(x_i, y_i)$  unde  $x_i$  este vectorul de trăsături și  $y_i$  este eticheta asociată (întreg), notând vectorul de scoruri cu:

$$s = f(x_i, W)$$

funcția de pierdere a clasificatorului SVM are forma:

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

$$\begin{aligned} L &= (2.9 + 0 + 12.9)/3 \\ &= 5.27 \end{aligned}$$

# Clasificatorul Softmax (Regresia Logistică Multinomială)



pisică	<b>3.2</b>
mașină	<b>5.1</b>
broască	<b>-1.7</b>

**scoruri = log-probabilitățile nenormalizate ale claselor**

$$P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$
 unde  $s = f(x_i; W)$

Vrem să maximizăm the log-probabilitatea, sau (pentru o funcție de pierdere) să minimizăm log probabilitatea negativă a clasei corecte:

$$L_i = -\log P(Y = y_i | X = x_i)$$

---

În concluzie:  $L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$

# Clasificatorul Softmax (Regresia Logistică Multinomială)



$$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

probabilități nenormalizate

pisică

mașină

broască

3.2
5.1
-1.7

exp

24.5
164.0
0.18

normalizare

0.13
0.87
0.00

$$\rightarrow L_i = -\log(0.13) = 2.04$$

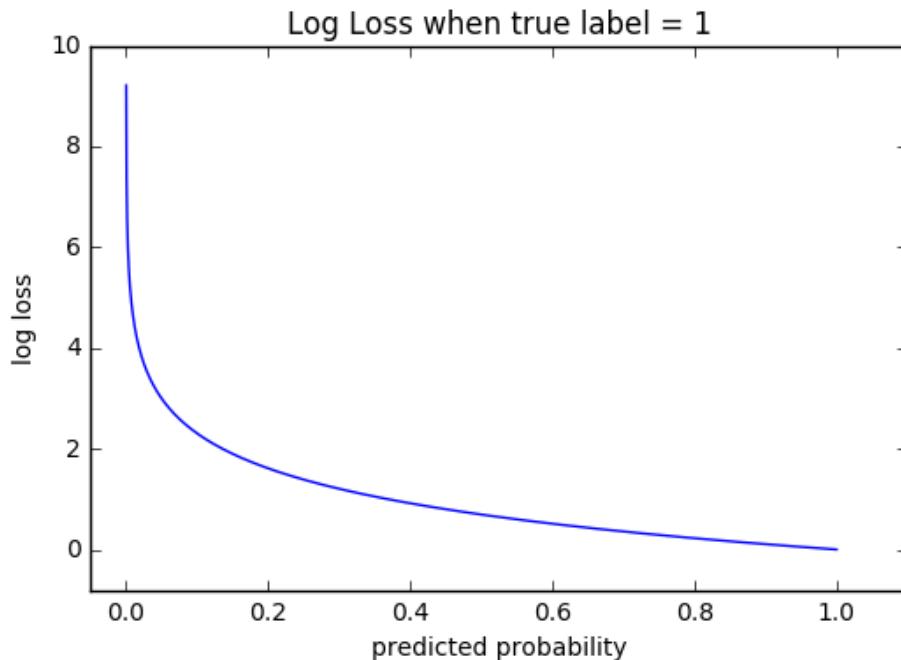
log-probabilități nenormalizate

probabilități

Q: Care sunt valorile minime/maxime pe care le poate avea funcție de pierdere  $L_i$ ?

# Clasificatorul Softmax (Regresia Logistică Multinomială)

$$\mathcal{L}_i = -\log \left( \frac{e^{s_{y_i}}}{\sum_j e^{s_j}} \right)$$



hinge loss (SVM)

matrix multiply + bias offset

0.01	-0.05	0.1	0.05
0.7	0.2	0.05	0.16
0.0	-0.45	-0.2	0.03

$W$

-15	0.0
22	0.2
-44	-0.3
56	

+

0.0	0.2	0.28
0.86		
0.28		

-2.85
0.86
0.28

$$\begin{aligned} & \max(0, -2.85 - 0.28 + 1) + \\ & \max(0, 0.86 - 0.28 + 1) \\ & = \\ & \mathbf{1.58} \end{aligned}$$

cross-entropy loss (Softmax)

$y_i$  2

-2.85	0.058	0.016
0.86	2.36	0.631
0.28	1.32	0.353

$\exp$

normalize  
(to sum to one)

$$\begin{aligned} & -\log(0.353) \\ & = \\ & \mathbf{0.452} \end{aligned}$$

# Softmax vs. SVM

$$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

Presupunem scorurile:

[10, -2, 3]

[10, 9, 9]

[10, -100, -100]

și  $y_i = 0$

Q: Dacă perturbăm vectorul de trăsături cu valori mici (schimbând scorurile rezultate), ce se întâmplă cu funcția de pierdere în cele două cazuri?

# Optimizarea funcțiilor de pierdere

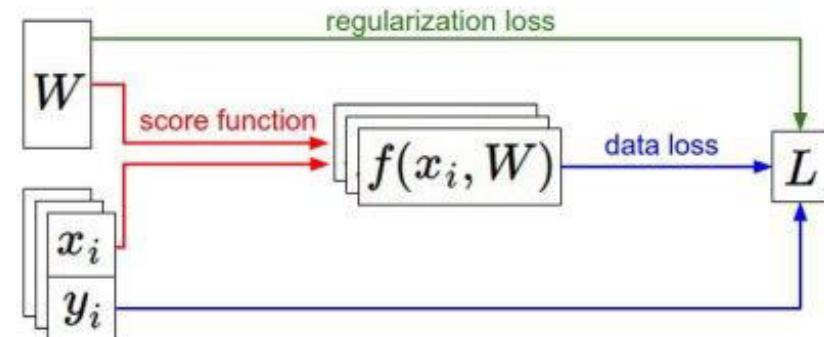
# Până acum avem:

- O mulțime de perechi  $(x, y)$
- O funcție de atribuire a scorului:  $s = f(x; W) = Wx$
- O funcție de pierdere:

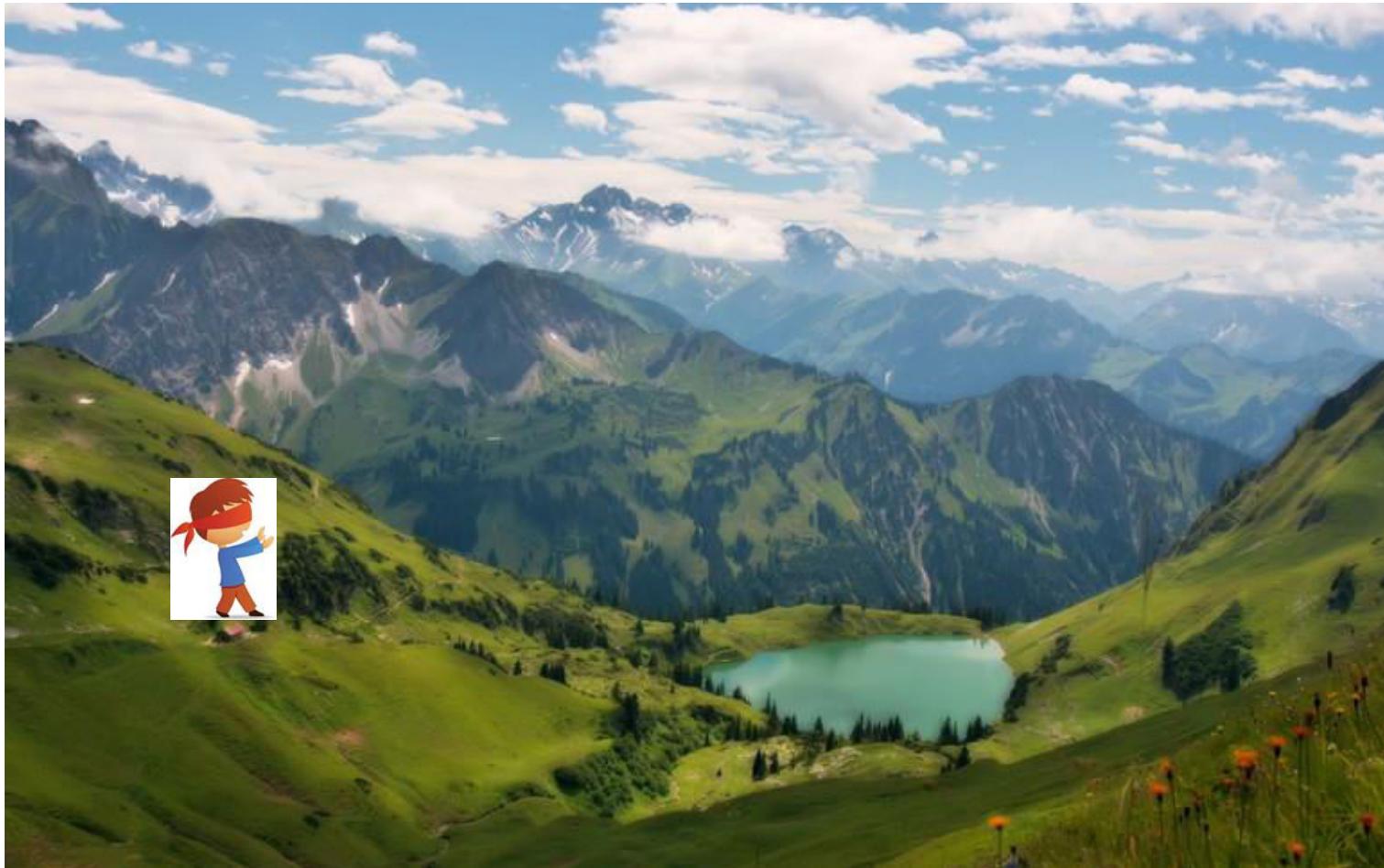
$$L_i = -\log\left(\frac{e^{sy_i}}{\sum_j e^{sj}}\right) \quad \text{Softmax}$$

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1) \quad \text{SVM}$$

$$L = \frac{1}{N} \sum_{i=1}^N L_i + R(W) \quad \text{Cu regularizare}$$



# Algoritm: Coborârea pe gradient



## Algoritm: Coborârea pe gradient

Într-o singură dimensiune, derivata unei funcții este:

$$\frac{df(x)}{dx} = \lim_{h \rightarrow 0} \frac{f(x + h) - f(x)}{h}$$

În mai multe dimensiuni, **gradientul** este un vector cu derivate parțiale.

**W actual:**

[0.34,  
-1.11,  
0.78,  
0.12,  
0.55,  
2.81,  
-3.1,  
-1.5,  
0.33,...]

**loss 1.25347**

**W + h (dim 1):**

[0.34 + 0.0001,  
-1.11,  
0.78,  
0.12,  
0.55,  
2.81,  
-3.1,  
-1.5,  
0.33,...]

**loss 1.25322**

**gradientul dW:**

[?,  
?,  
?,  
?,  
?,  
?,  
?,  
?,  
?,  
?,...]

**W actual:**

[0.34,  
-1.11,  
0.78,  
0.12,  
0.55,  
2.81,  
-3.1,  
-1.5,  
0.33,...]

**loss 1.25347**

**W + h (dim 1):**

[0.34 + 0.0001,  
-1.11,  
0.78,  
0.12,  
0.55,  
2.81,  
-3.1,  
-1.5,  
0.33,...]

**loss 1.25322**

**gradientul dW:**

**-2.5,**

**?,**

**?,**

$$\frac{(1.25322 - 1.25347)}{0.0001} = -2.5$$

$$\frac{df(x)}{dx} = \lim_{h \rightarrow 0} \frac{f(x + h) - f(x)}{h}$$

**?,**

**?,...]**

**W actual:**

[0.34,  
-1.11,  
0.78,  
0.12,  
0.55,  
2.81,  
-3.1,  
-1.5,  
0.33,...]

**loss 1.25347**

**W + h (dim 2):**

[0.34,  
-1.11 + **0.0001**,  
0.78,  
0.12,  
0.55,  
2.81,  
-3.1,  
-1.5,  
0.33,...]

**loss 1.25353**

**gradientul dW:**

[-2.5,  
?,  
?,  
?,  
?,  
?,  
?,  
?,  
?,  
?,...]

**W actual:**

[0.34,  
-1.11,  
0.78,  
0.12,  
0.55,  
2.81,  
-3.1,  
-1.5,  
0.33,...]

**loss 1.25347**

**W + h (dim 2):**

[0.34,  
-1.11 + **0.0001**,  
0.78,  
0.12,  
0.55,  
2.81,  
-3.1,  
-1.5,  
0.33,...]

**loss 1.25353**

**gradientul dW:**

**[-2.5,**  
**0.6,** ↗  
?,  
?.

$$\frac{(1.25353 - 1.25347)}{0.0001} = 0.6$$

$$\frac{df(x)}{dx} = \lim_{h \rightarrow 0} \frac{f(x + h) - f(x)}{h}$$

?,...]

# Evaluarea gradientului

## 1) Metoda numerică

Alegem un  $h$  pozitiv aproape de 0 și folosim formula:

$$\frac{df(x)}{dx} = \lim_{h \rightarrow 0} \frac{f(x + h) - f(x)}{h}$$

- Obținem o valoare aproximativă
- Foarte încet de calculat

## 2) Metoda analitică

Folosim analiza numerică pentru a determina formula gradientului în funcție  $X$  și  $W$

# Evaluarea gradientului (Python)

```
def f(x):
    y = 0.5 * (x**4) - 2 * (x**2) + x + 5
    return y
```

# 1) Metoda numerică

$h = 0.001$

$\text{gradient} = (f(x + h) - f(x)) / h$

# 2) Metoda analitică

```
def f_prime(x):
    y_prime = 2 * (x**3) - 4 * x + 1
    return y_prime
```

$\text{gradient} = f'_\text{prime}(x)$

**W actual:**

[0.34,  
-1.11,  
0.78,  
0.12,  
0.55,  
2.81,  
-3.1,  
-1.5,  
0.33,...]

**loss 1.25347**

**gradientul dW:**

[-2.5,  
0.6,  
0,  
0.2,  
0.7,  
-0.5,  
1.1,  
1.3,  
-2.1,...]

dW = ...  
(o funcție de x și W)



# În concluzie:

- Gradientul numeric: aproximativ, încet, ușor de scris
- Gradientul analitic: exact, rapid, înclinat spre greșeli

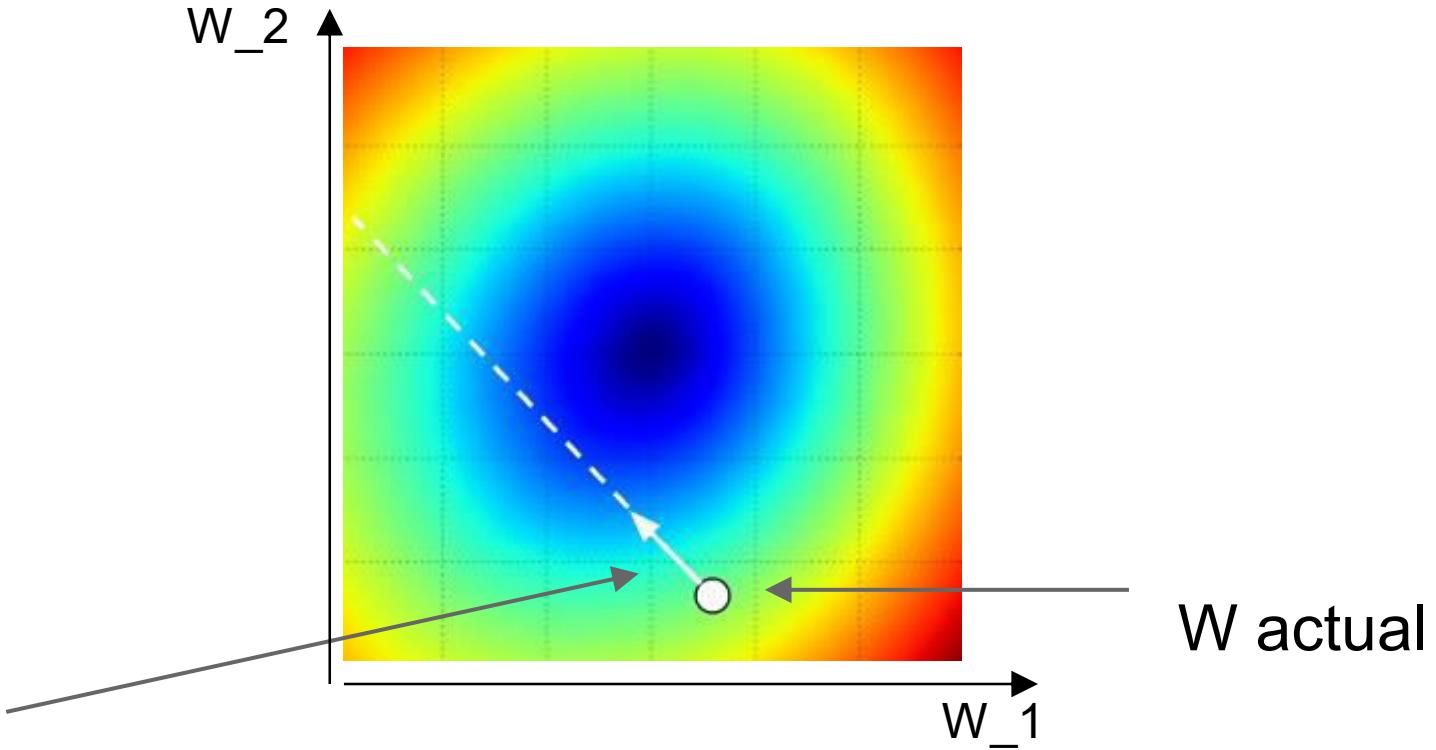
=>

În practică: Folosim întotdeauna gradientul analitic, dar verificăm implementarea cu gradientul numeric. Acest proces se numește **verificarea gradientului (gradient checking)**

# Algorimtul coborârii pe gradient (Python)

```
def GD(W0, X, goal, learningRate):
    perfGoalNotMet = true
    W = W0

    while perfGoalNotMet:
        gradient = eval_gradient(X, W)
        W_old = W
        W = W - learningRate * gradient
        perfGoalNotMet = sum(abs(W - W_old)) > goal
```



directia negativă a gradientului

# Coborârea pe gradient cu mini-batch

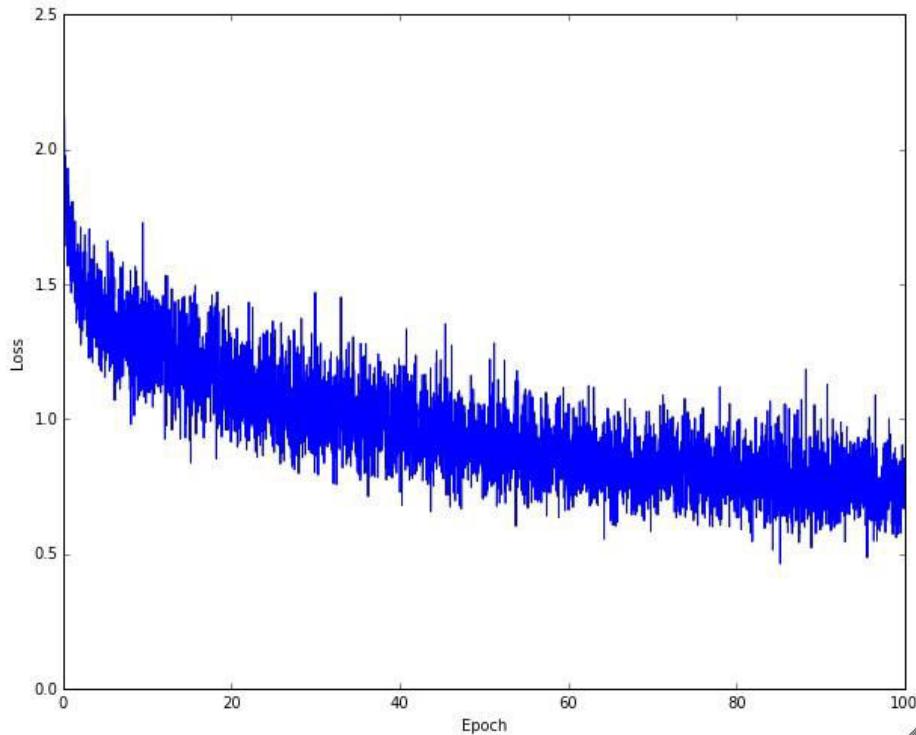
Utilizăm doar o mică parte a mulțimii de antrenare pentru a calcula gradientul:

```
...  
while perfGoalNotMet:
```

```
    X_batch = select_random_subsample(X)  
    gradient = eval_gradient(@loss, X_batch, W)
```

```
    ...
```

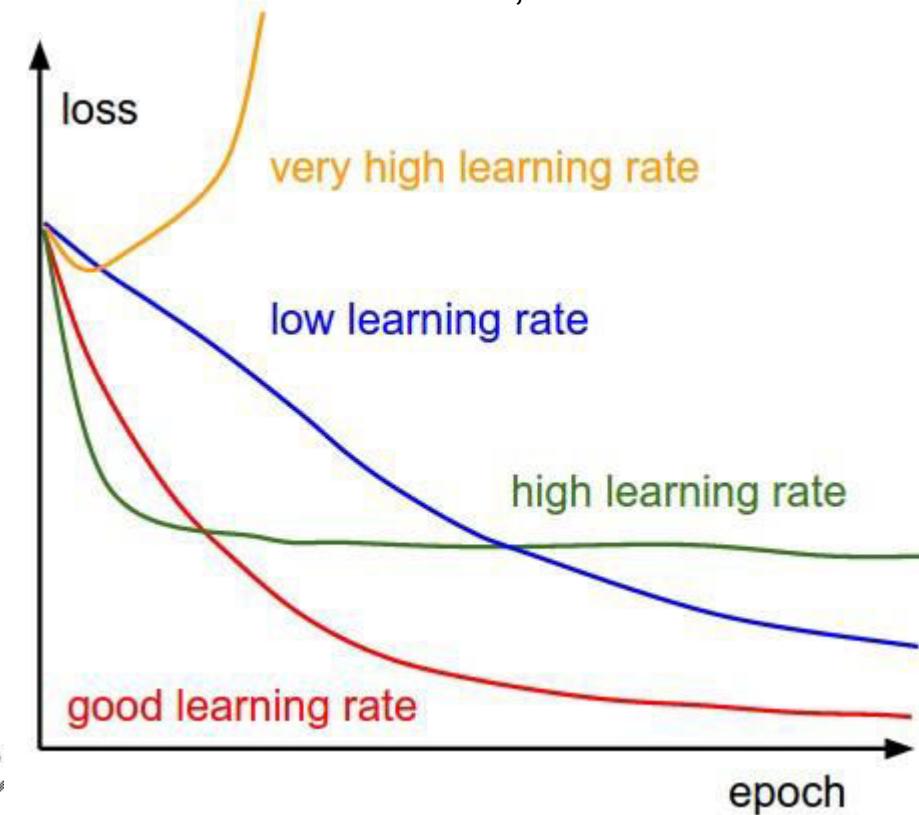
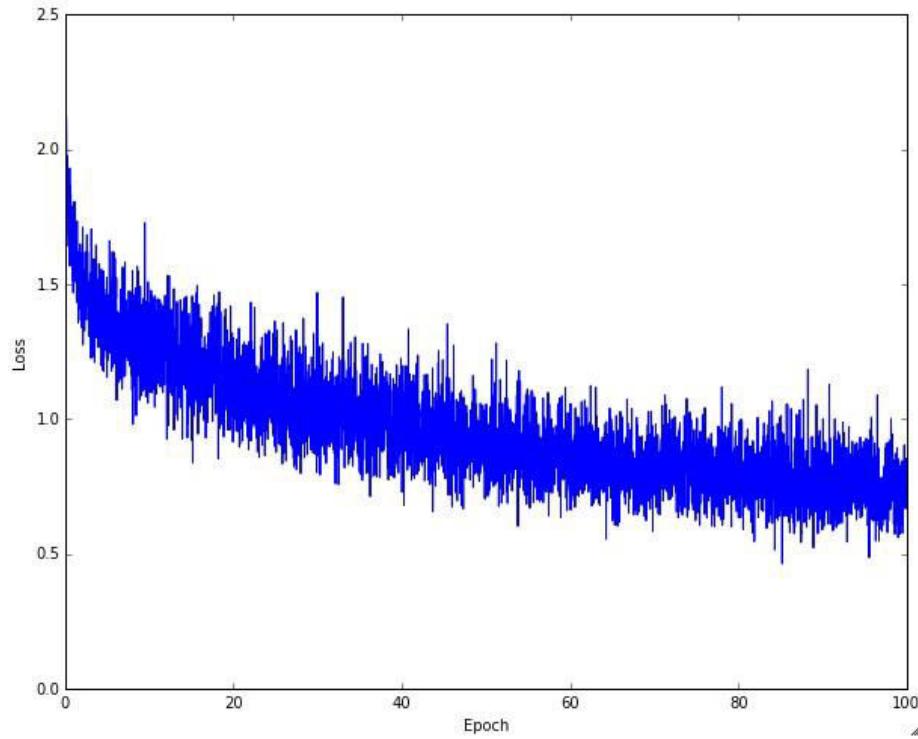
Mărimea mini-batch-ului este de obicei formată din 64/128/256 exemple  
e.g. AlexNet (Krizhevsky ILSVRC ConvNet) folosește 256 exemple



Exemplu de progres al optimizării în timpul antrenării unei rețele neuronale.

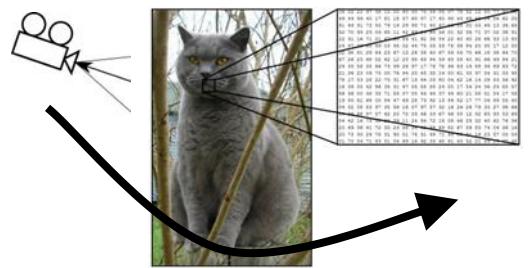
(Funcția de pierdere calculată pe mini-batch-uri scade în timp)

## Efectele ratei de învățare



# Varietatea intra-clasă

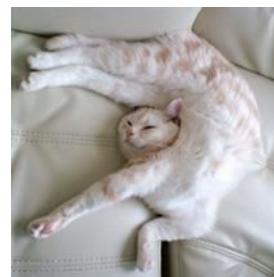
Poziția camerei



Iluminare



Deformare



Ocluzie



Background confuz



Variatăie intra-clasă



# Similaritatea inter-clasă



# De la extragere “manuală” către învățare

vector ce descrie statistici despre imagine, e.g. bag-of-visual-words



Extragere de  
trăsături

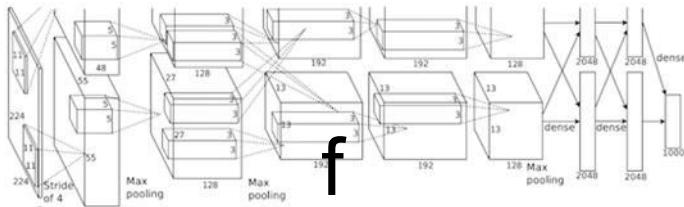


$f$

învățare

$N$  numere ce indică scorurile pentru fiecare clasă

[32x32x3]



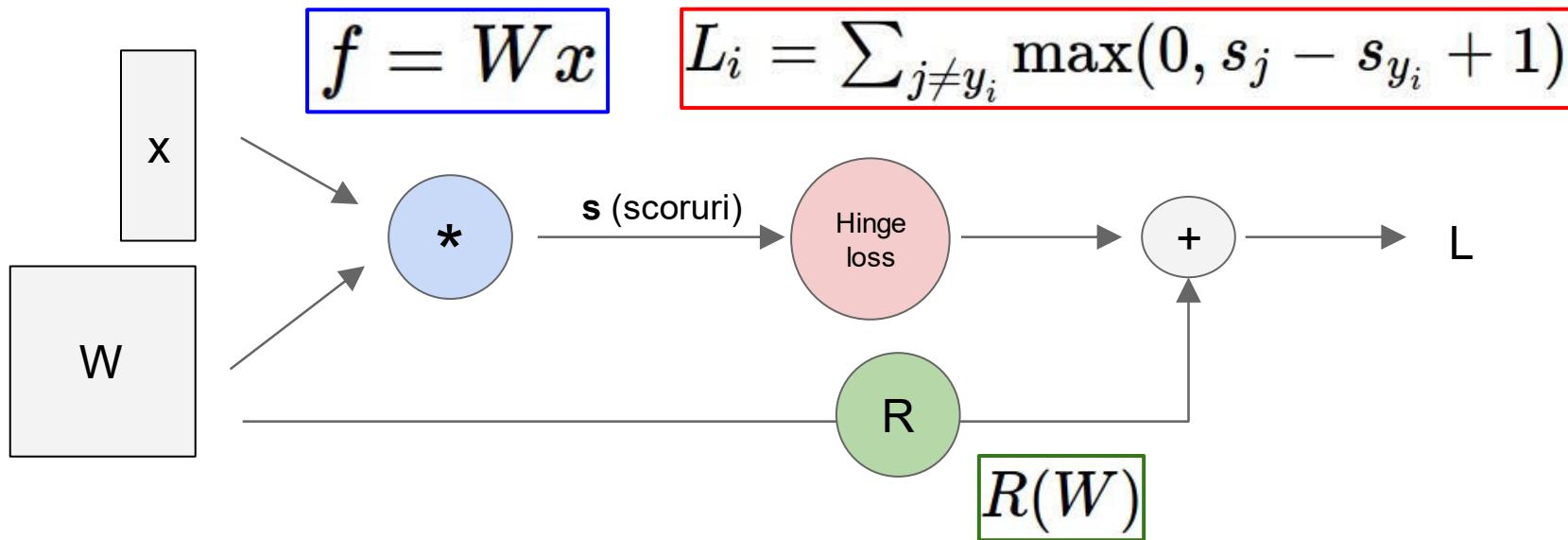
învățare “end-to-end”

$N$  numere ce indică scorurile pentru fiecare clasă

[32x32x3]



# Privim algoritmul ca un graf computational

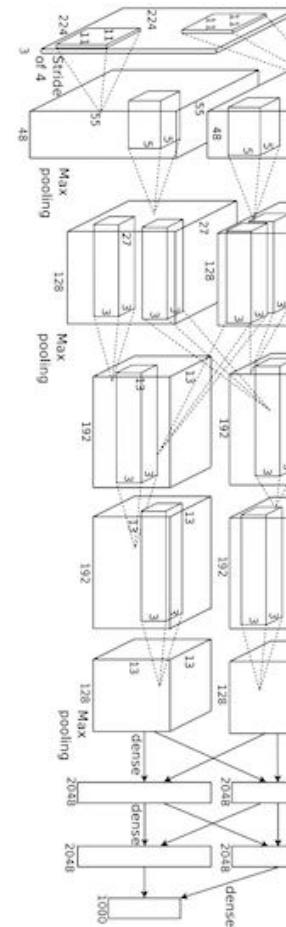


# Rețea convețională (AlexNet)

imagine de input

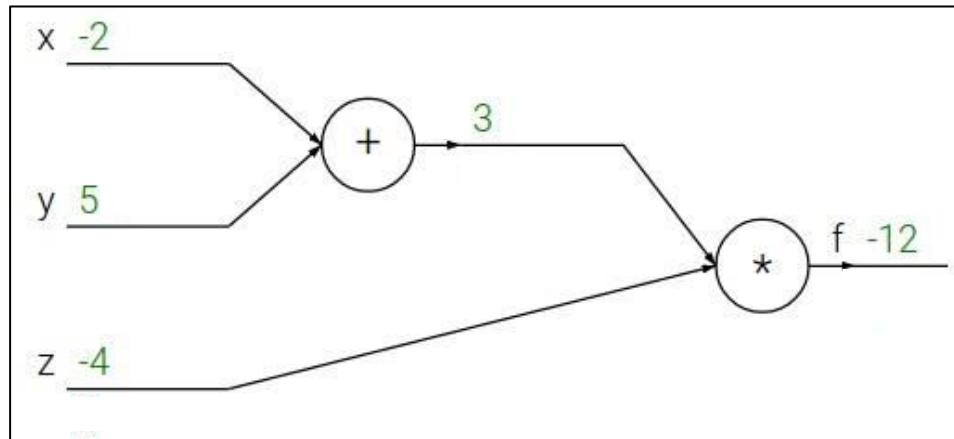
ponderi

funcție de pierdere



$$f(x, y, z) = (x + y)z$$

e.g.  $x = -2, y = 5, z = -4$



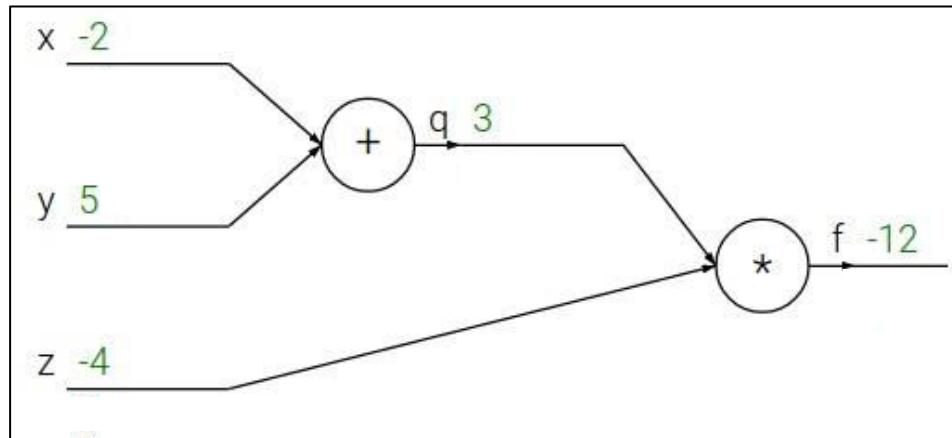
$$f(x, y, z) = (x + y)z$$

e.g.  $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

vrem:  $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



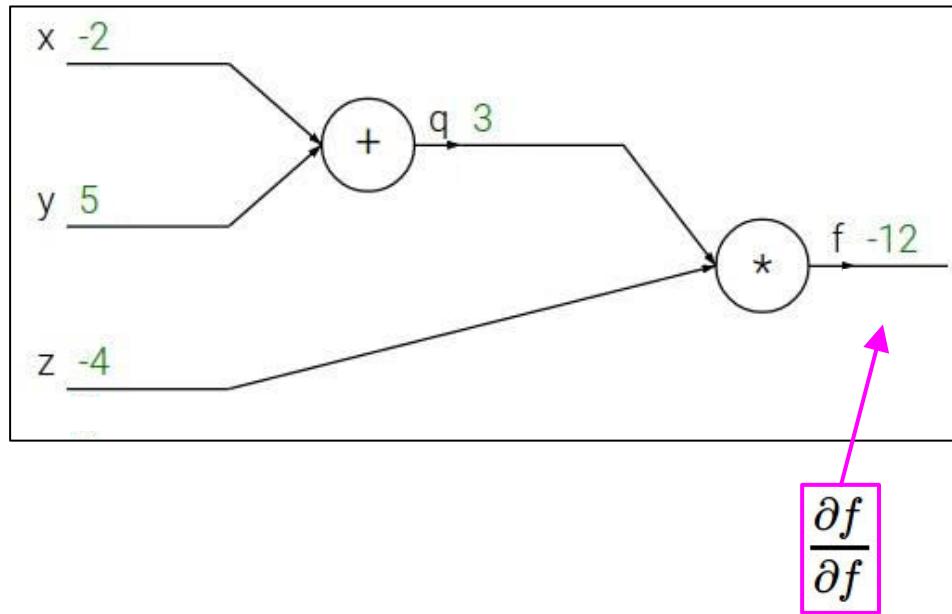
$$f(x, y, z) = (x + y)z$$

e.g.  $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

vrem:  $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



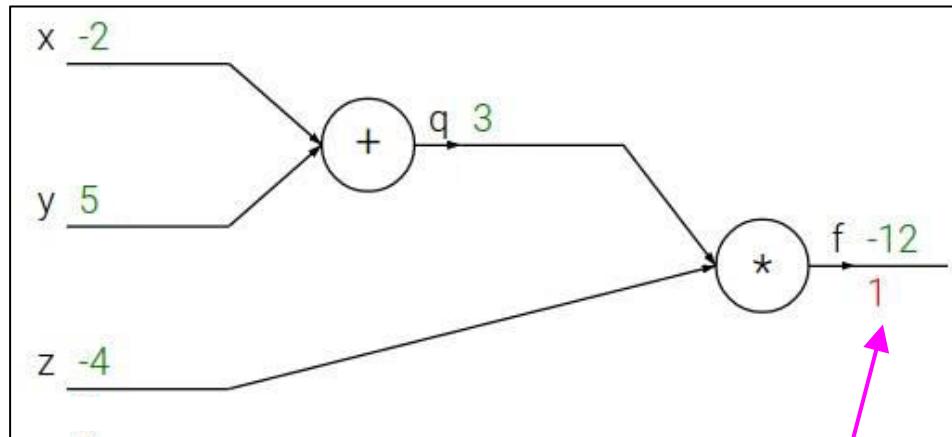
$$f(x, y, z) = (x + y)z$$

e.g.  $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

vrem:  $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



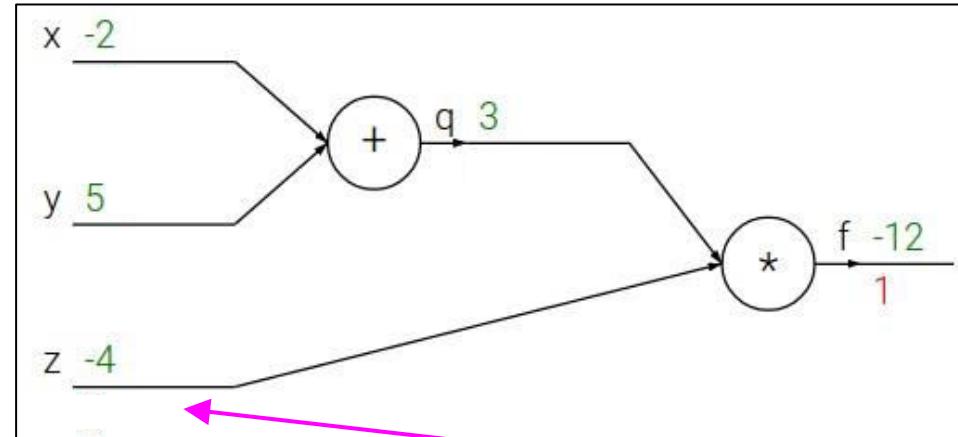
$$\frac{\partial f}{\partial f}$$

$$f(x, y, z) = (x + y)z$$

e.g.  $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$



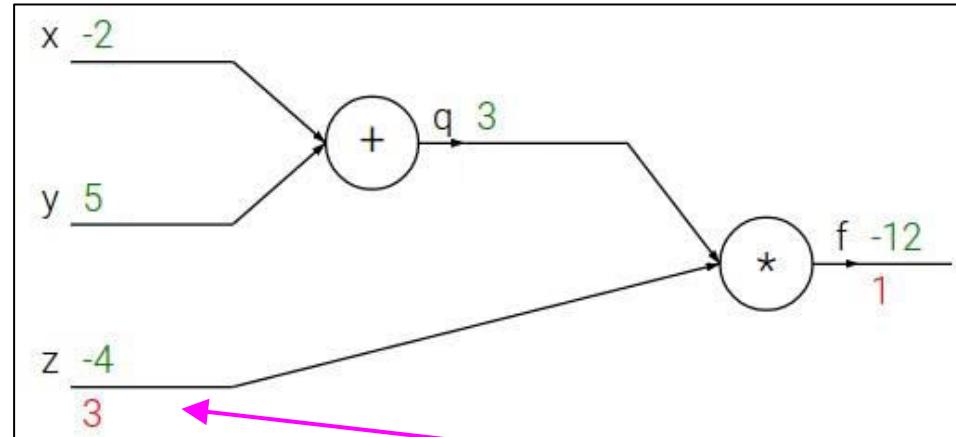
vrem:  $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

$$f(x, y, z) = (x + y)z$$

e.g.  $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$



vrem:  $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

$$\frac{\partial f}{\partial z}$$

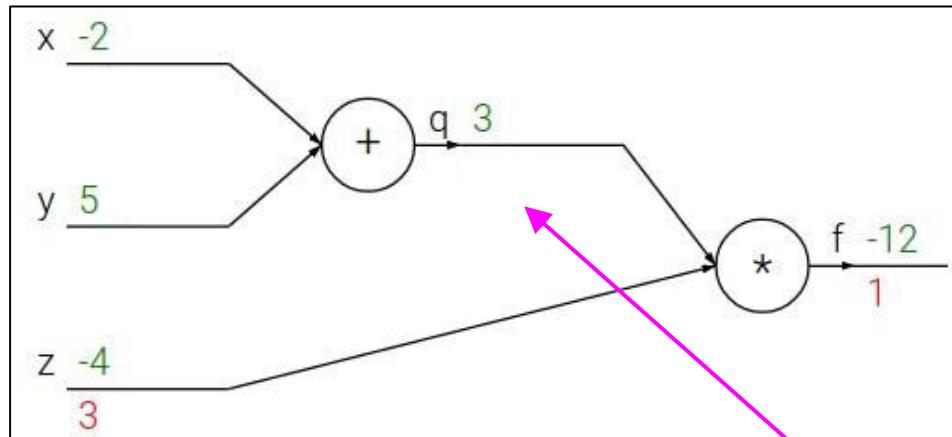
$$f(x, y, z) = (x + y)z$$

e.g.  $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

vrem:  $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$$\frac{\partial f}{\partial q}$$

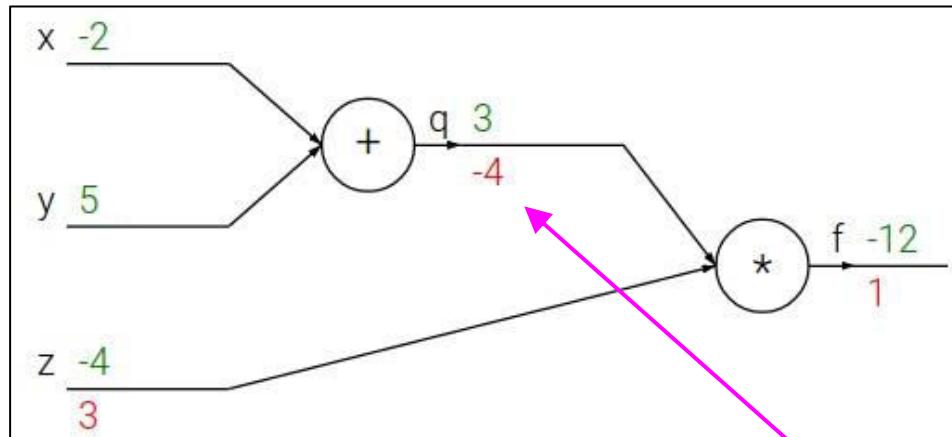
$$f(x, y, z) = (x + y)z$$

e.g.  $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

vrem:  $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$$\frac{\partial f}{\partial q}$$

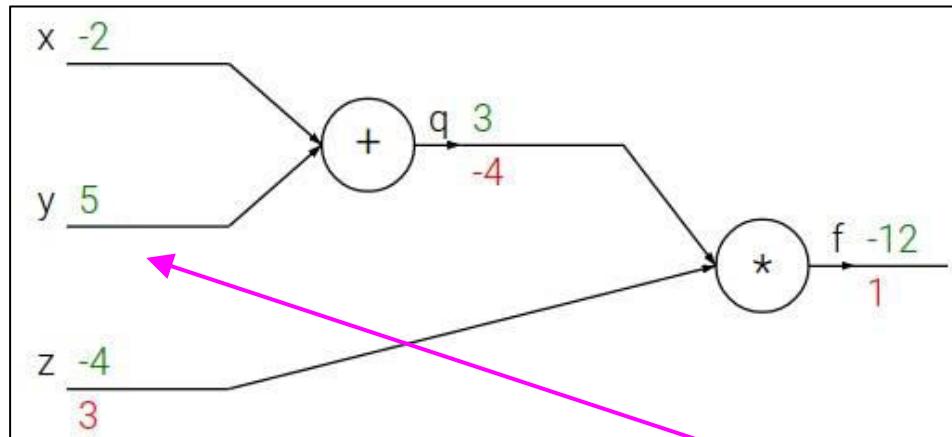
$$f(x, y, z) = (x + y)z$$

e.g.  $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

vrem:  $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$$\frac{\partial f}{\partial y}$$

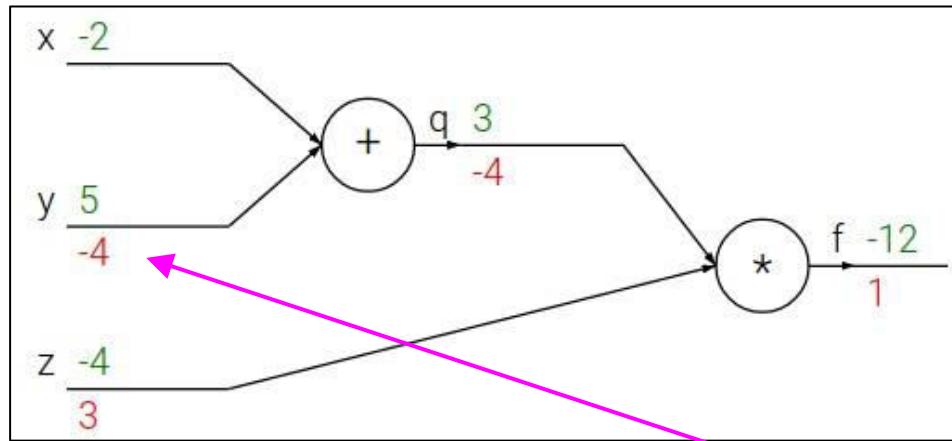
$$f(x, y, z) = (x + y)z$$

e.g.  $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

vrem:  $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



Regula de înlățuire:

$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial y}$$

$$\boxed{\frac{\partial f}{\partial y}}$$

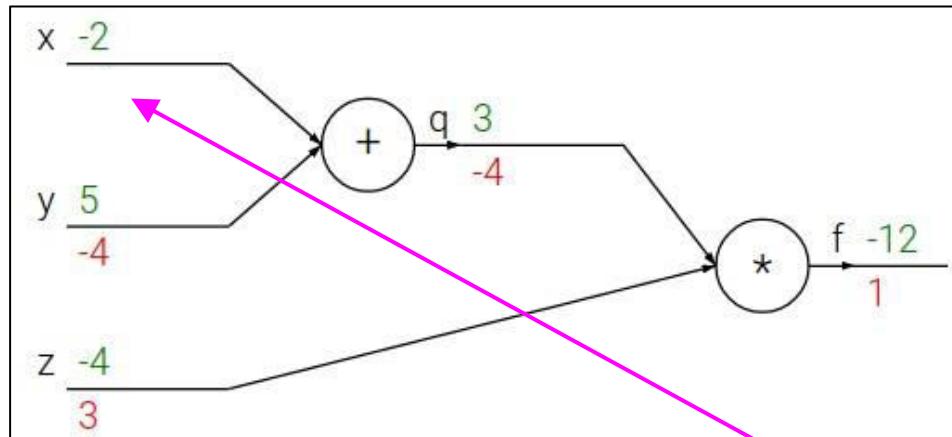
$$f(x, y, z) = (x + y)z$$

e.g.  $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

vrem:  $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



$$\frac{\partial f}{\partial x}$$

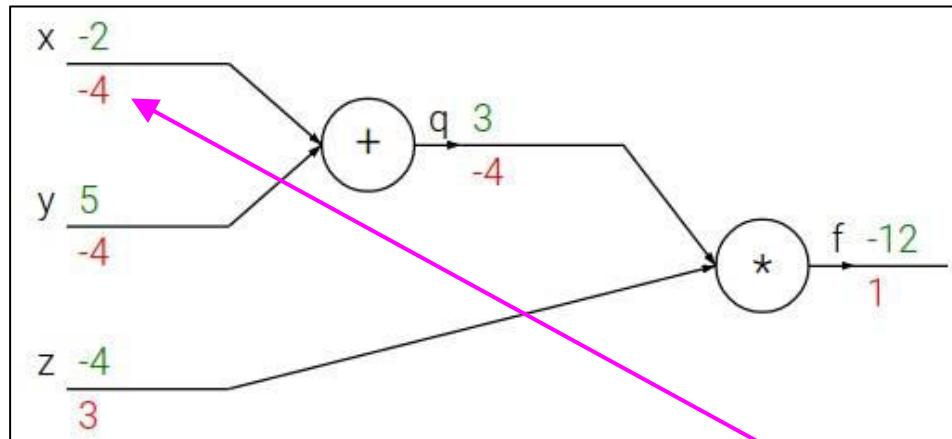
$$f(x, y, z) = (x + y)z$$

e.g.  $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

vrem:  $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

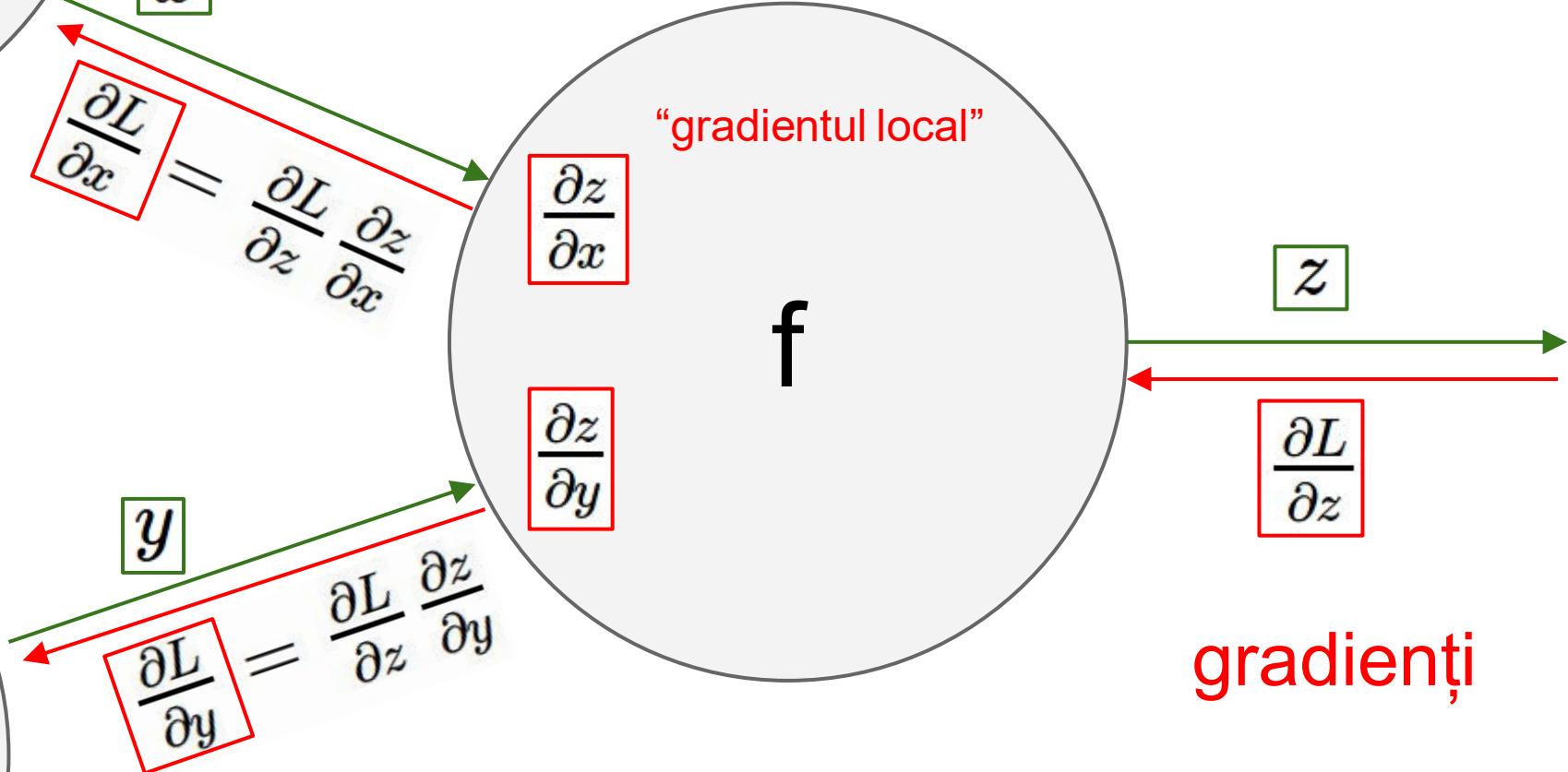


Regula de înlățuire:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial x}$$

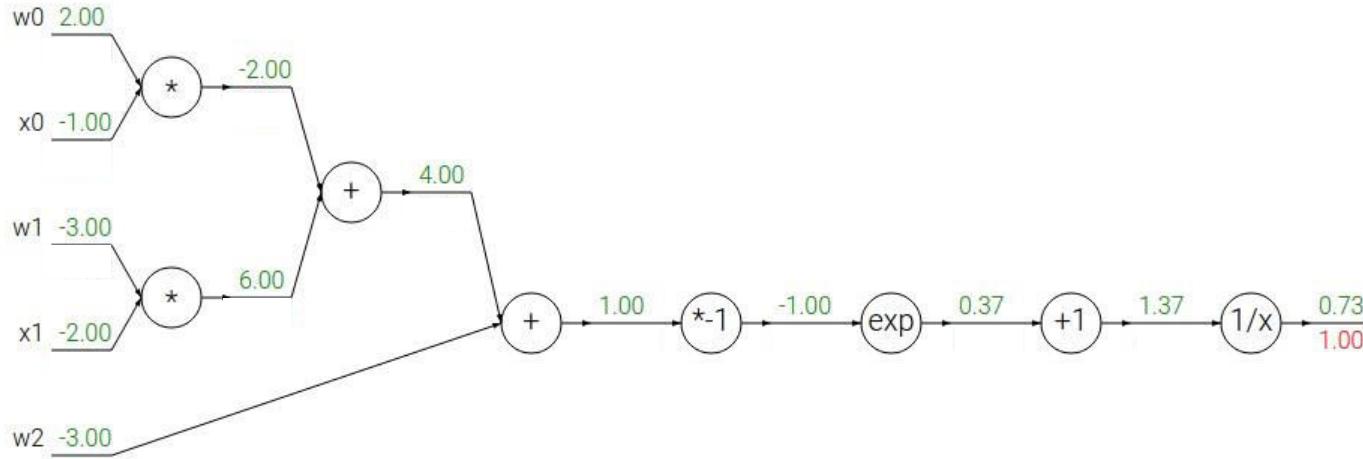
$$\frac{\partial f}{\partial x}$$

# Propagarea gradientului prin regula de înlăntuire activări



Un alt exemplu:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



$$f(x) = e^x$$

→

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

→

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

$$f_c(x) = c + x$$

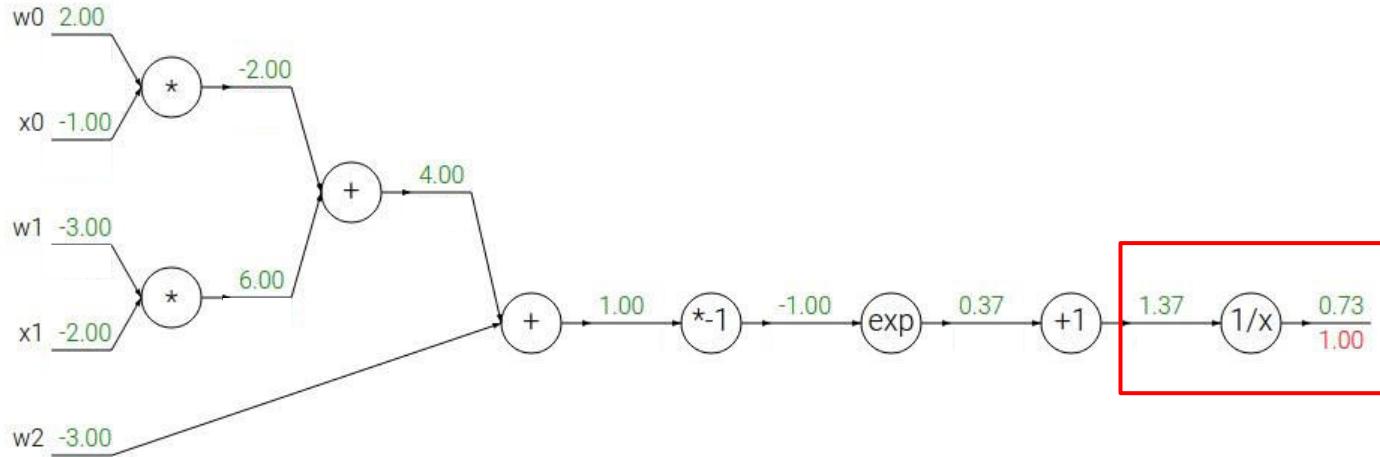
→

$$\frac{df}{dx} = -1/x^2$$

$$\frac{df}{dx} = 1$$

Un alt exemplu:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



$$f(x) = e^x$$

$\rightarrow$

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

$\rightarrow$

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

$\rightarrow$

$$\frac{df}{dx} = -1/x^2$$

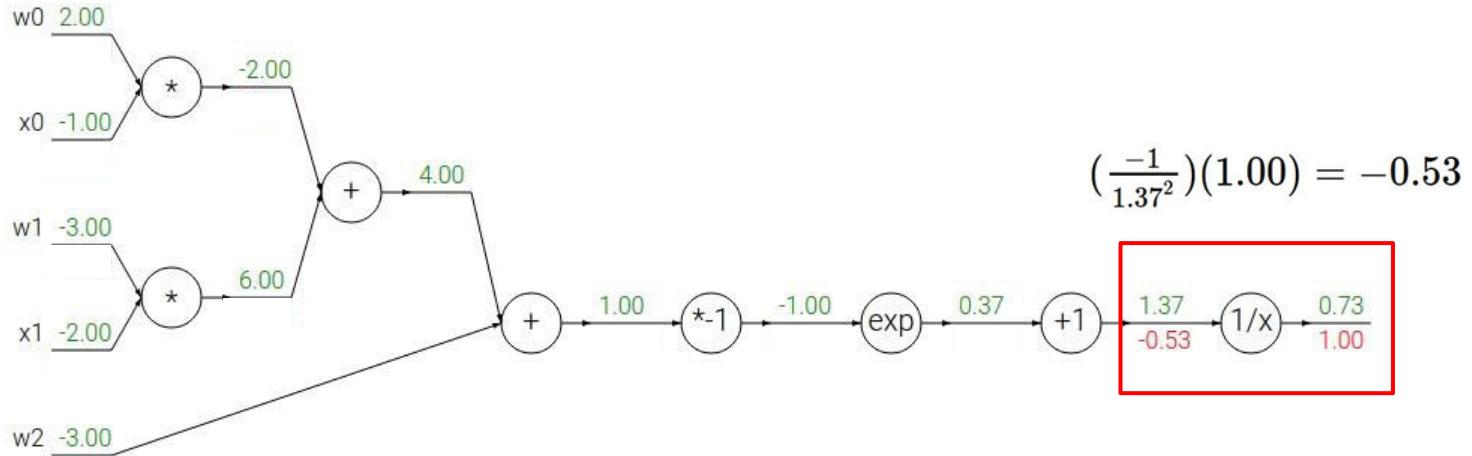
$$f_c(x) = c + x$$

$\rightarrow$

$$\frac{df}{dx} = 1$$

Un alt exemplu:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



$$f(x) = e^x$$

$\rightarrow$

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

$\rightarrow$

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

$\rightarrow$

$$\frac{df}{dx} = -1/x^2$$

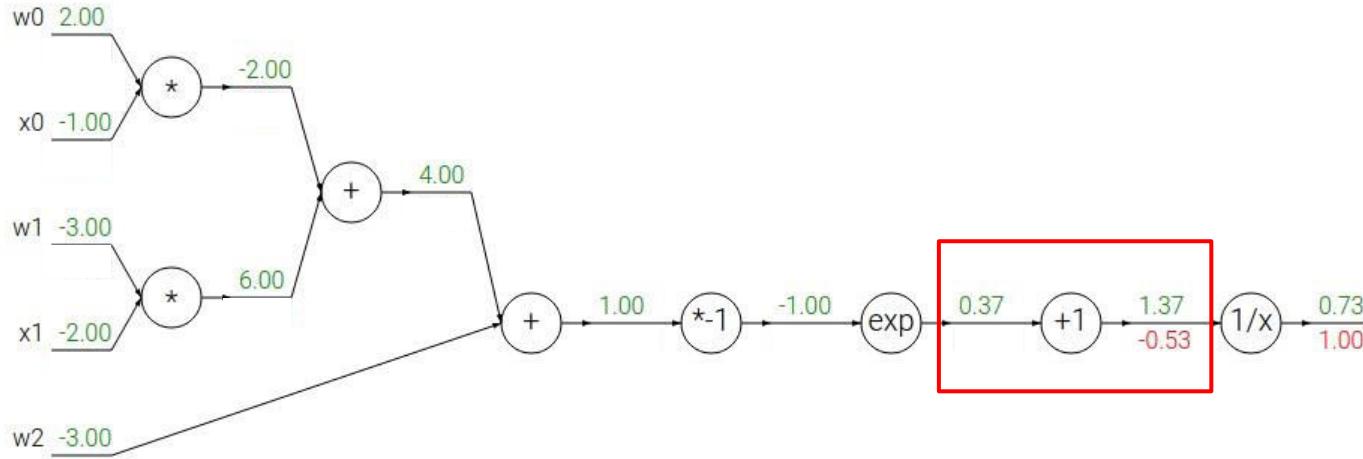
$$f_c(x) = c + x$$

$\rightarrow$

$$\frac{df}{dx} = 1$$

Un alt exemplu:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



$$f(x) = e^x$$

$\rightarrow$

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

$\rightarrow$

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

$\rightarrow$

$$\frac{df}{dx} = -1/x^2$$

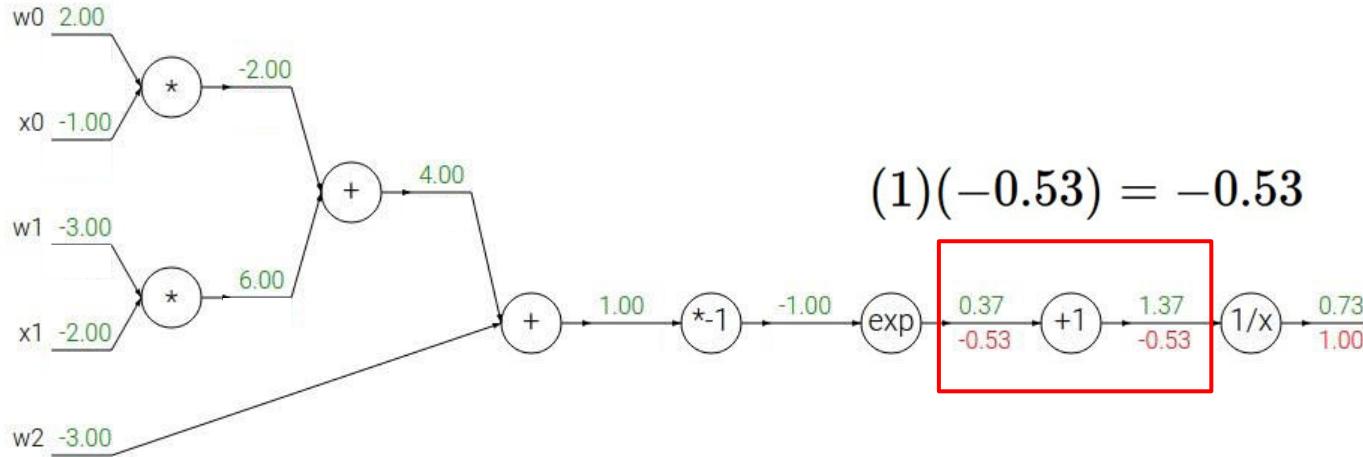
$$f_c(x) = c + x$$

$\rightarrow$

$$\frac{df}{dx} = 1$$

Un alt exemplu:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



$$f(x) = e^x$$

→

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

→

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

→

$$\frac{df}{dx} = -1/x^2$$

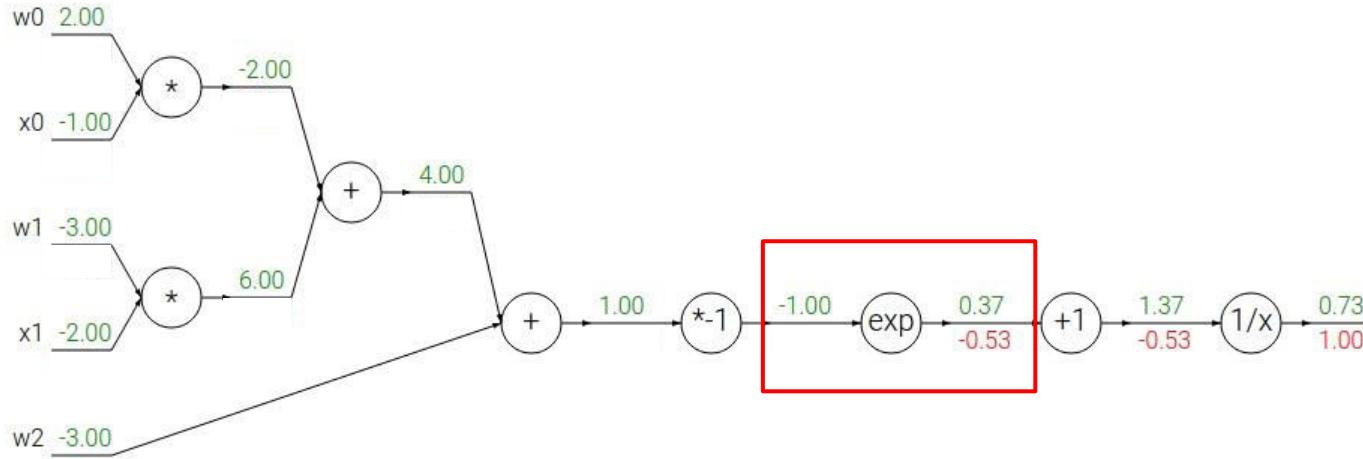
$$f_c(x) = c + x$$

→

$$\frac{df}{dx} = 1$$

Un alt exemplu:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



$$f(x) = e^x$$

$\rightarrow$

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

$\rightarrow$

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

$\rightarrow$

$$\frac{df}{dx} = -1/x^2$$

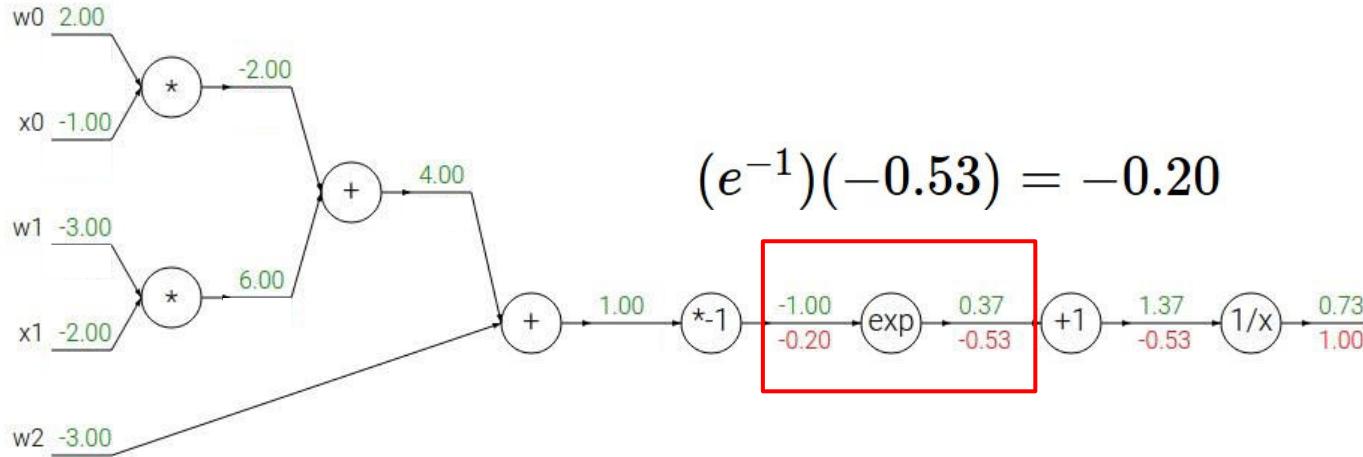
$$f_c(x) = c + x$$

$\rightarrow$

$$\frac{df}{dx} = 1$$

Un alt exemplu:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



$$f(x) = e^x$$

$\rightarrow$

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

$\rightarrow$

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

$\rightarrow$

$$\frac{df}{dx} = -1/x^2$$

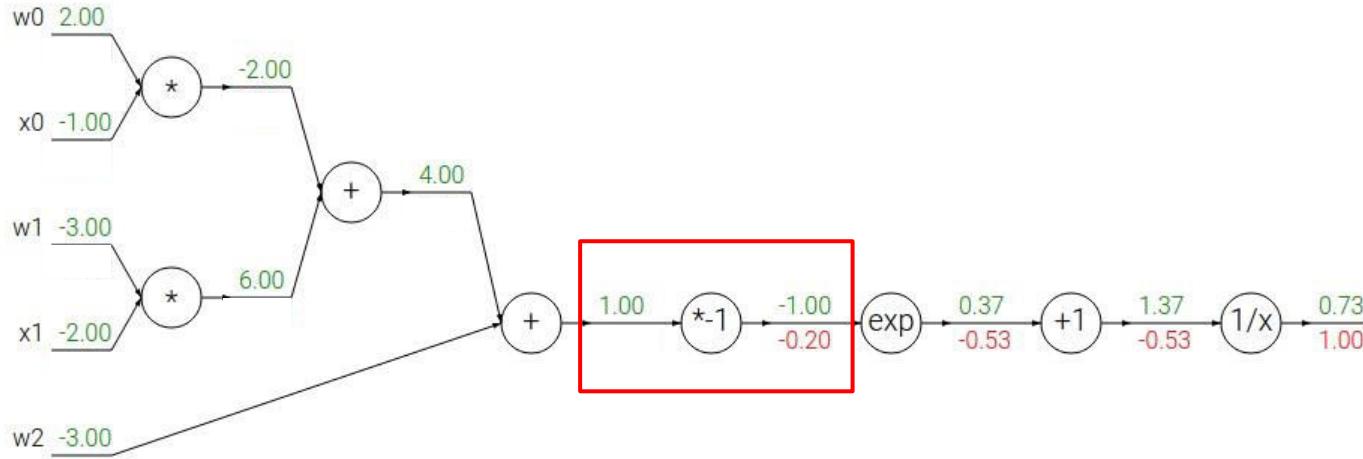
$$f_c(x) = c + x$$

$\rightarrow$

$$\frac{df}{dx} = 1$$

Un alt exemplu:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



$$f(x) = e^x$$

$\rightarrow$

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

$\rightarrow$

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

$$f_c(x) = c + x$$

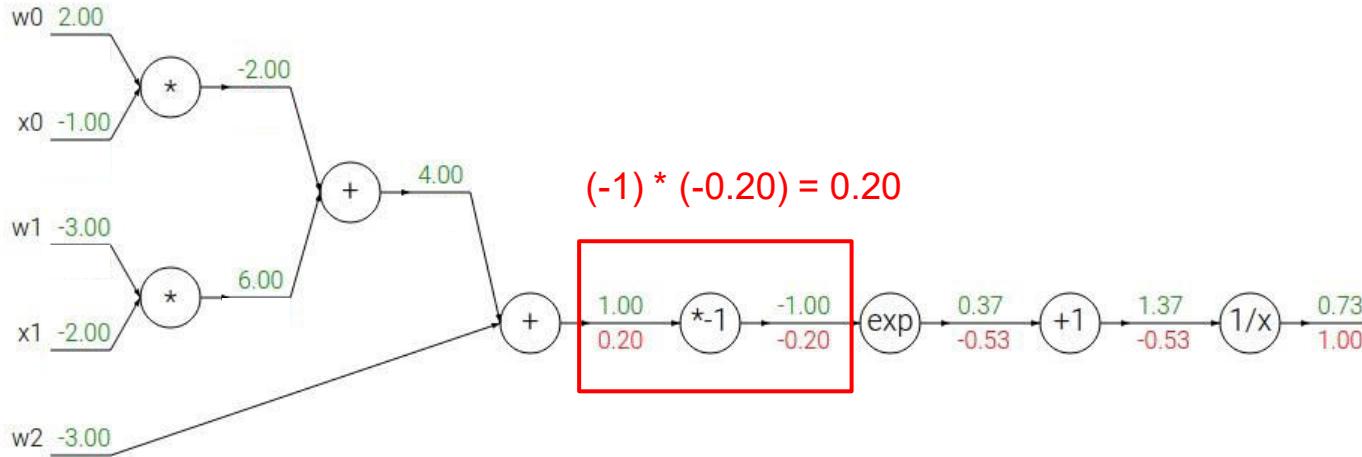
$\rightarrow$

$$\frac{df}{dx} = -1/x^2$$

$$\frac{df}{dx} = 1$$

Un alt exemplu:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



$$f(x) = e^x$$

$\rightarrow$

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

$\rightarrow$

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

$$f_c(x) = c + x$$

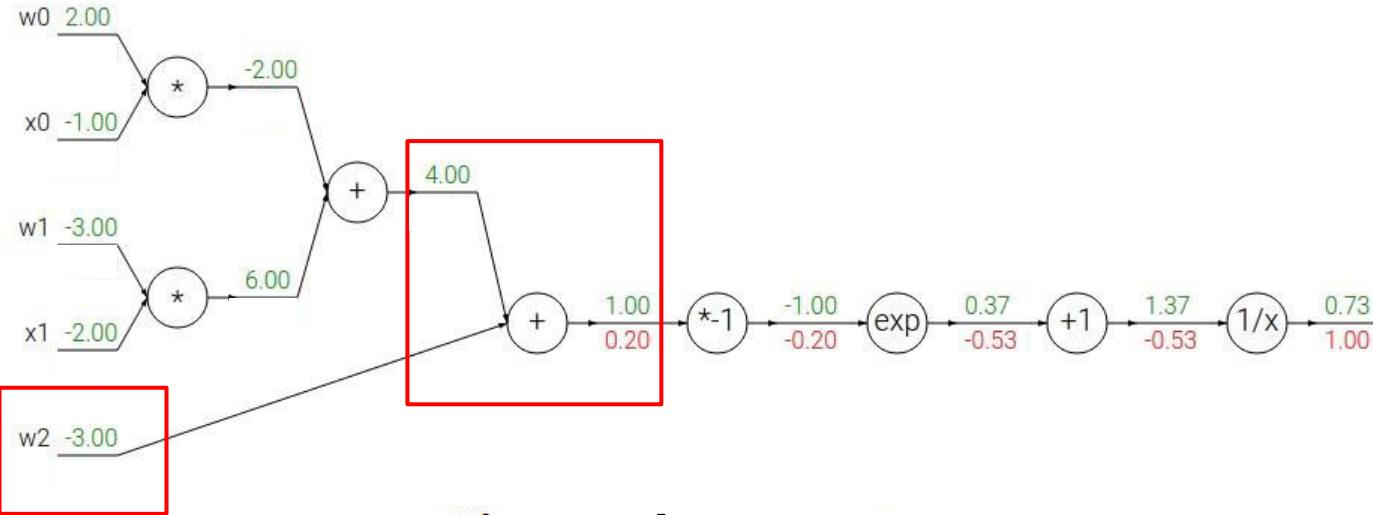
$\rightarrow$

$$\frac{df}{dx} = -1/x^2$$

$$\frac{df}{dx} = 1$$

Un alt exemplu:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



$$f(x) = e^x$$

$\rightarrow$

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

$\rightarrow$

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

$$f_c(x) = c + x$$

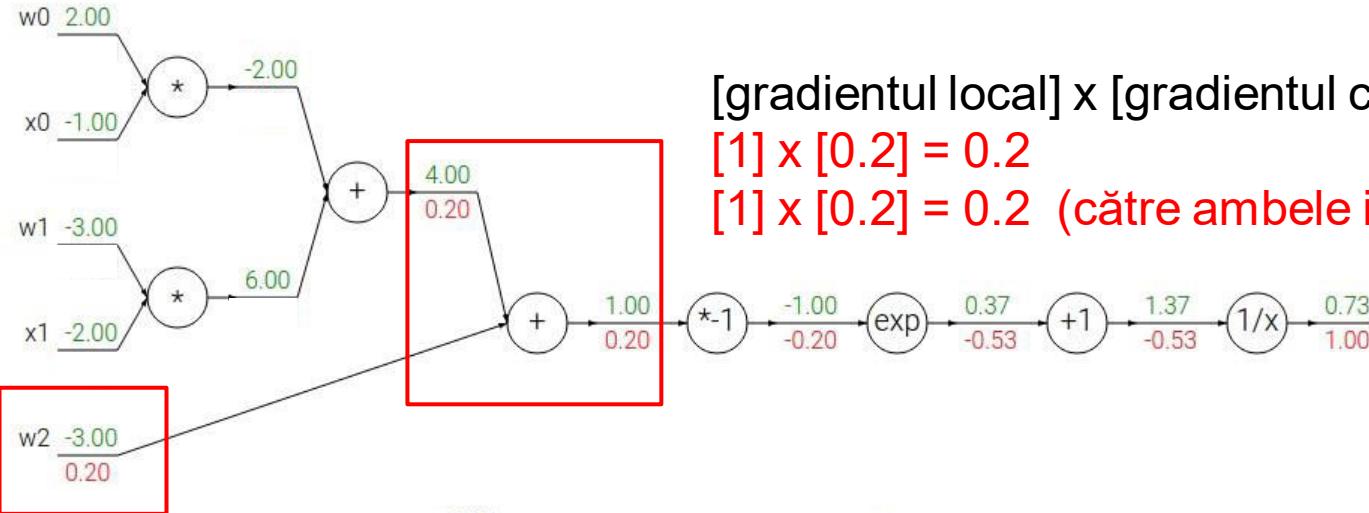
$\rightarrow$

$$\frac{df}{dx} = -1/x^2$$

$$\frac{df}{dx} = 1$$

Un alt exemplu:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



[gradientul local] x [gradientul calculat]

$$[1] \times [0.2] = 0.2$$

[1] x [0.2] = 0.2 (către ambele intrări)

$$f(x) = e^x$$

$\rightarrow$

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

$\rightarrow$

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

$$f_c(x) = c + x$$

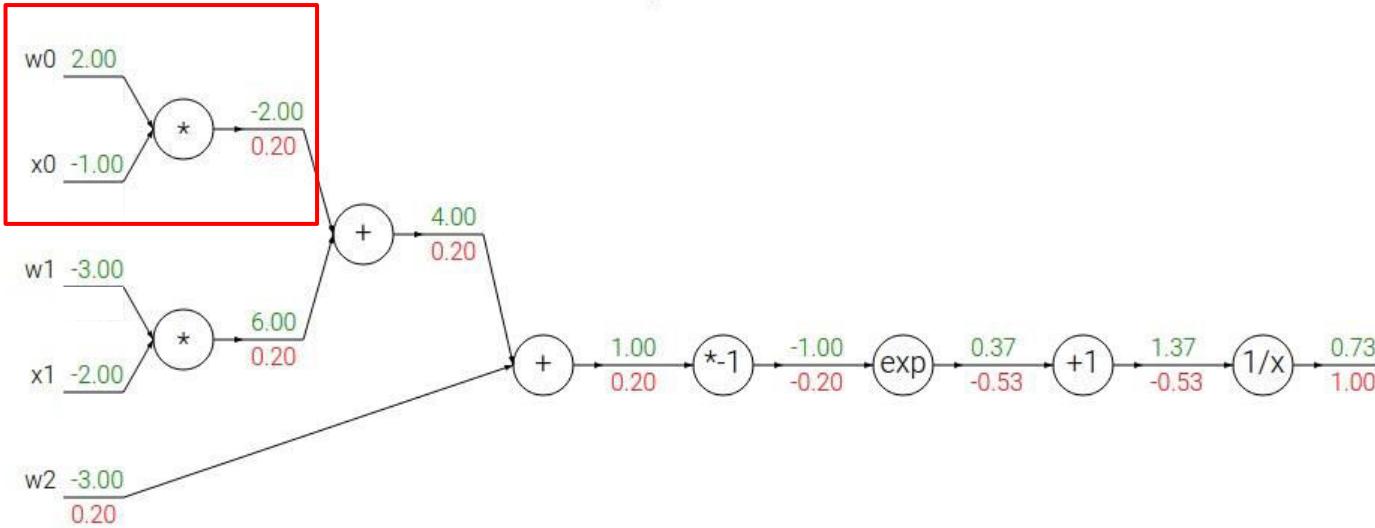
$\rightarrow$

$$\frac{df}{dx} = -1/x^2$$

$$\frac{df}{dx} = 1$$

Un alt exemplu:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



$$f(x) = e^x$$

→

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

→

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

$$f_c(x) = c + x$$

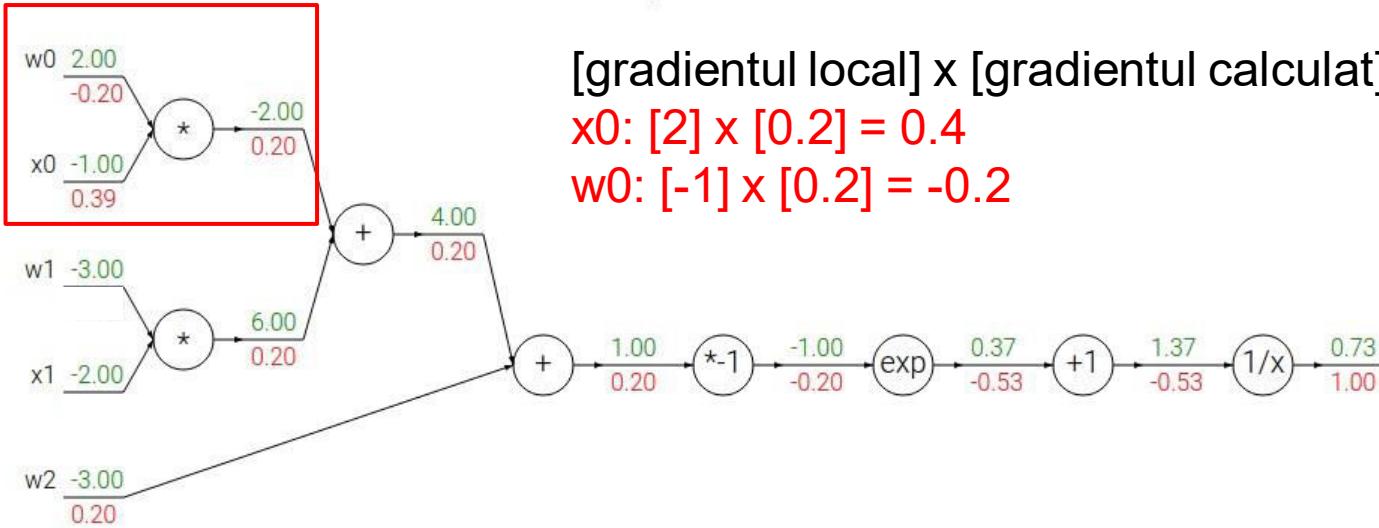
→

$$\frac{df}{dx} = -1/x^2$$

$$\frac{df}{dx} = 1$$

Un alt exemplu:

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$



$$f(x) = e^x$$

$\rightarrow$

$$\frac{df}{dx} = e^x$$

$$f_a(x) = ax$$

$\rightarrow$

$$\frac{df}{dx} = a$$

$$f(x) = \frac{1}{x}$$

$$f_c(x) = c + x$$

$\rightarrow$

$$\frac{df}{dx} = -1/x^2$$

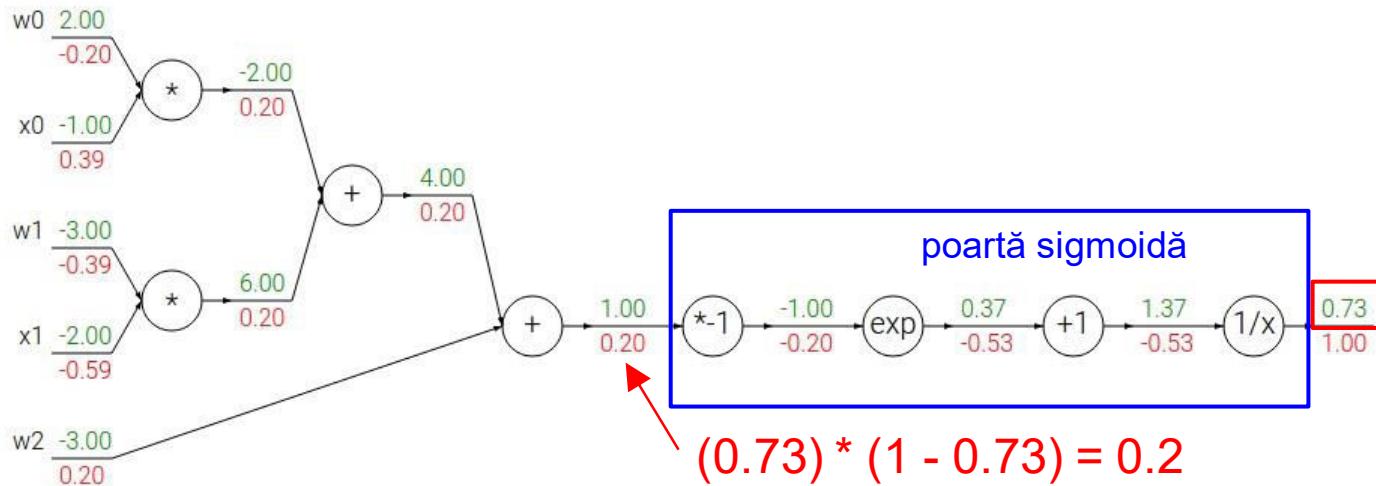
$$\frac{df}{dx} = 1$$

$$f(w, x) = \frac{1}{1 + e^{-(w_0x_0 + w_1x_1 + w_2)}}$$

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

funcția sigmoidă

$$\frac{d\sigma(x)}{dx} = \frac{e^{-x}}{(1 + e^{-x})^2} = \left( \frac{1 + e^{-x} - 1}{1 + e^{-x}} \right) \left( \frac{1}{1 + e^{-x}} \right) = (1 - \sigma(x))\sigma(x)$$

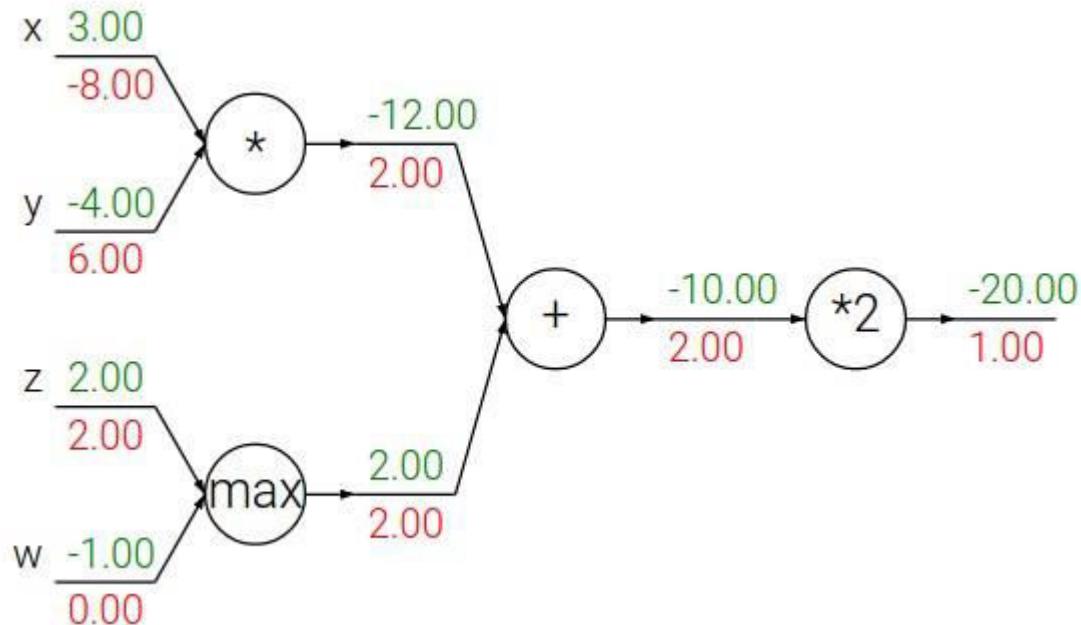


# Tipuri ce apar în propagarea înapoi a gradientului

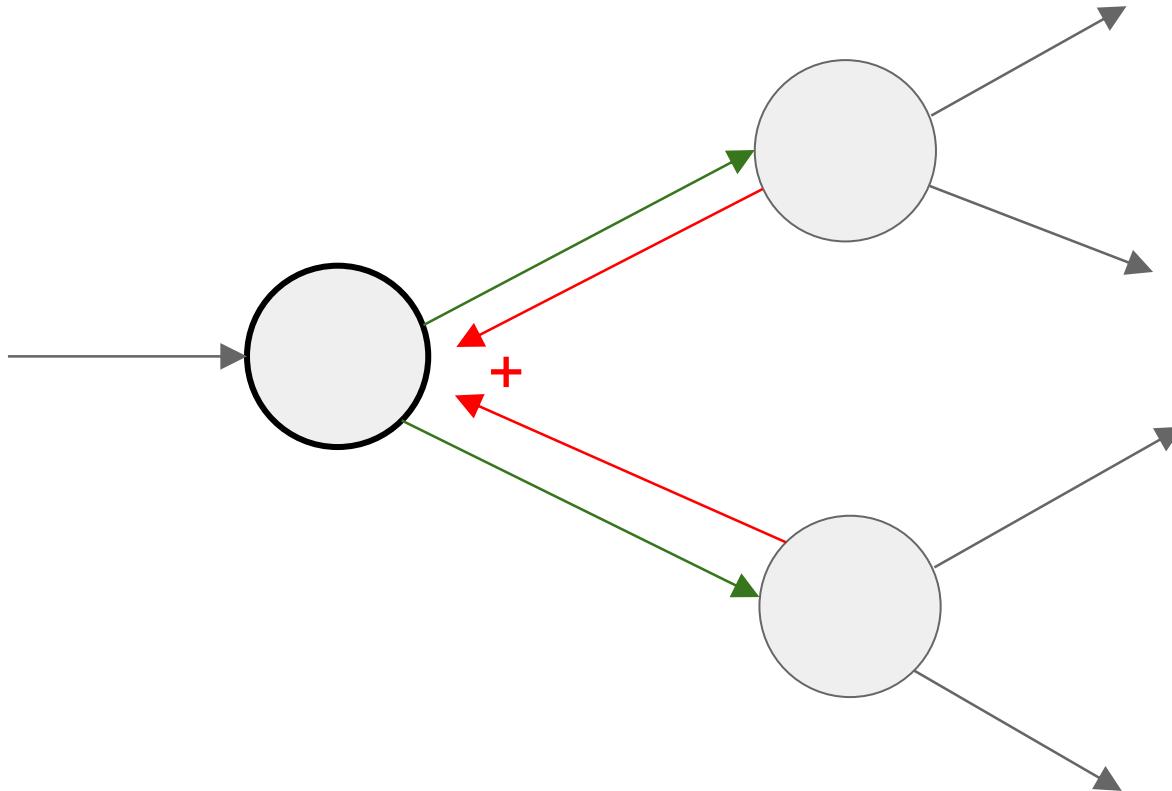
**Poartă add:** distribuie gradientul

**Poartă max:** rutează gradientul

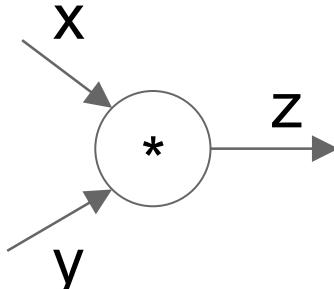
**Poartă ori:** comută gradientul



Atunci când se ramifică, gradienții se adună



# Propagare înainte/înapoi pentru poarta ori (Python)



( $x$ ,  $y$ ,  $z$  sunt scalari)

```
def forward(x,y):
```

$$z = x * y$$

```
layer.input = [x, y] # pt. backward
```

```
return z
```

$$\frac{\partial L}{\partial z}$$

```
def backward(dz):
```

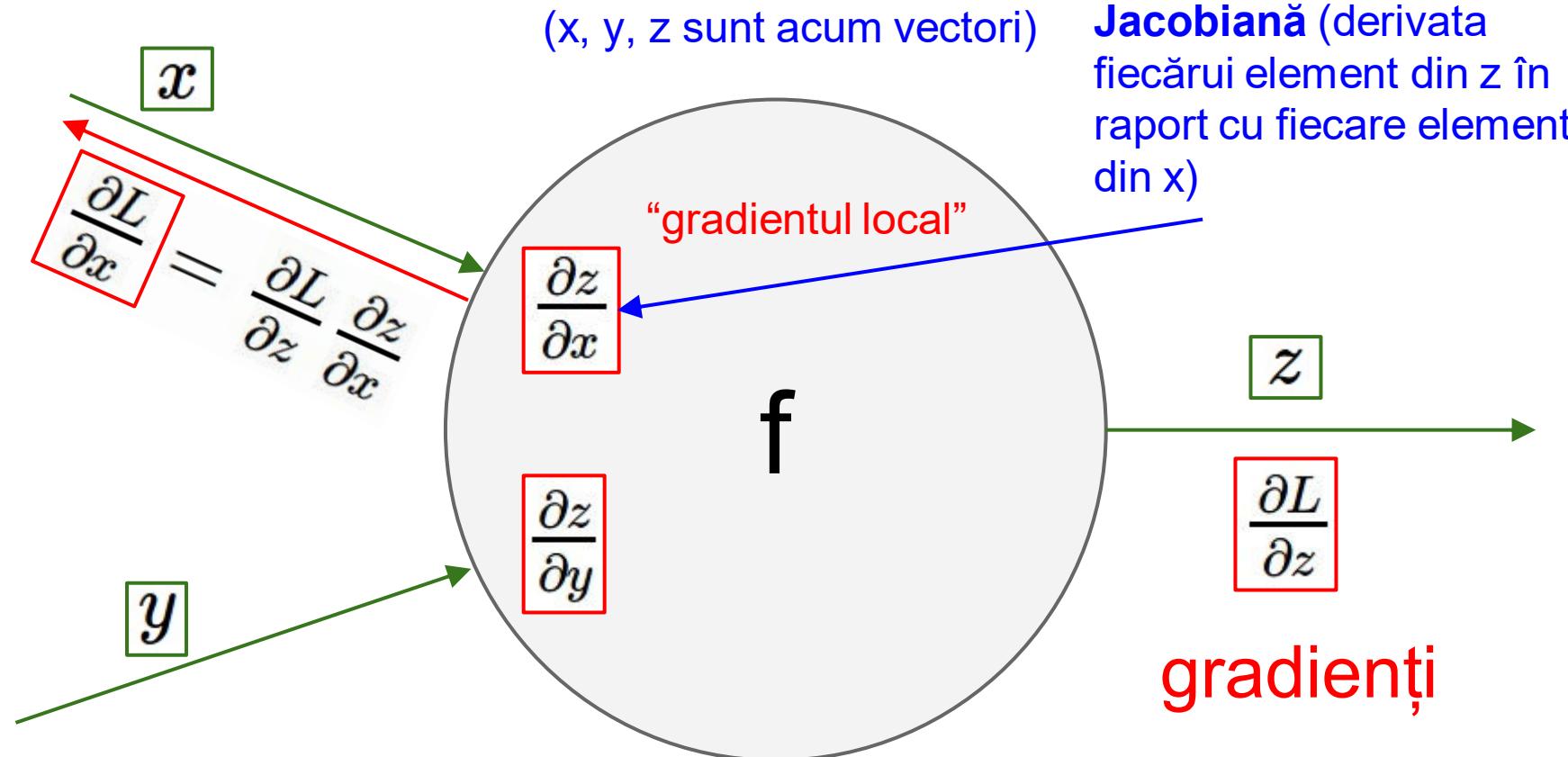
$$dx = layer.input[1] * dz \# dz/dx * dL/dz$$

$$dy = layer.input[0] * dz \# dz/dy * dL/dz$$

```
return [dx, dy]
```

$$\frac{\partial L}{\partial x}$$

# Gradienții pentru cod vectorial

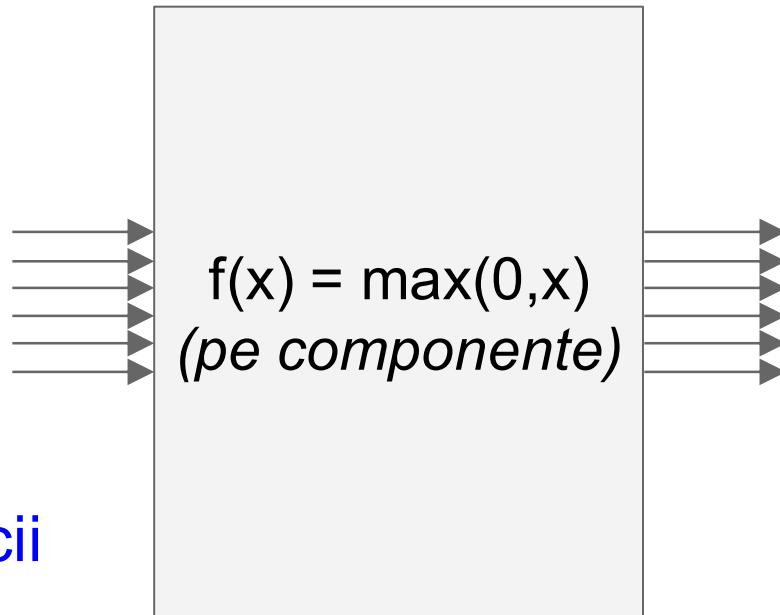


# Operații vectoriale

$$\frac{\partial L}{\partial x} = \boxed{\frac{\partial f}{\partial x}} \frac{\partial L}{\partial f}$$

matricea Jacobiană

Vector de input  
4096-dimensional



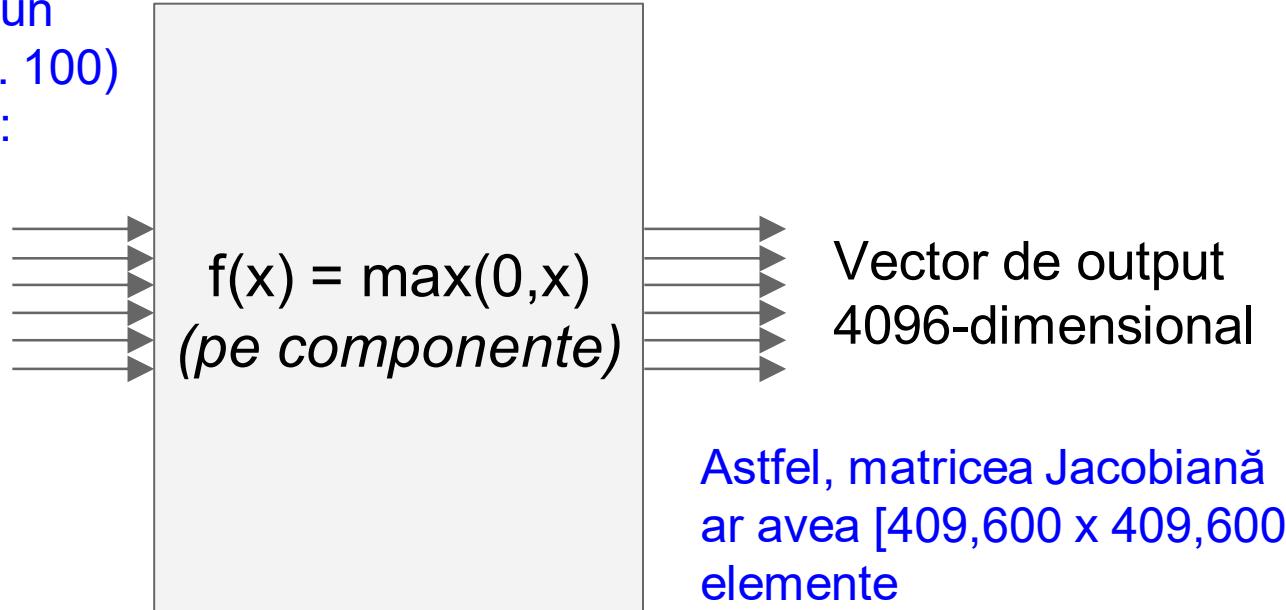
Vector de output  
4096-dimensional

Q: care este  
mărimea matricii  
Jacobiene?  
[4096 x 4096]

# Operații vectoriale

În practică procesăm un întreg mini-batch (e.g. 100) de exemple la un pas:

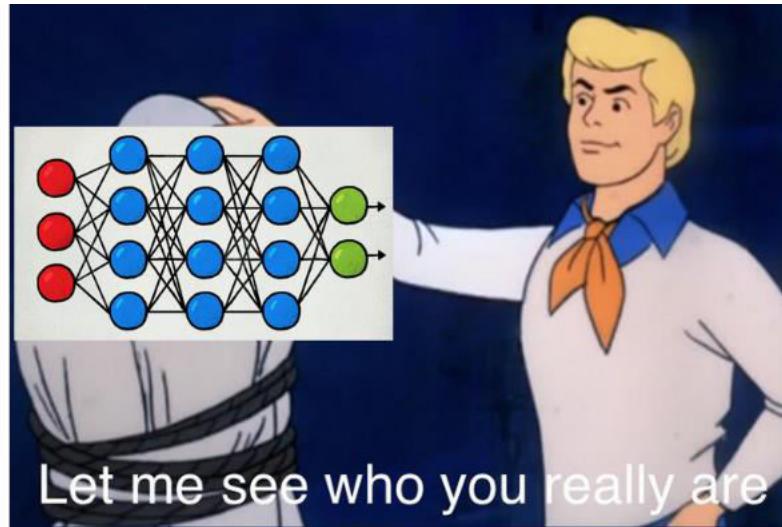
Vector de input  
4096-dimensional



Astfel, matricea Jacobiană ar avea [409,600 x 409,600] elemente

# Până acum...

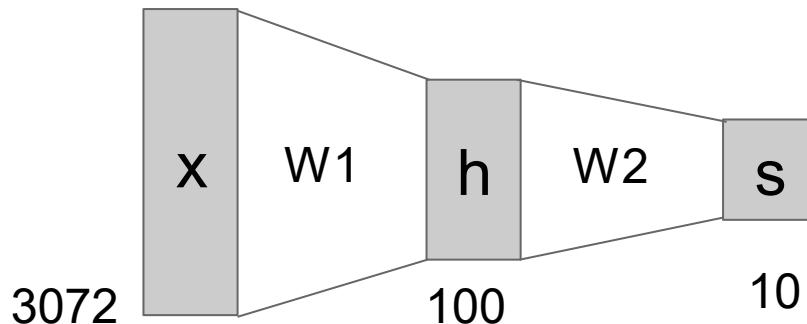
- Rețelele neuronale vor fi foarte mari: nici o speranță să scriem formula de mână pentru toți parameterii (folosim gradientul analitic)
- **Backpropagare** = aplicarea recursivă a regulii de înlăntuire (chain rule) de-a lungul unui graf computațional pentru calcularea gradientilor parametrilor / intrărilor
- Implementările mențin o structură de graf în care nodurile implementează funcțiile **forward()** / **backward()**
- **forward**: calculează rezultatul unei operații și salvează în memorie intrările / rezultatele intermediare necesare la calcularea gradientului
- **backward**: aplicarea regulii de înlăntuire pentru calcularea gradientului funcției de pierdere în raport cu intrările



# Rețele neuronale: din punct de vedere matematic

(Înainte) Funcție liniară de scoring:  $f = Wx$

(Acum) Rețea neuronală cu 2 nivele:  $f = W_2 \max(0, W_1 x)$



# Rețele neuronale: fără paralela cu neurologia

(Înainte) Funcție liniară de scoring:  $f = Wx$

(Acum) Rețea neuronală cu 2 nivele:  $f = W_2 \max(0, W_1 x)$

sau cu 3 nivele:

$$f = W_3 \max(0, W_2 \max(0, W_1 x))$$

## Antrenarea unei rețele cu două niveluri necesită ~11 linii de cod (Python)

```
X = np.array([[0,0,1],[0,1,1],[1,0,1],[1,1,1]])
```

```
Y = np.array([[0,1,1,0]]).T
```

```
W0 = 2 * np.random.random((3,4)) - 1
```

```
W1 = 2 * np.random.random((4,1)) - 1
```

```
for i in range(5000):
```

```
# forward pass
```

```
I1 = 1 / (1 + np.exp(-np.matmul(X, W0)))
```

```
I2 = 1 / (1 + np.exp(-np.matmul(I1, W1)))
```

```
# backward pass
```

```
delta_I2 = (Y - I2) * (I2 * (1 - I2))
```

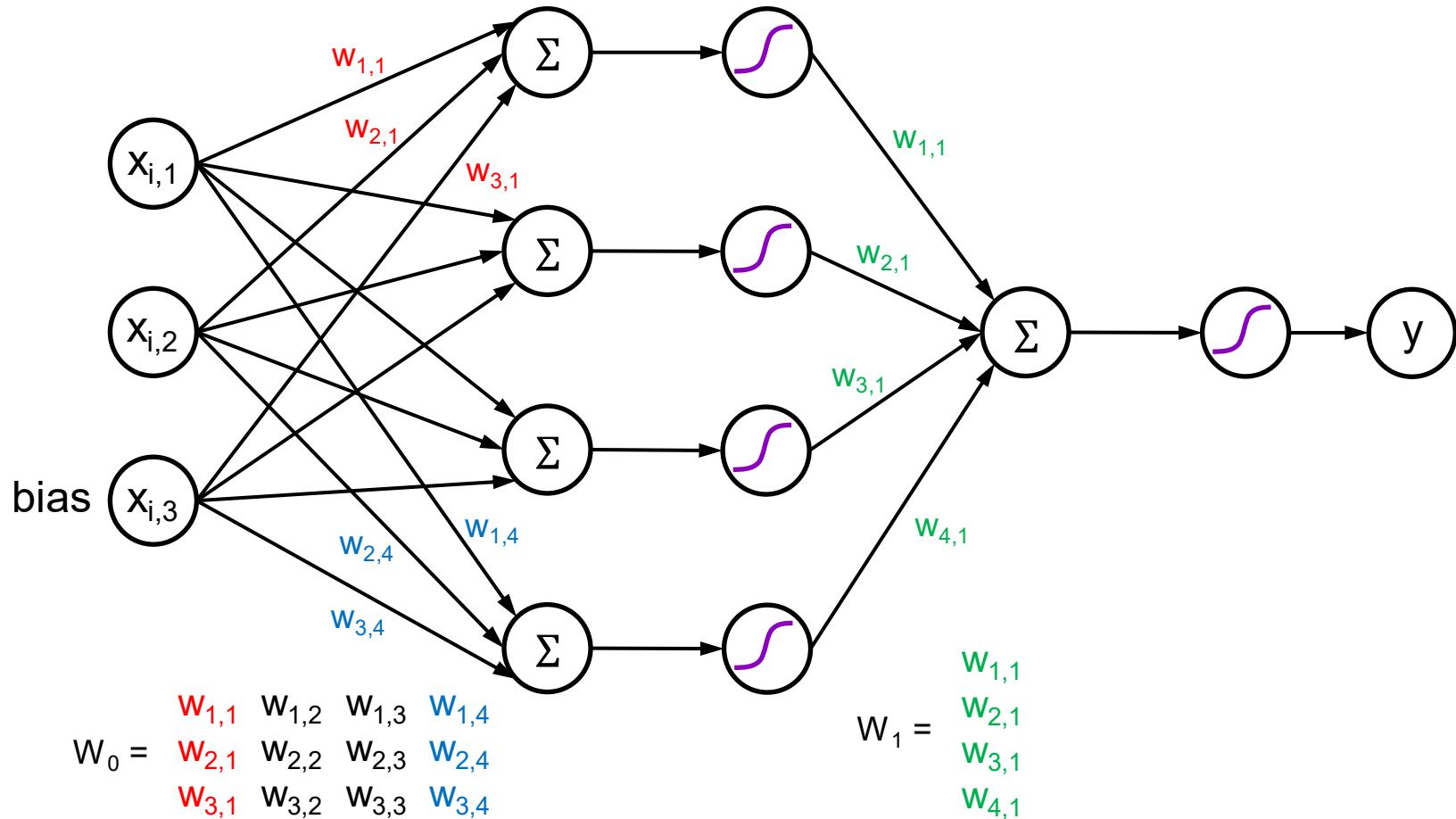
```
delta_I1 = np.matmul(delta_I2, W1.T) * (I1 * (1 - I1))
```

```
# gradient descent
```

```
W1 = W1 + np.matmul(I1.T, delta_I2)
```

```
W0 = W0 + np.matmul(X.T, delta_I1)
```

## Arhitectura rețelei cu două niveluri implementată anterior



# Rețele neuronale. Concepte despre modele de învățare deep.

Prof. Dr. Radu Ionescu  
[raducu.ionescu@gmail.com](mailto:raducu.ionescu@gmail.com)

Facultatea de Matematică și Informatică  
Universitatea din București

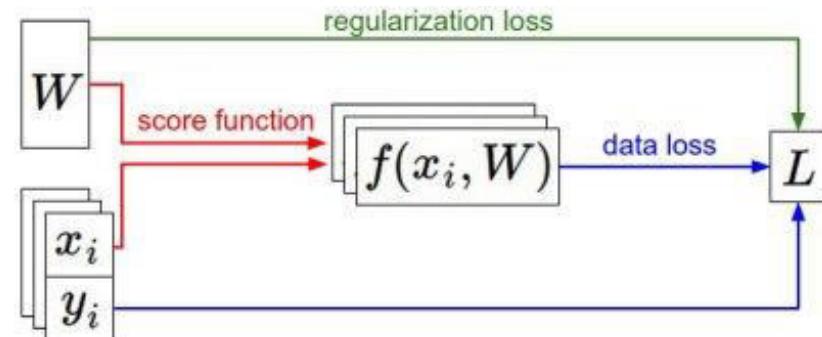
# Din cursul trecut:

- O mulțime de perechi  $(x, y)$
- O funcție de atribuire a scorului:  $s = f(x; W) = Wx$
- O funcție de pierdere:

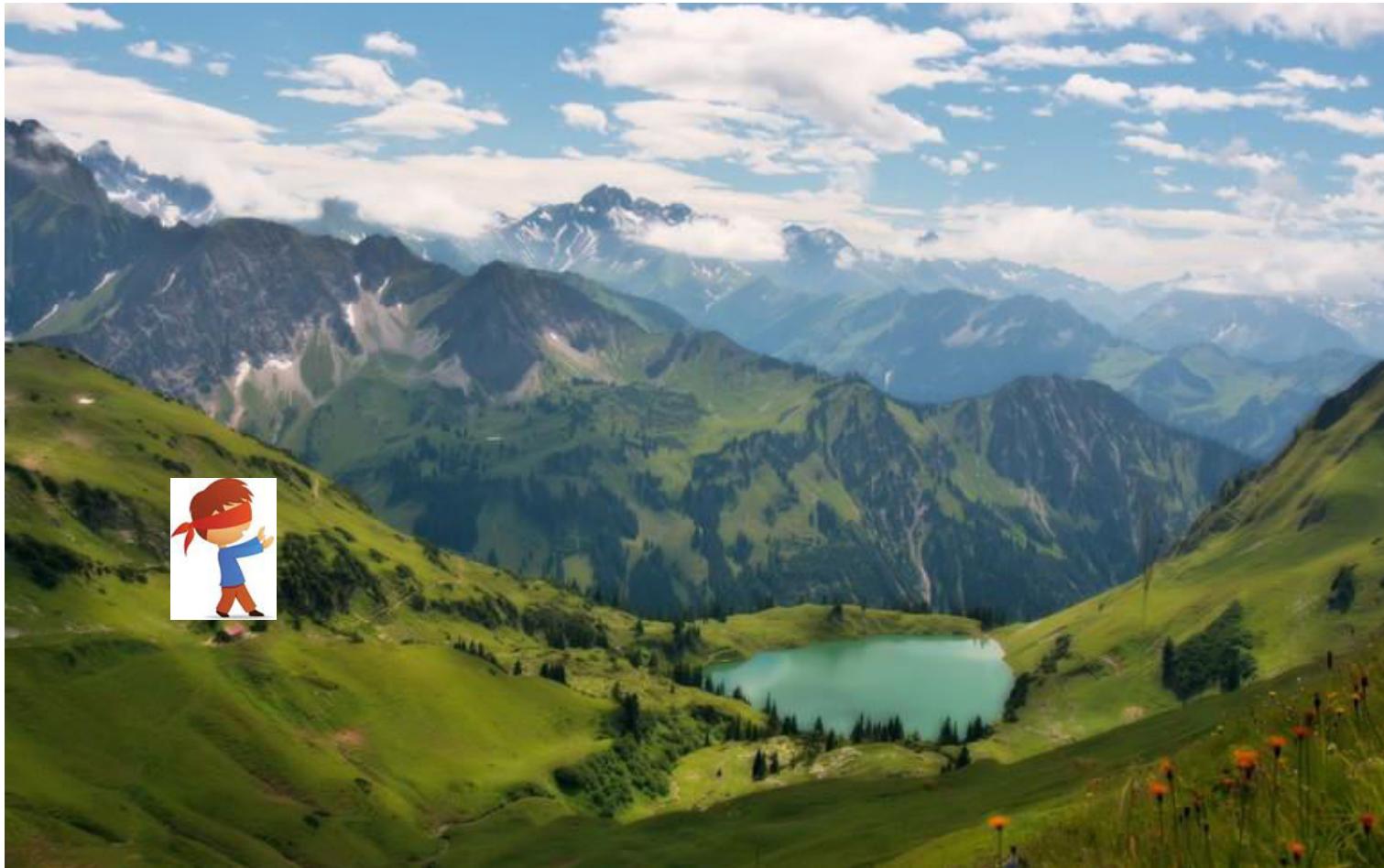
$$L_i = -\log\left(\frac{e^{sy_i}}{\sum_j e^{sj}}\right) \quad \text{Softmax}$$

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1) \quad \text{SVM}$$

$$L = \frac{1}{N} \sum_{i=1}^N L_i + R(W) \quad \text{Cu regularizare}$$



# Algoritm: Coborârea pe gradient



# În concluzie:

- Gradientul numeric: aproximativ, încet, ușor de scris
- Gradientul analitic: exact, rapid, înclinat spre greșeli

=>

În practică: Folosim întotdeauna gradientul analitic, dar verificăm implementarea cu gradientul numeric. Acest proces se numește **verificarea gradientului (gradient checking)**

# Algorimtul coborârii pe gradient (Python)

```
def GD(W0, X, goal, learningRate):
    perfGoalNotMet = true
    W = W0

    while perfGoalNotMet:
        gradient = eval_gradient(X, W)
        W_old = W
        W = W - learningRate * gradient
        perfGoalNotMet = sum(abs(W - W_old)) > goal
```

# De la extragere “manuală” către învățare

vector ce descrie statistici despre imagine, e.g. bag-of-words



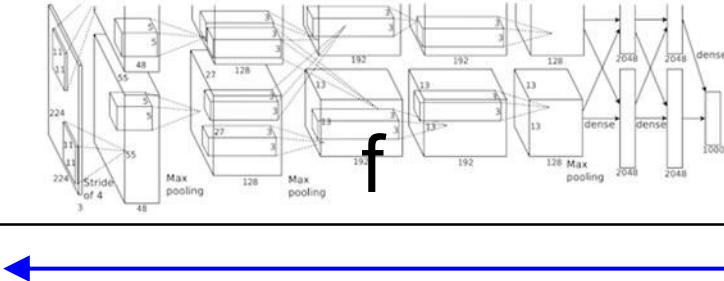
Extragere de  
trăsături



$f$

$N$  numere ce indică scorurile pentru fiecare clasă

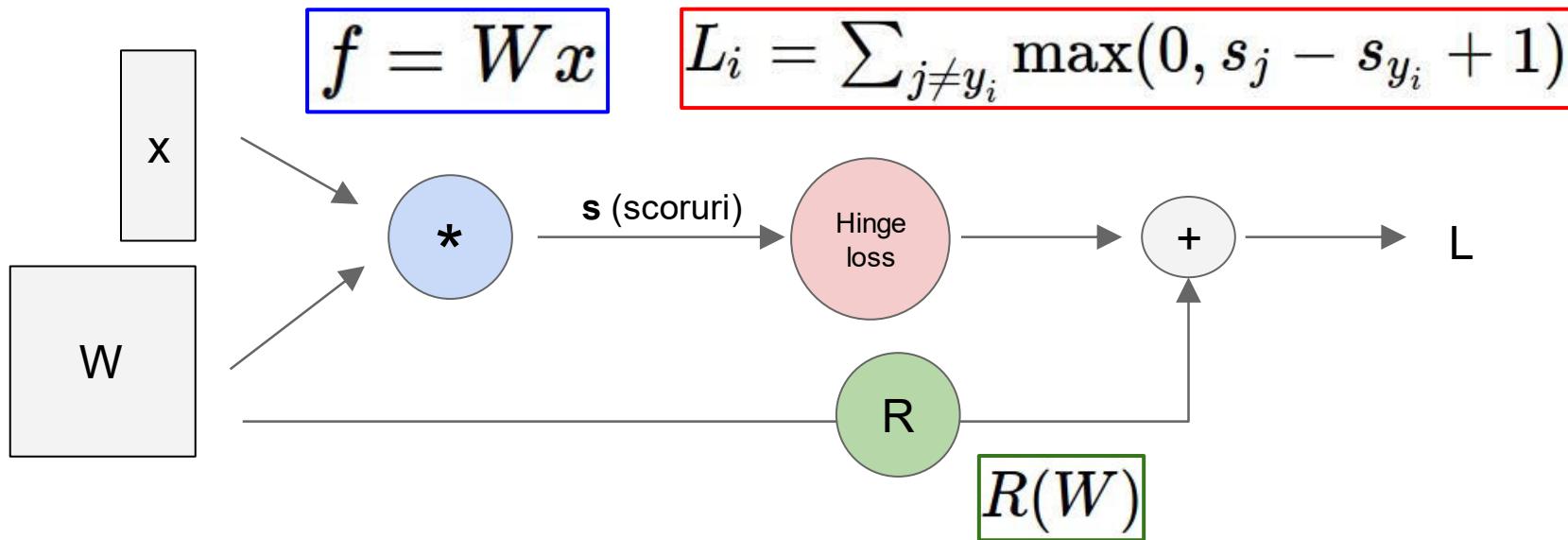
învățare



învățare “end-to-end”

$N$  numere ce indică scorurile pentru fiecare clasă

# Privim algoritmul ca un graf computational



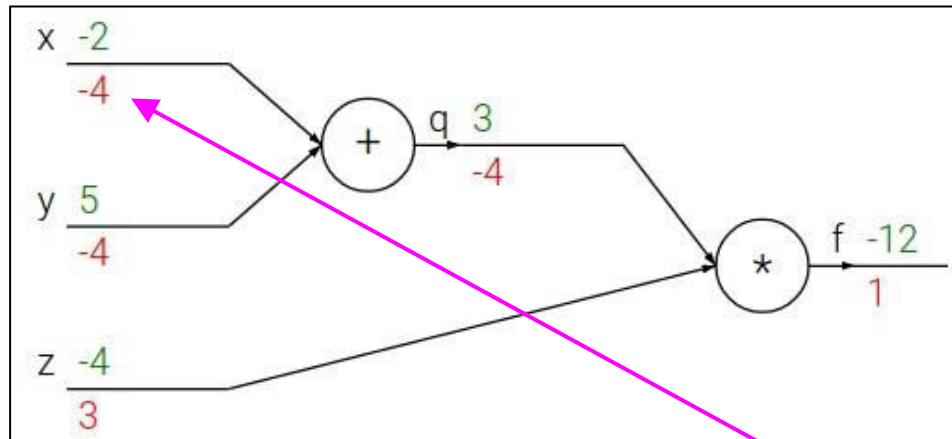
$$f(x, y, z) = (x + y)z$$

e.g.  $x = -2, y = 5, z = -4$

$$q = x + y \quad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \quad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

vrem:  $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

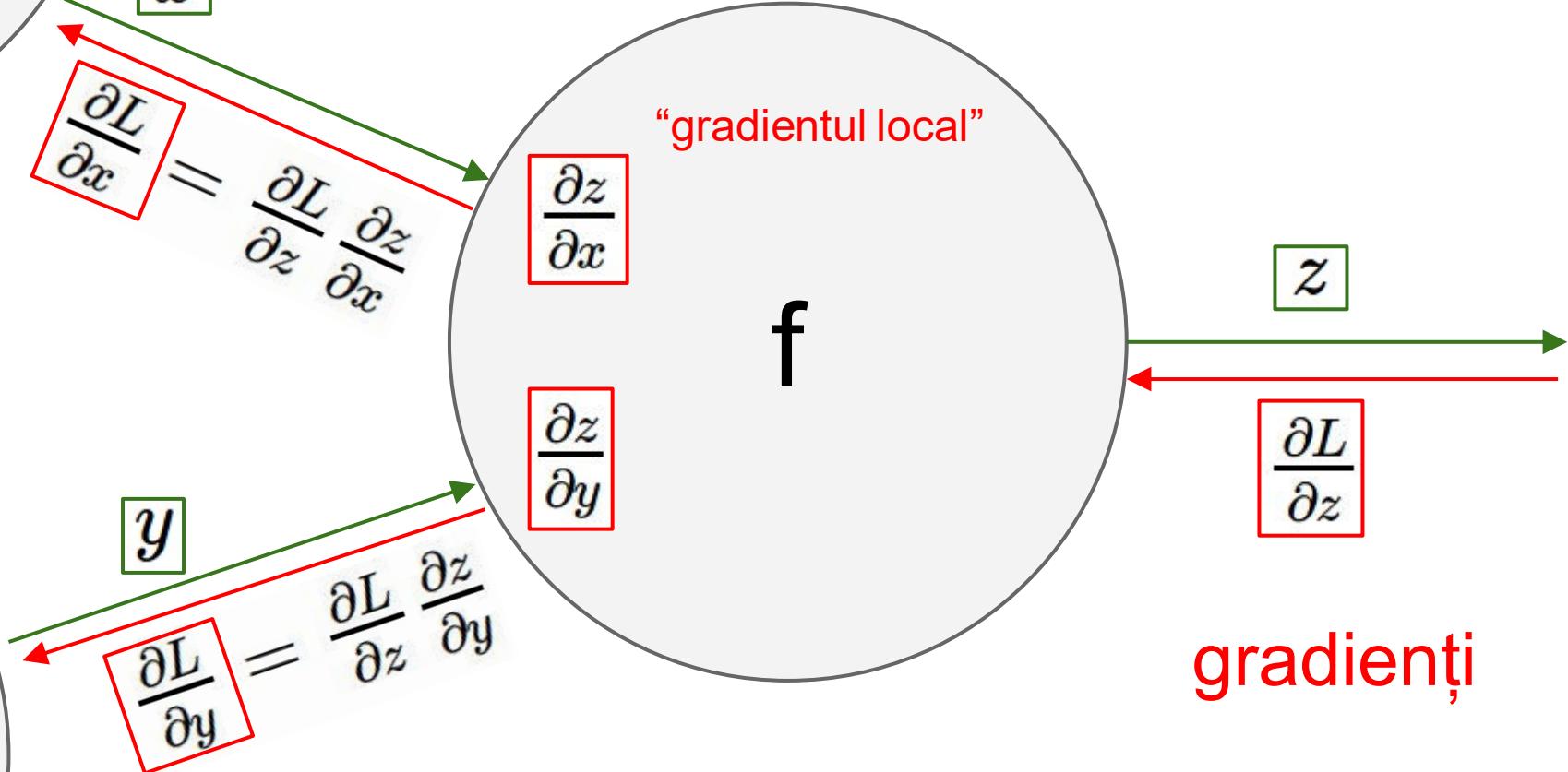


Regula de înlățuire:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q} \frac{\partial q}{\partial x}$$

$$\frac{\partial f}{\partial x}$$

# Propagarea gradientului prin regula de înlăntuire activări



# Din cursul trecut...

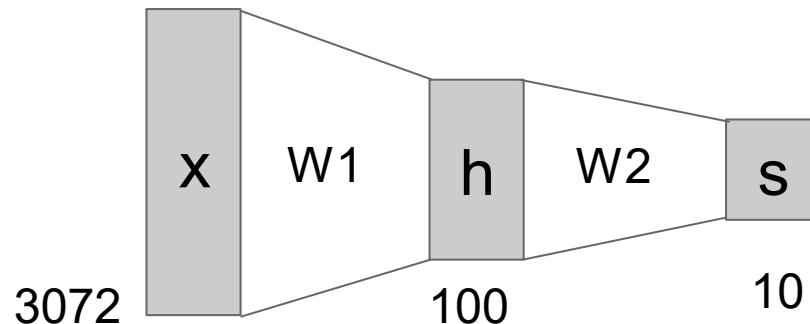
- Rețelele neuronale pot fi foarte mari: nici o speranță să scriem formula de mână pentru toți parameterii (folosim gradientul analitic)
- **Backpropagare** = aplicarea recursivă a regulii de înlățuire (chain rule) de-a lungul unui graf computațional pentru calcularea gradientilor parametrilor / intrărilor
- Implementările mențin o structură de graf în care nodurile implementează funcțiile **forward()** / **backward()**
- **forward**: calculează rezultatul unei operații și salvează în memorie intrările / rezultatele intermediare necesare la calcularea gradientului
- **backward**: aplicarea regulii de înlățuire pentru calcularea gradientului funcției de pierdere în raport cu intrările



# Rețele neuronale: fără paralela cu neurologia

(Înainte) Funcție liniară de scoring:  $f = Wx$

(Acum) Rețea neuronală cu 2 nivele:  $f = W_2 \max(0, W_1 x)$



# Rețele neuronale: fără paralela cu neurologia

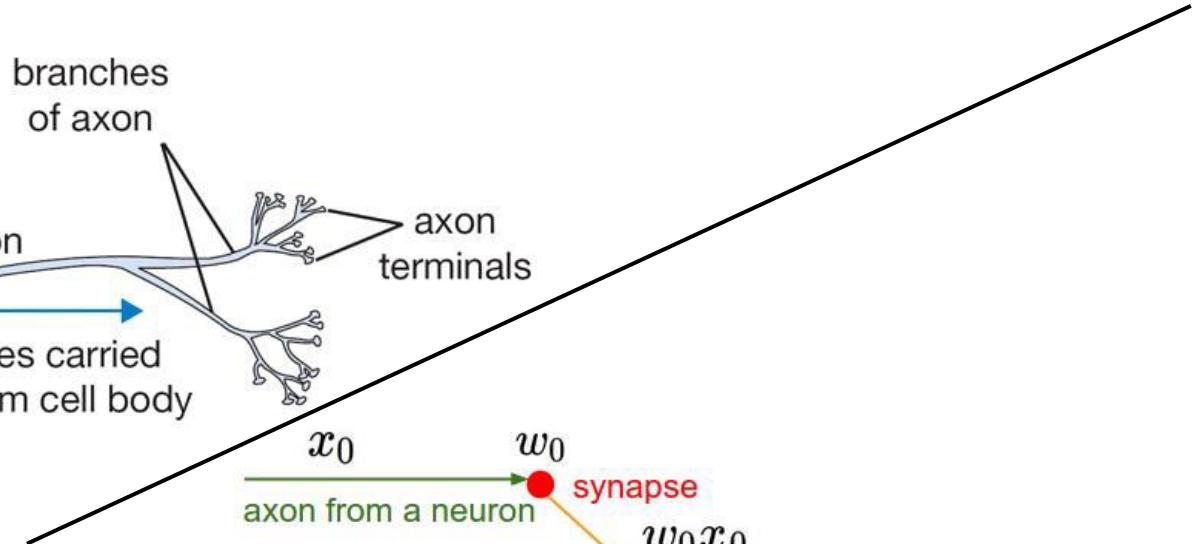
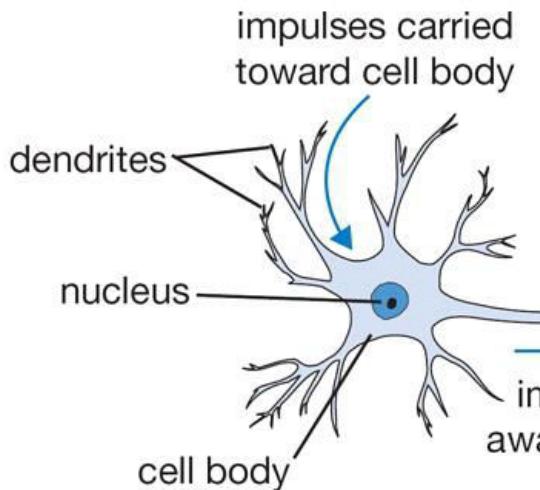
(Înainte) Funcție liniară de scoring:  $f = Wx$

(Acum) Rețea neuronală cu 2 nivele:  $f = W_2 \max(0, W_1 x)$

sau cu 3 nivele:

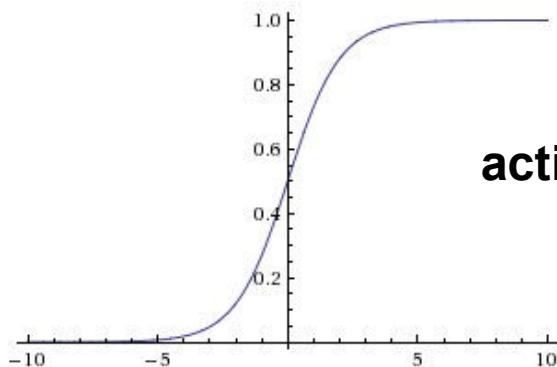
$$f = W_3 \max(0, W_2 \max(0, W_1 x))$$





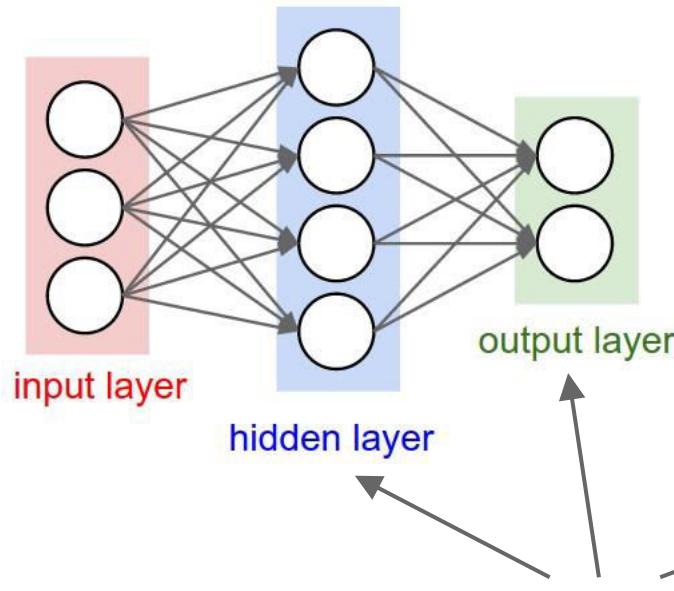
**funcția de activare sigmoidă**

$$\frac{1}{1 + e^{-x}}$$

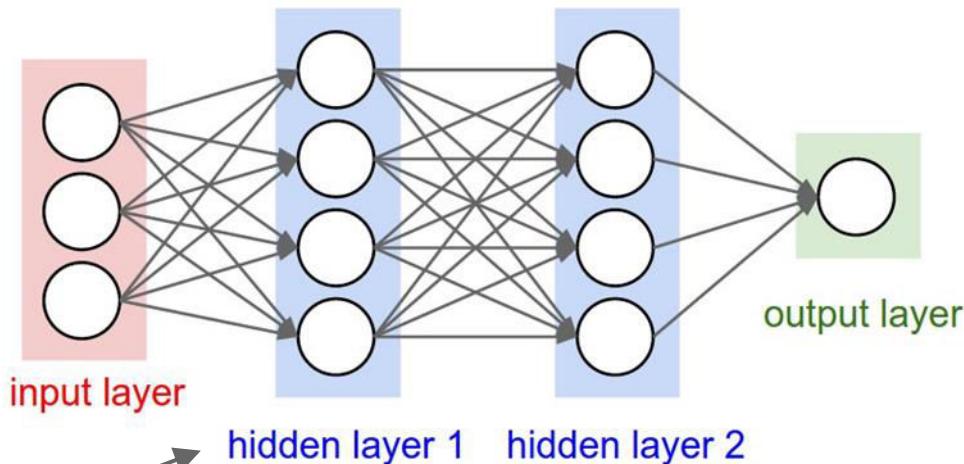


# Arhitecturi de rețele neuronale

Rețea neuronală cu două straturi  
(cu un singur strat ascuns)



Rețea neuronală cu trei straturi  
(cu două straturi ascunse)



**Straturi “fully-connected”**

## Antrenarea unei rețele cu două niveluri necesită ~11 linii de cod (Python)

```
X = np.array([[0,0,1],[0,1,1],[1,0,1],[1,1,1]])
```

```
Y = np.array([[0,1,1,0]]).T
```

```
W0 = 2 * np.random.random((3,4)) - 1
```

```
W1 = 2 * np.random.random((4,1)) - 1
```

```
for i in range(5000):
```

```
# forward pass
```

```
I1 = 1 / (1 + np.exp(-np.matmul(X, W0)))
```

```
I2 = 1 / (1 + np.exp(-np.matmul(I1, W1)))
```

```
# backward pass
```

```
delta_I2 = (Y - I2) * (I2 * (1 - I2))
```

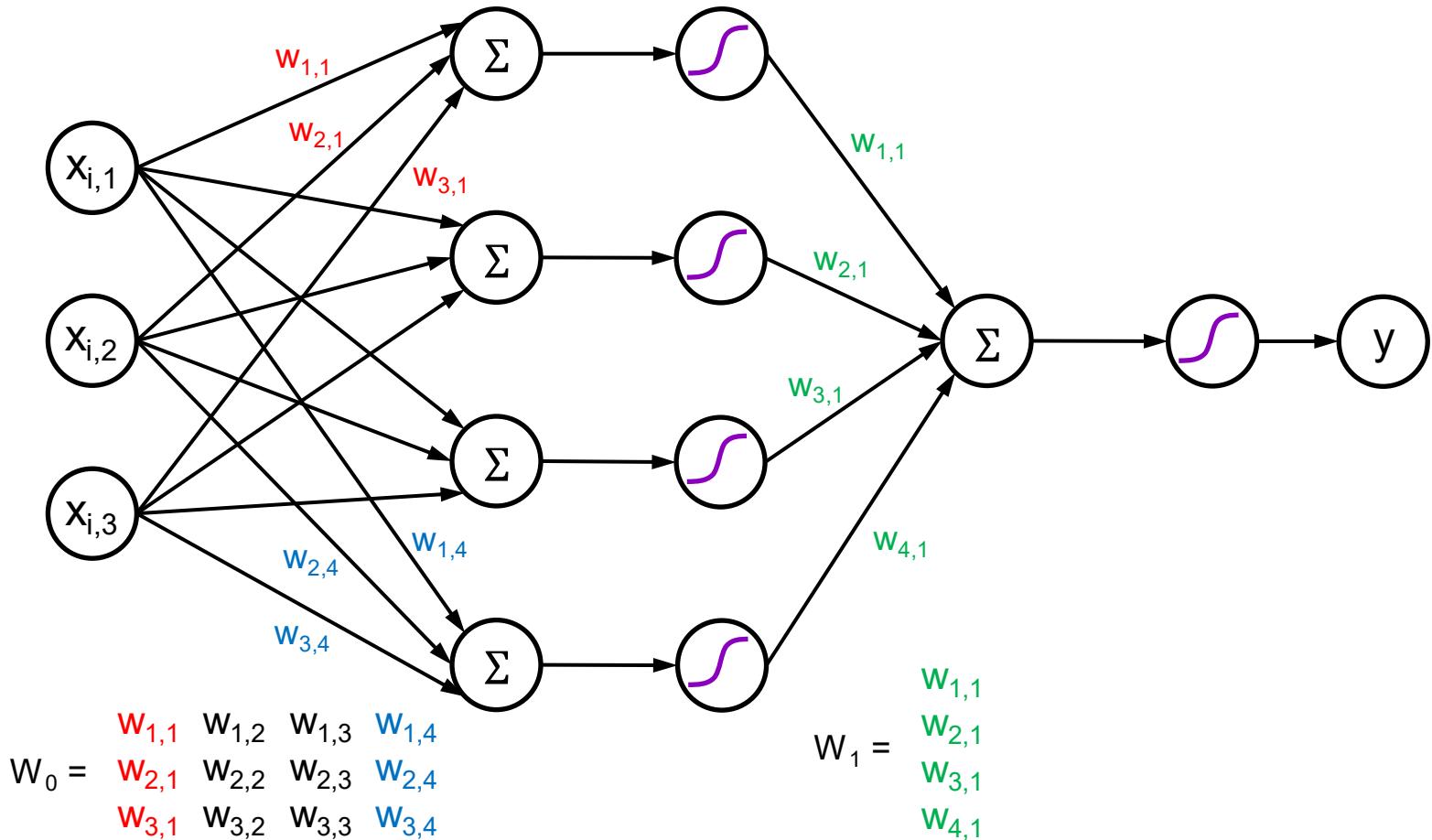
```
delta_I1 = np.matmul(delta_I2, W1.T) * (I1 * (1 - I1))
```

```
# gradient descent
```

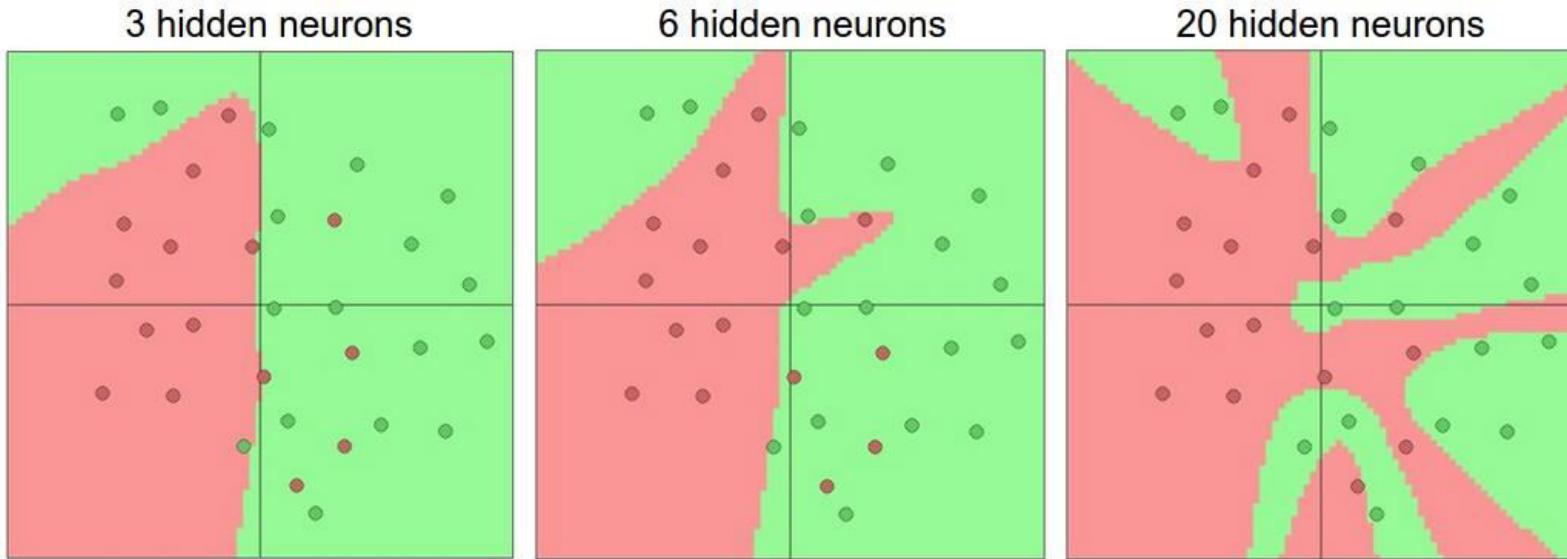
```
W1 = W1 + np.matmul(I1.T, delta_I2)
```

```
W0 = W0 + np.matmul(X.T, delta_I1)
```

## Arhitectura rețelei cu două niveluri implementată anterior



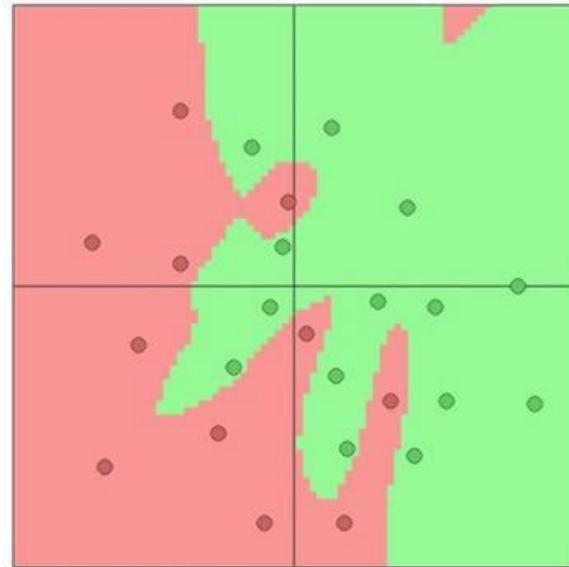
# Alegerea numărului de straturi și a numărului de neuroni



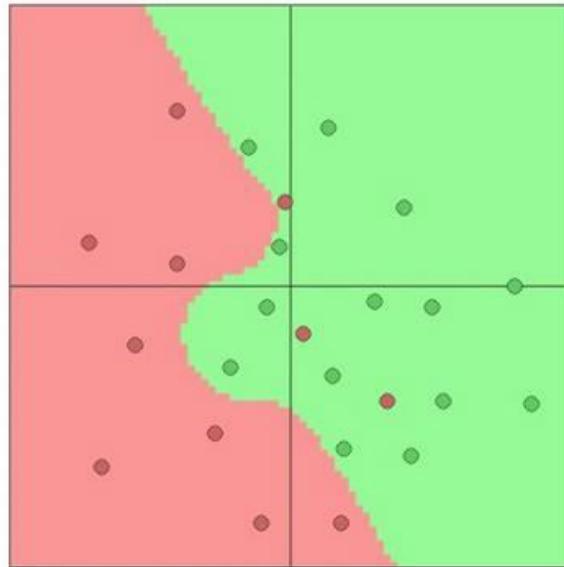
mai mulți neuroni = mai multă capacitate

# Alegerea parametrului de regularizare

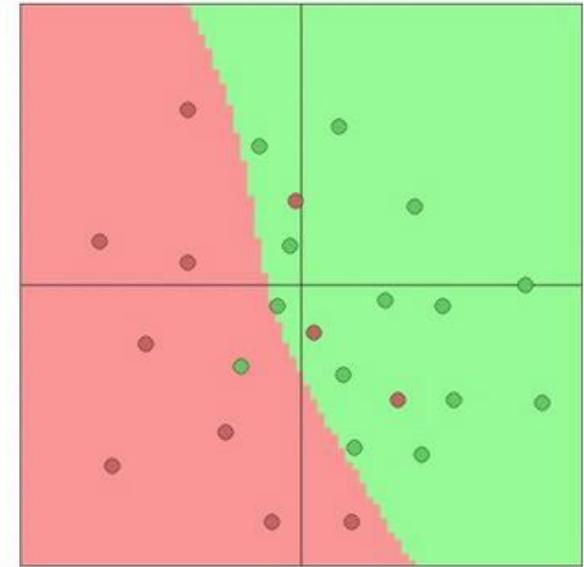
$\lambda = 0.001$



$\lambda = 0.01$



$\lambda = 0.1$



Sfat practic: În general este mai bine să folosim regularizare mai puternică  
în loc să reducem capacitatea modelului

# Alegerea arhitecturii potrivite

- Aranjăm neuronii în straturi fully-connected
- La nivel de implementare, abstractizarea unui strat ne permite să utilizăm cod vectorial (e.g. înmulțirea matricilor)
- Performanța crește cu cât arhitectura rețelei este mai adâncă (deep), i.e. are mai multe straturi (**dar trebuie să folosim o regularizare mai puternică**)

# Antrenarea rețelelor neuronale

# Scurt istoric

Mașina **Mark I Perceptron** a fost prima implementare a algoritmului perceptronului.

Mașina era conectată la o camera cu  $20 \times 20$  fotocelule de sulfat de cadmiu pentru a produce o imagine cu 400 de pixeli.

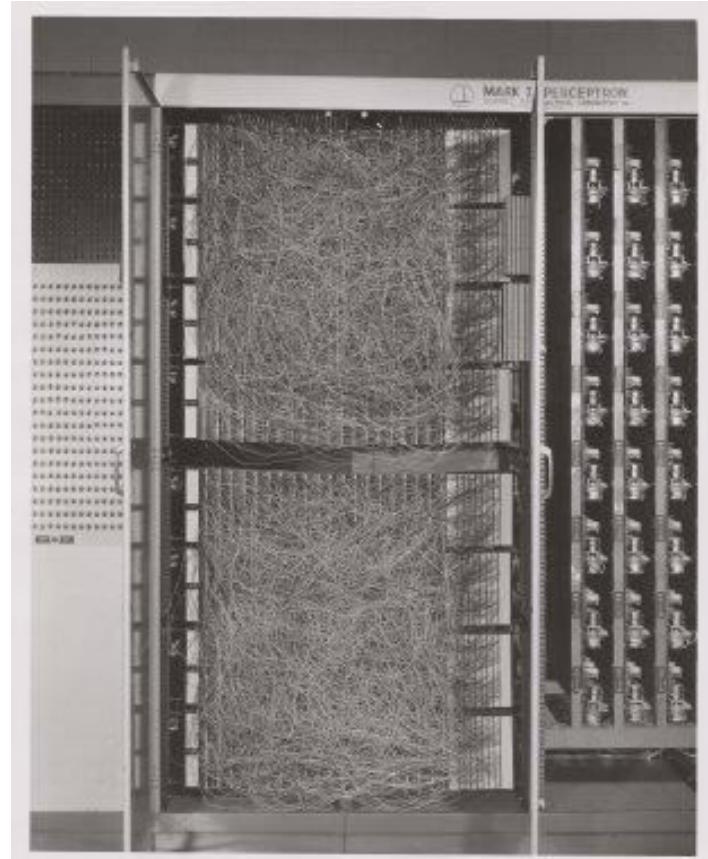
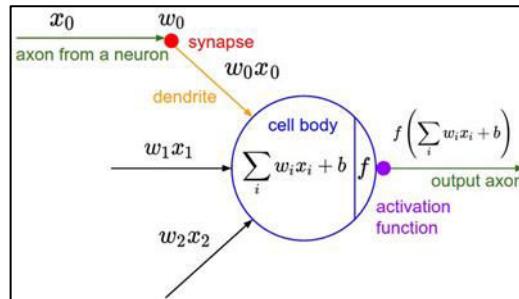
Folosită pentru a recunoaște litere din alfabet.

$$f(x) = \begin{cases} 1 & \text{if } w \cdot x + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

Regula de actualizare

$$w_i(t+1) = w_i(t) + \alpha(d_j - y_j(t))x_{j,i}$$

*Frank Rosenblatt, ~1957: Perceptron*



# Scurt istoric

Mașina **ADALINE** folosea rezistoare cu memorie capabile să execute operații logice și să stocheze informații.

Funcția de pierdere (suma pătratelor erorilor)

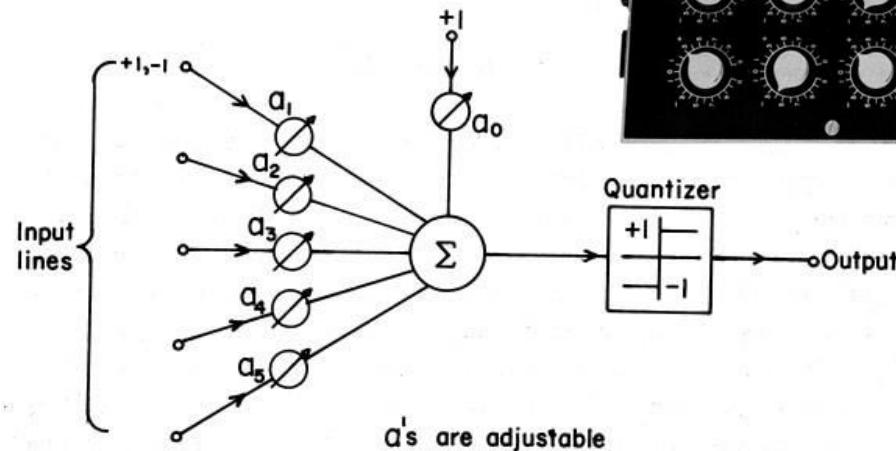
$$\frac{1}{2} \sum_i (d^i - y^i)^2, \text{ unde } y^i = (x^i)^T w + b$$

Regula de actualizare

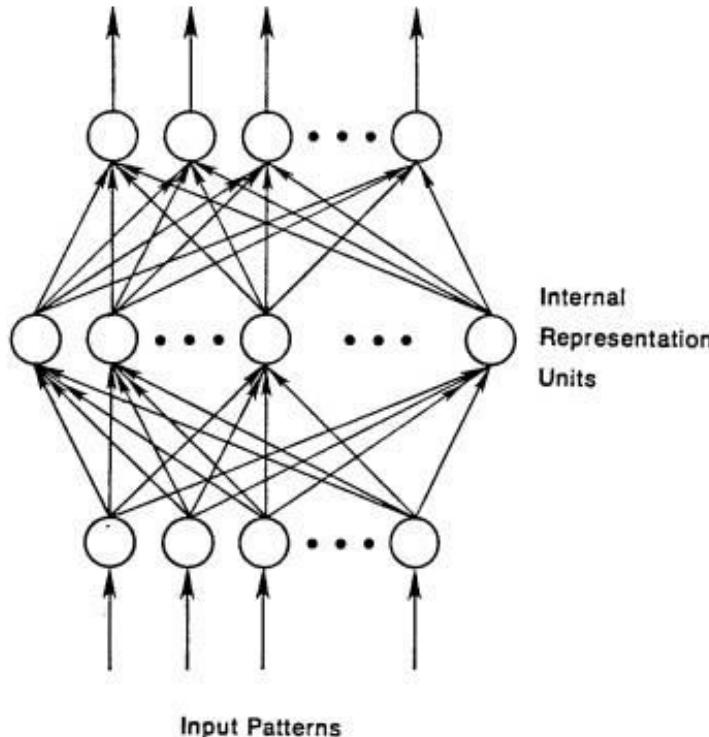
$$w^{k+1} = w^k + \mu \sum_{i=1}^m (d^i - y^i) x^i$$

$$b^{k+1} = b^k + \mu \sum_{i=1}^m (d^i - y^i)$$

*Widrow and Hoff, ~1960: Adaline*



# Scurt istoric



Formule matematice  
mai ușor de înțeles

To be more specific, then, let

$$E_p = \frac{1}{2} \sum_j (t_{pj} - o_{pj})^2 \quad (2)$$

be our measure of the error on input/output pattern  $p$  and let  $E = \sum E_p$  be our overall measure of the error. We wish to show that the delta rule implements a gradient descent in  $E$  when the units are linear. We will proceed by simply showing that

$$-\frac{\partial E_p}{\partial w_{ji}} = \delta_{pj} i_{pi},$$

which is proportional to  $\Delta_p w_{ji}$  as prescribed by the delta rule. When there are no hidden units it is straightforward to compute the relevant derivative. For this purpose we use the chain rule to write the derivative as the product of two parts: the derivative of the error with respect to the output of the unit times the derivative of the output with respect to the weight.

$$\frac{\partial E_p}{\partial w_{ji}} = \frac{\partial E_p}{\partial o_{pj}} \frac{\partial o_{pj}}{\partial w_{ji}}. \quad (3)$$

The first part tells how the error changes with the output of the  $j$ th unit and the second part tells how much changing  $w_{ji}$  changes that output. Now, the derivatives are easy to compute. First, from Equation 2

$$\frac{\partial E_p}{\partial o_{pj}} = -(t_{pj} - o_{pj}) = -\delta_{pj}. \quad (4)$$

Not surprisingly, the contribution of unit  $o_j$  to the error is simply proportional to  $\delta_{pj}$ . Moreover, since we have linear units,

$$o_{pj} = \sum_i w_{ji} i_{pi}, \quad (5)$$

from which we conclude that

$$\frac{\partial o_{pj}}{\partial w_{ji}} = i_{pi}$$

Thus, substituting back into Equation 3, we see that

$$-\frac{\partial E_p}{\partial w_{ji}} = \delta_{pj} i_{pi} \quad (6)$$

Hinton et al. 1986: Algoritmul de propagare înapoi a erorii (backpropagation)

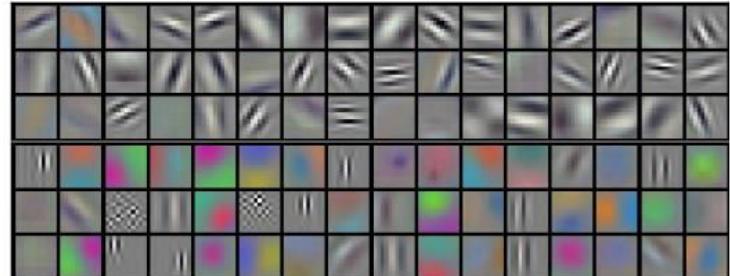
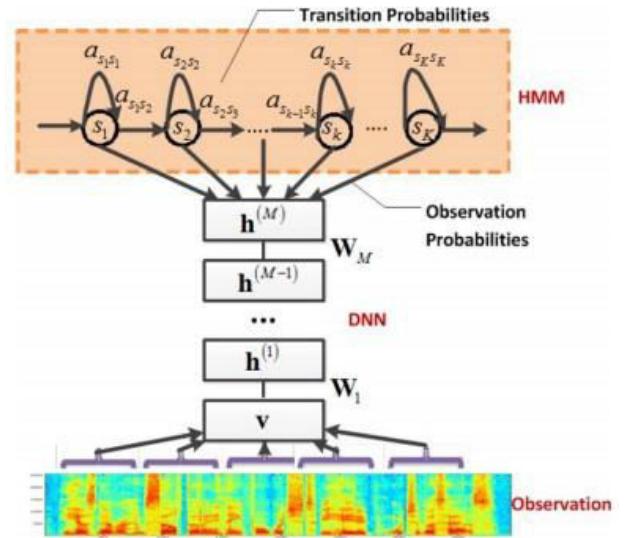
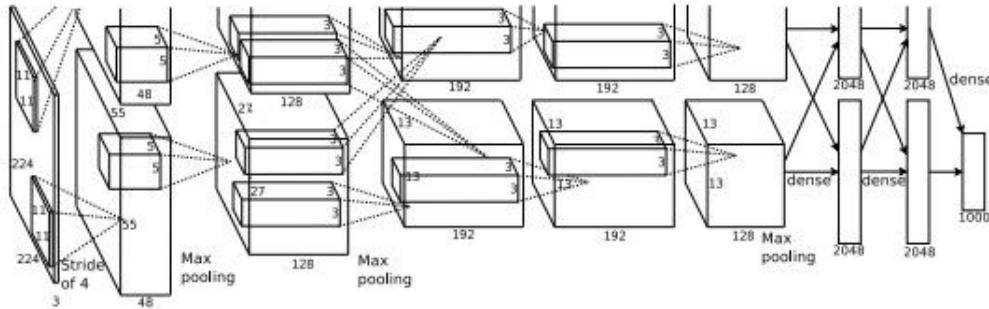
# Primele rezultate semnificative bazate pe învățare cu modele deep

***Context-Dependent Pre-trained Deep Neural Networks for Large Vocabulary Speech Recognition***

George Dahl, Dong Yu, Li Deng, Alex Acero, 2010

***ImageNet classification with deep convolutional neural networks***

Alex Krizhevsky, Ilya Sutskever, Geoffrey E Hinton, 2012



# Antrenarea rețelelor neuronale: privire de ansamblu

## 1. Ce trebuie să stabilim la început (o dată)

*Functiile de activare, procesarea, initializarea ponderilor, regularizarea, verificarea gradientului*

## 2. Ce ține de dinamica antrenării

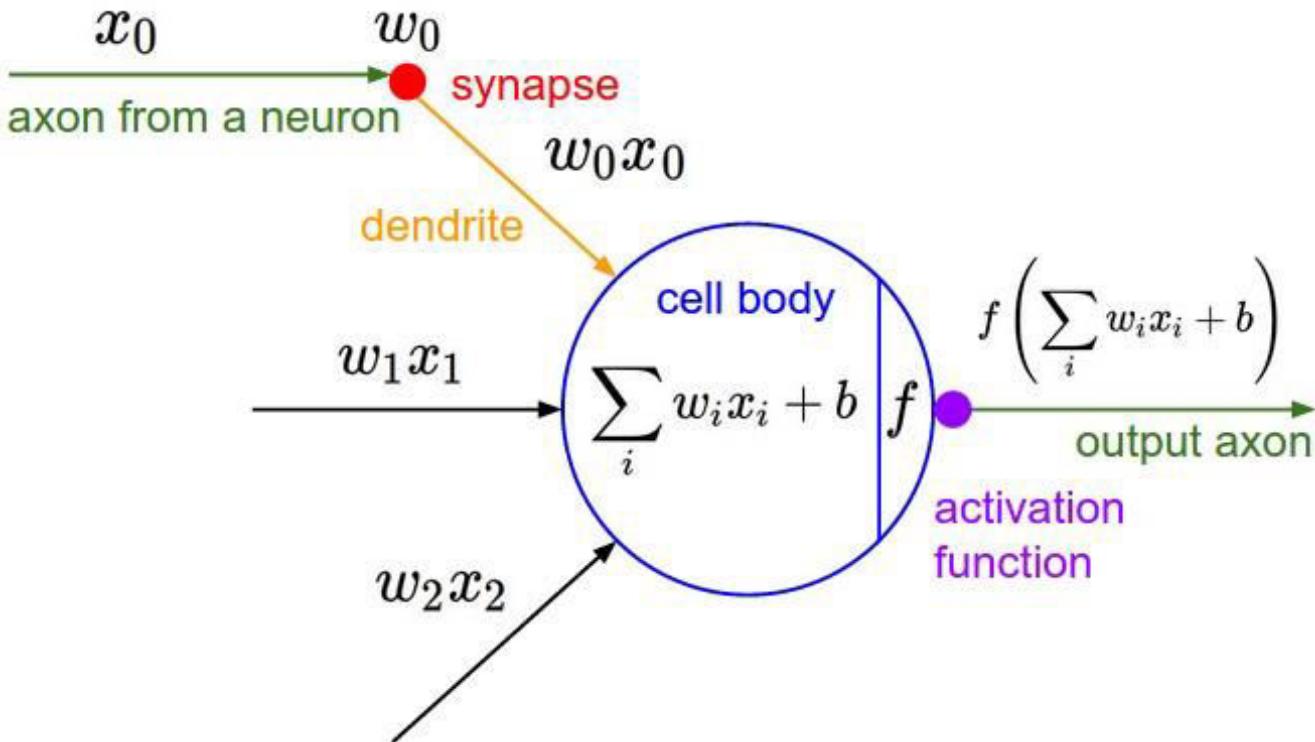
*Asistarea procesului de învățare, actualizarea parametrilor, optimizarea hiperparametrilor*

## 3. Evaluare

*Ansamble de modele*

# Functii de activare

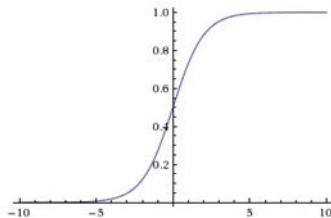
# Funcții de activare



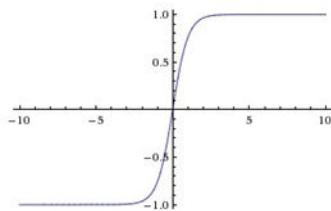
# Funcții de activare

**sigmoidă**

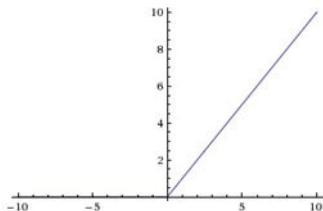
$$\sigma(x) = 1/(1 + e^{-x})$$



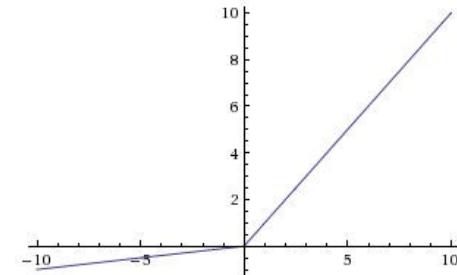
**tanh**     $\tanh(x)$



**ReLU**     $\max(0, x)$

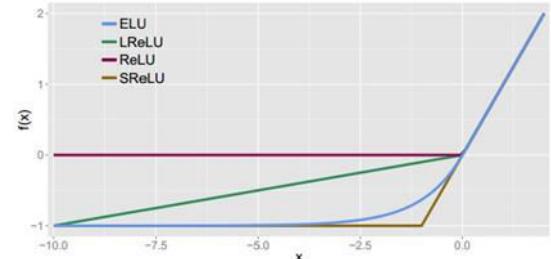


**Leaky ReLU**  
 $\max(0.1x, x)$

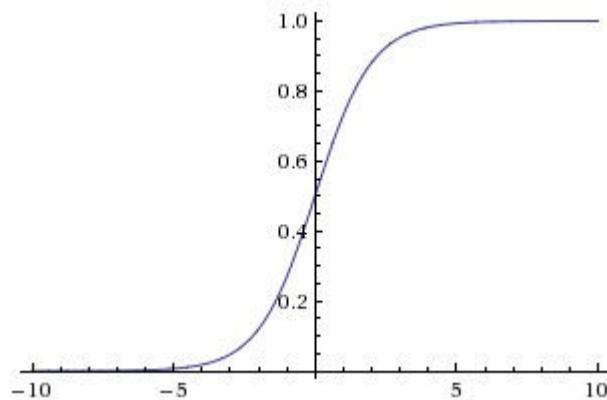


**Maxout**     $\max(w_1^T x + b_1, w_2^T x + b_2)$

**ELU**     $f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha (\exp(x) - 1) & \text{if } x \leq 0 \end{cases}$



# Functii de activare



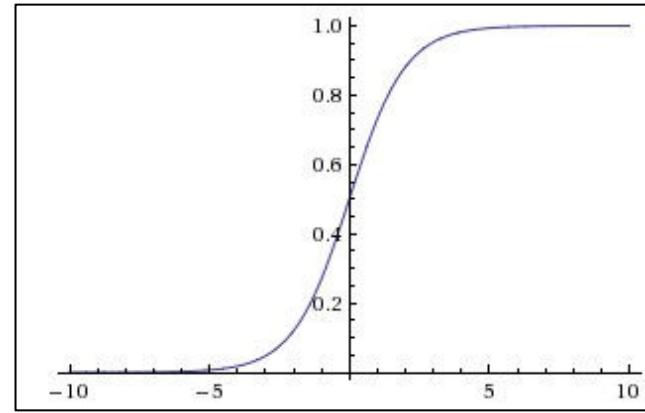
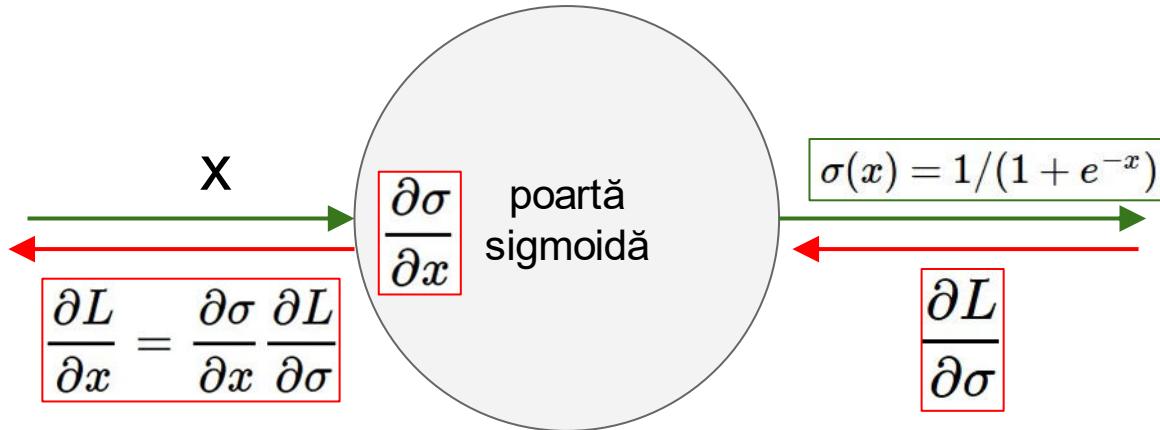
**sigmoidă**

$$\sigma(x) = 1/(1 + e^{-x})$$

- Aduce numerele în intervalul [0, 1]
- Populară din punct de vedere istoric deoarece are interpretarea biologică a saturării ratei de activare a unui neuron

3 probleme:

1. Neuronii saturați “omoară” gradientii

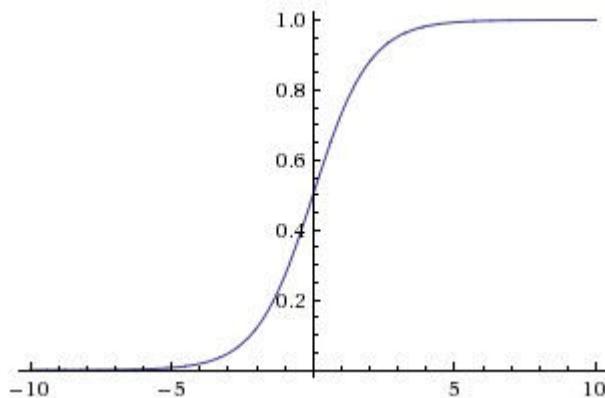


Ce se întâmplă când  $x = -10$ ?

Ce se întâmplă când  $x = 0$ ?

Ce se întâmplă când  $x = 10$ ?

# Functii de activare



**sigmoidă**

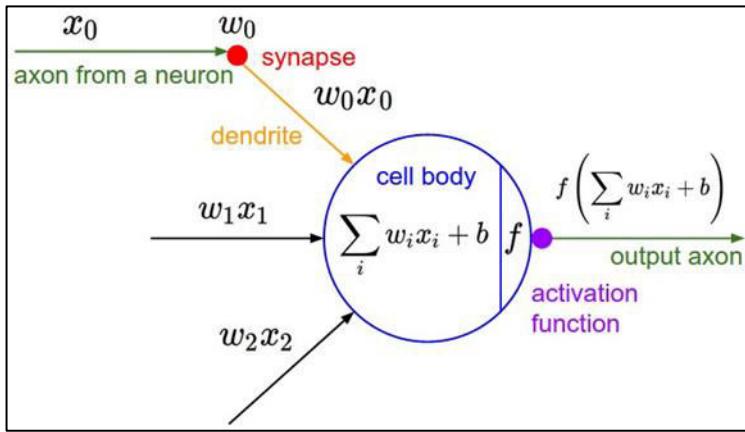
$$\sigma(x) = 1/(1 + e^{-x})$$

- Aduce numerele în intervalul [0, 1]
- Populară din punct de vedere istoric deoarece are interpretarea biologică a saturării ratei de activare a unui neuron

3 probleme:

1. Neuronii saturați “omoară” gradienții
2. Output-ul funcției sigmoide nu este centrat în zero

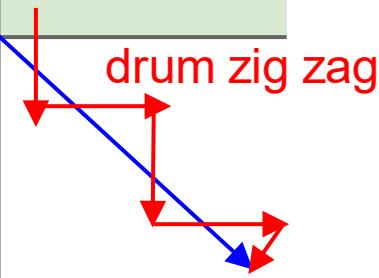
Să considerăm ce se întâmplă dacă intrarea  $x$  este întotdeauna pozitivă:



$$f \left( \sum_i w_i x_i + b \right)$$

Direcții posibile de actualizare a gradientilor

Direcții posibile de actualizare a gradientilor

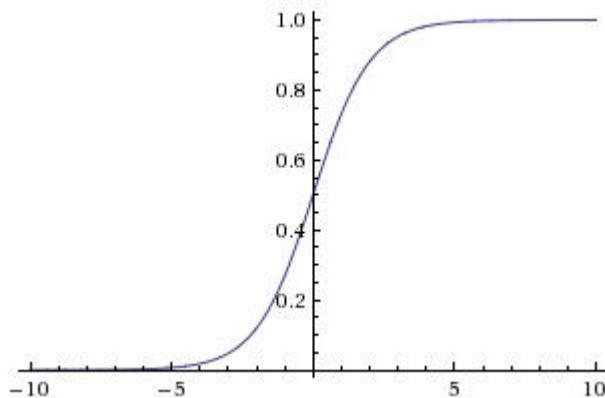


Ce putem spune despre gradientii în raport cu  $w$ ?

Fie toți pozitivi, fie toți negativi :(

(din acest motiv ne dorim date de medie zero)

# Functii de activare



**sigmoidă**

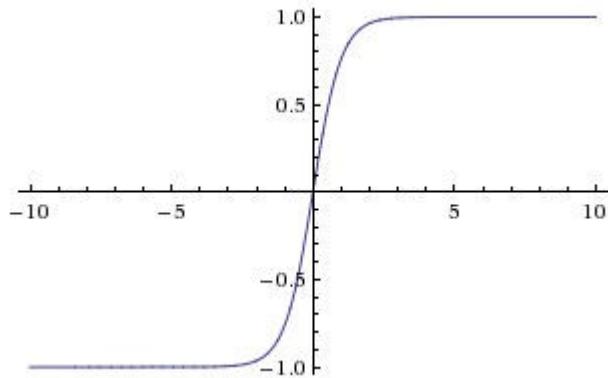
$$\sigma(x) = 1/(1 + e^{-x})$$

- Aduce numerele în intervalul [0, 1]
- Populară din punct de vedere istoric deoarece are interpretarea biologică a saturării ratei de activare a unui neuron

3 probleme:

1. Neuronii saturați “omoară” gradienții
2. Output-ul funcției sigmoide nu este centrat în zero
3.  $\exp()$  are un cost computațional ridicat

# Funcții de activare



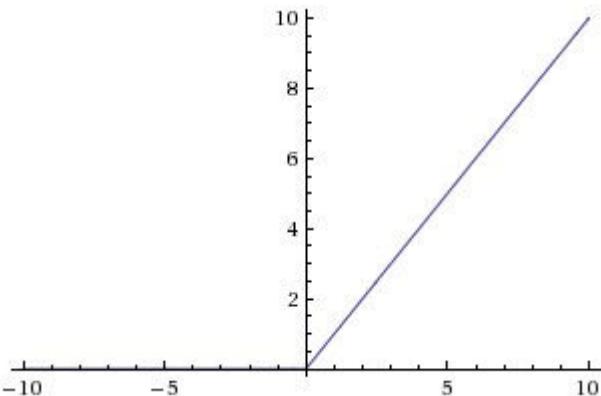
**tanh(x)**

$$\frac{e^x - e^{-x}}{e^x + e^{-x}}$$

- Aduce numerele în intervalul [-1, 1]
- De medie zero (bine)
- Încă omoară gradienții atunci când se saturează :(

[LeCun et al., 1991]

# Funcții de activare

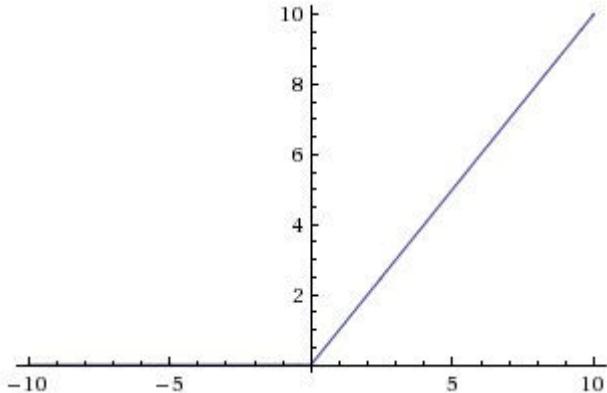
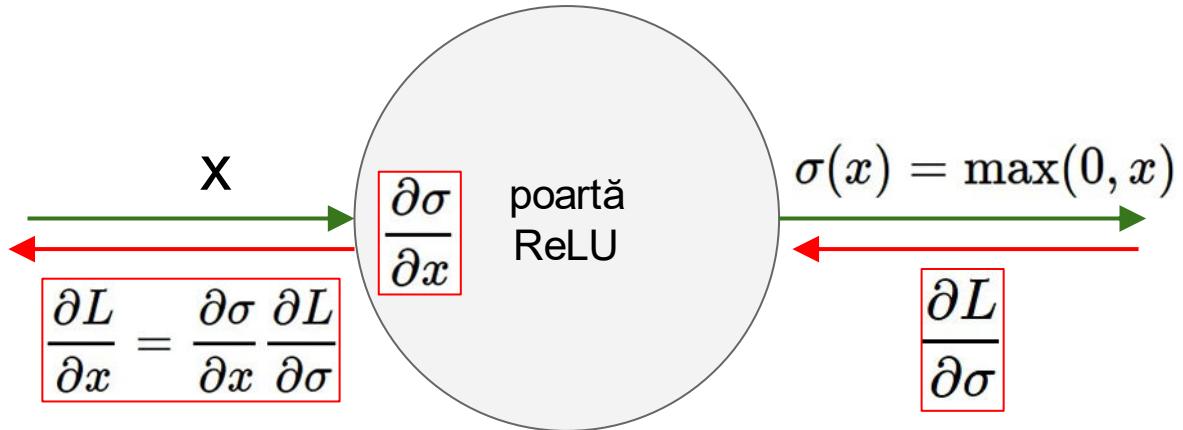


**ReLU**  
(Rectified Linear Unit)  
 $f(x) = \max(0, x)$

- Nu se saturează (în partea pozitivă)
- Foarte eficient computațional
- În practică, converge mult mai rapid decât sigmoida/tanh (e.g.  $6x$ )

- Output-ul nu are media zero
- O situație neplăcută (atunci când  $x < 0$ , gradientul este 0)

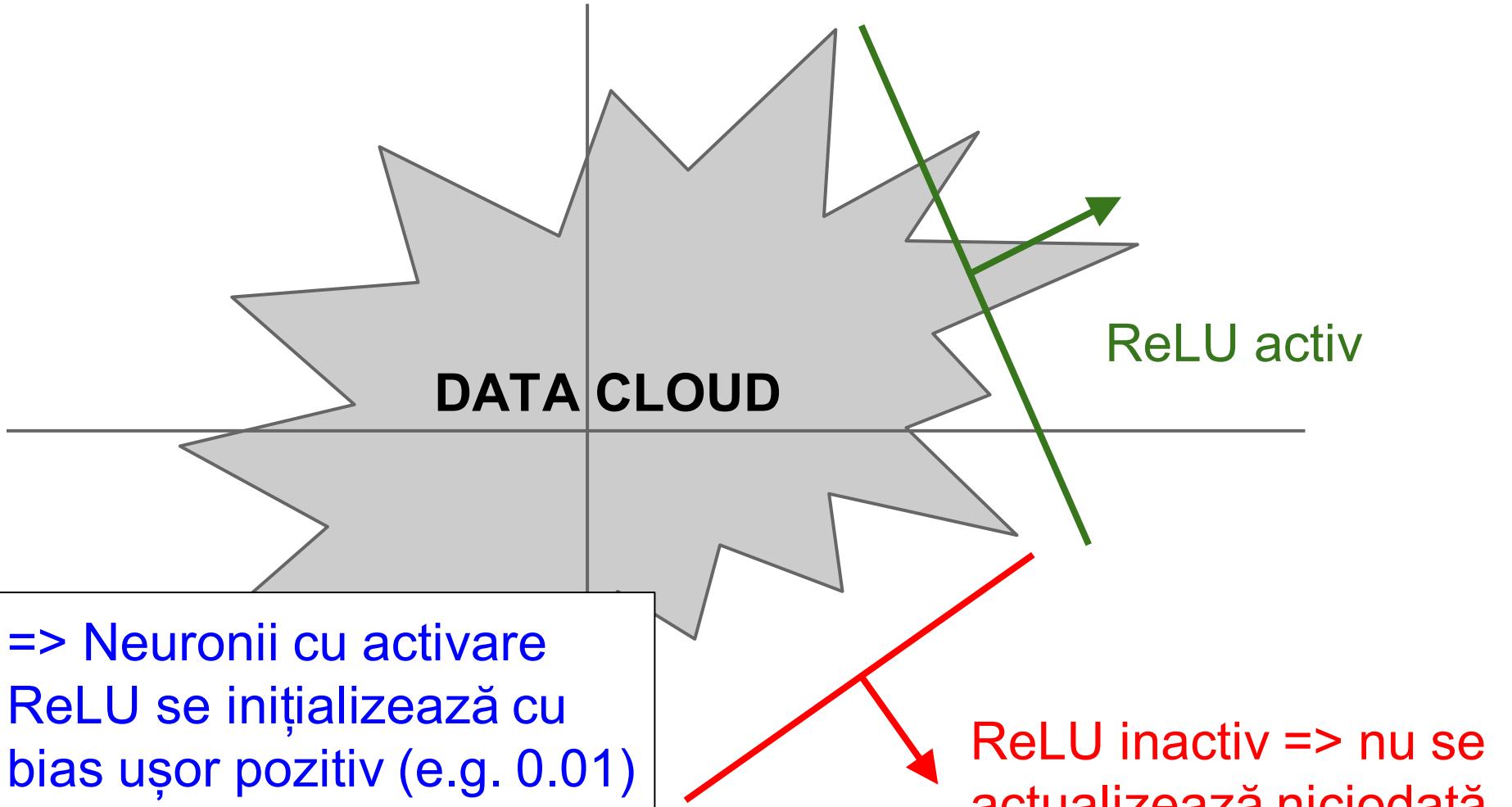
[Krizhevsky et al., 2012]



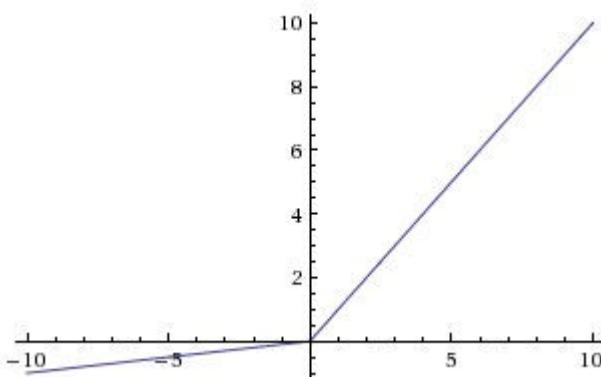
Ce se întâmplă când  $x = -10$ ?

Ce se întâmplă când  $x = 0$ ?

Ce se întâmplă când  $x = 10$ ?



# Functii de activare



## Leaky ReLU

$$f(x) = \max(0.01x, x)$$

- Foarte eficient computațional
- În practică, converge mult mai rapid decât sigmoida/tanh (e.g. 6x)
- **Nu se saturează**

## Parametric Rectifier (PReLU)

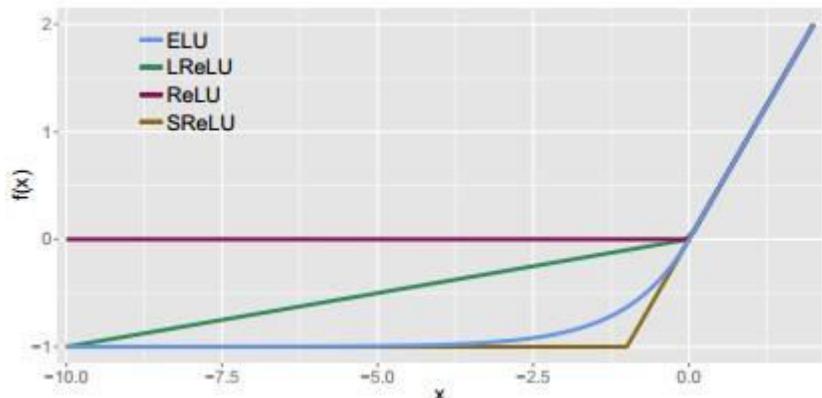
$$f(x) = \max(\alpha x, x)$$

[Mass et al., 2013]

[He et al., 2015]

# Functii de activare

## Exponential Linear Units (ELU)



- Toate beneficiile ReLU
- Nu se saturează
- Output aproape de medie zero
- Implică calculul  $\exp()$

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha (\exp(x) - 1) & \text{if } x \leq 0 \end{cases}$$

[Clevert et al., 2015]

# Neuronul cu funcție de activare Maxout

- Nu are forma generală a produsului scalar => non-liniaritate
- Generalizează ReLU și Leaky ReLU
- Liniar pe intervale! Nu se saturează!

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

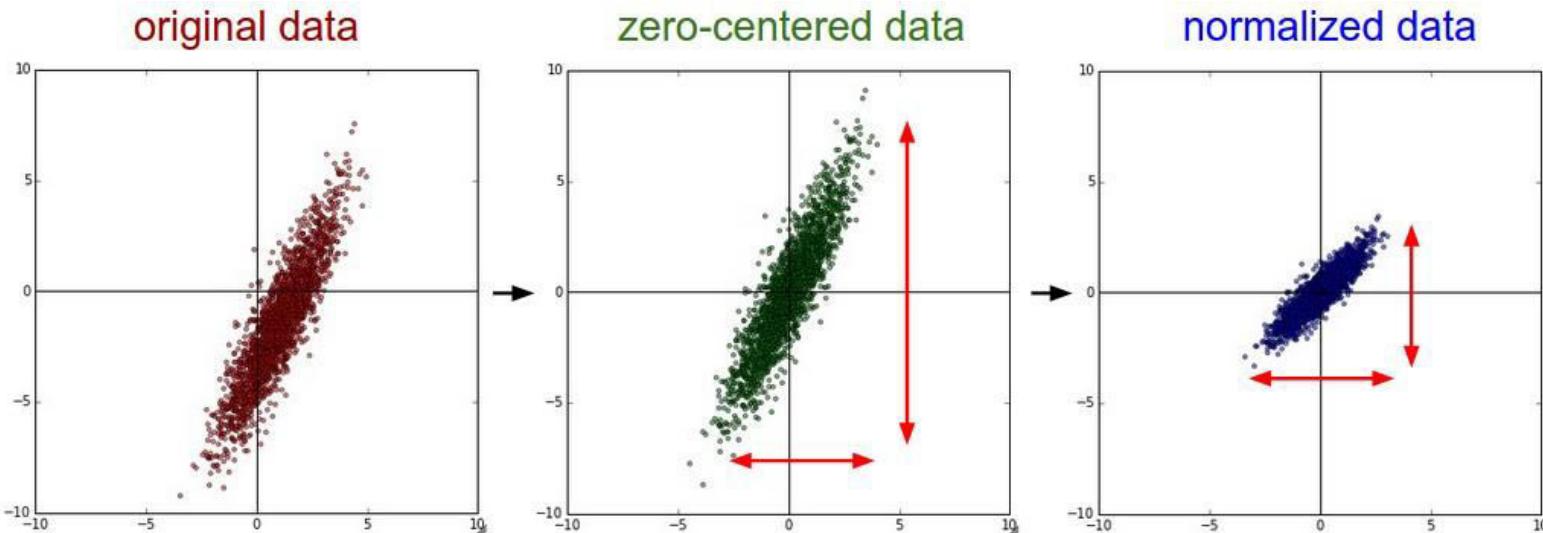
Problemă: se dublează numărul de parametri/neuroni :(

# Ce funcții de activare în practică?

- Utilizăm ReLU. Trebuie să avem grijă cu rata de învățare
- Putem încerca Leaky ReLU / Maxout / ELU
- Putem încerca tanh (fără așteptări prea mari)
- Evităm pe cât posibil sigmoida

# Preprocesarea datelor

# Preprocesarea datelor



$X = X - \text{np.mean}(X, \text{axis}=0, \text{keepdims=True})$

$X = X / \text{np.std}(X, \text{axis}=0, \text{keepdims=True}))$

( $X$  este o matrice  $[NxD]$ , câte un exemplu pe linie)

# Pentru imagini este suficient să centrăm datele

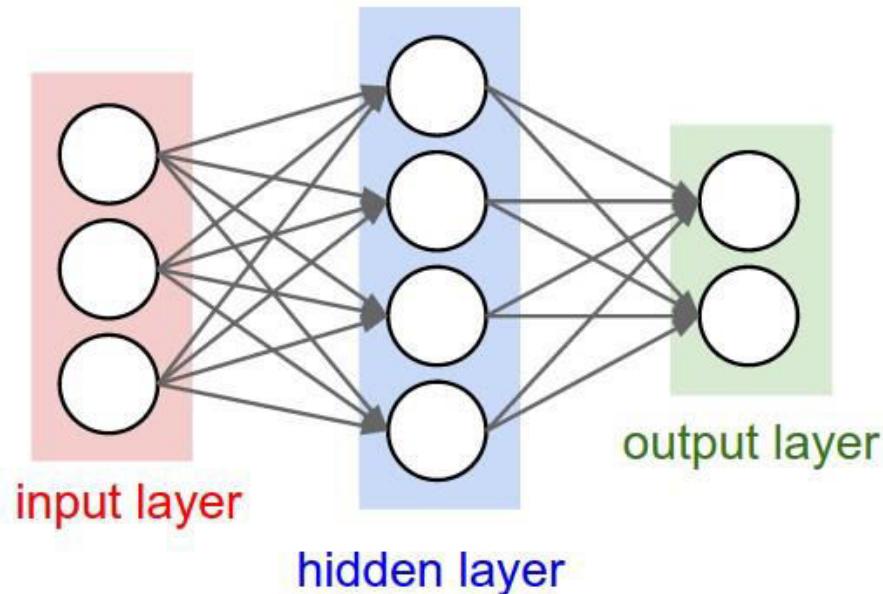
Considerând setul de date CIFAR-10 cu imagini de [32,32,3]

- Scădem imaginea medie (e.g. AlexNet)  
(imaginea medie = matrice [32,32,3])
- Scădem media pe fiecare canal (e.g. VGGNet)  
(media pe fiecare canal = 3 numere)

Nu este o practică comună să normalizăm imaginile

# Initializarea ponderilor

Ce se întâmplă dacă inițializăm  $W=0$ ?



O primă idee: Inițializăm cu numere aleatorii aproape de zero

$W = np.random.normal(0, 0.01, (N,D))$   
(distribuție normală de medie zero și dispersie 0.01)

Funcționează ~bine pentru rețele mici, dar poate conduce la distribuții neomogene ale funcțiilor de activare din straturile unei rețele.

Aproape toți neuronii se saturează complet, fie spre -1 fie spre 1. Gradienții vor fi zero.

# A doua abordare: Inițializare Xavier

Probleme cu alegerea ponderilor inițiale:

- Dacă sunt prea mici, semnalul care se propagă în rețea se diminuează cu fiecare nivel și devine prea mic pentru a fi util
- Dacă sunt prea mari, semnalul care se propagă în rețea crește cu fiecare nivel până când devine prea mare pentru a fi util

Inițializarea Xavier ne asigură că ponderile au magnitudinea potrivită, păstrând semnalul într-un interval rezonabil.

Ponderile inițiale provin dintr-o distribuție normală de medie 0 și o disperzie dată de numărul de perceptri de pe stratul anterior / posterior:

$$\text{Var}(W) = \frac{2}{n_{\text{in}} + n_{\text{out}}}$$

[Glorot and Bengio, 2010]

# Normalizarea Batch

Vrem activări normale de medie 0 și dispersie 1?  
Le transformăm a.î. să devină aşa.

Considerăm activările pe un anumit strat pentru un mini-batch. Pentru a transforma fiecare dimensiune aplicăm:

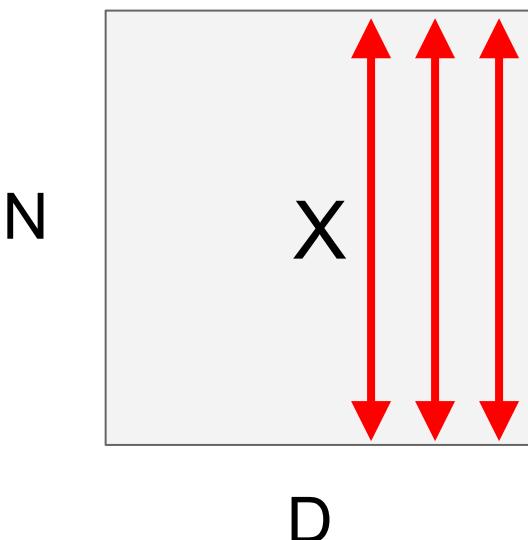
$$\hat{x}^{(k)} = \frac{x^{(k)} - \text{E}[x^{(k)}]}{\sqrt{\text{Var}[x^{(k)}]}}$$

[Ioffe and Szegedy, 2015]

# Normalizarea Batch

Vrem activări normale de medie 0 și dispersie 1?

Le transformăm a.î. să devină aşa.



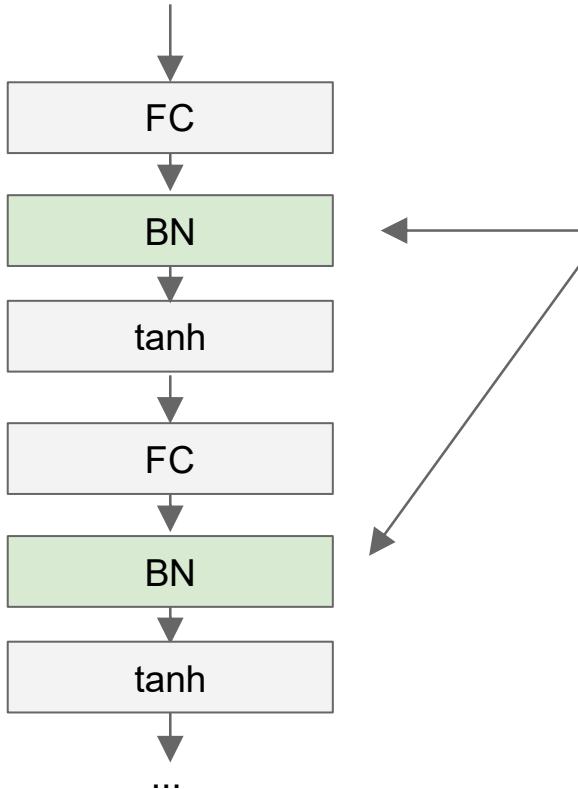
1. Calculăm media empirică și  
dispersia pentru fiecare dimensiune  
(independent)

2. Normalizăm

$$\hat{x}^{(k)} = \frac{x^{(k)} - \text{E}[x^{(k)}]}{\sqrt{\text{Var}[x^{(k)}]}}$$

[Ioffe and Szegedy, 2015]

# Normalizarea Batch



Se inserează de obicei după straturile “fully connected” sau după cele conoluționale, înainte de non-liniarități.

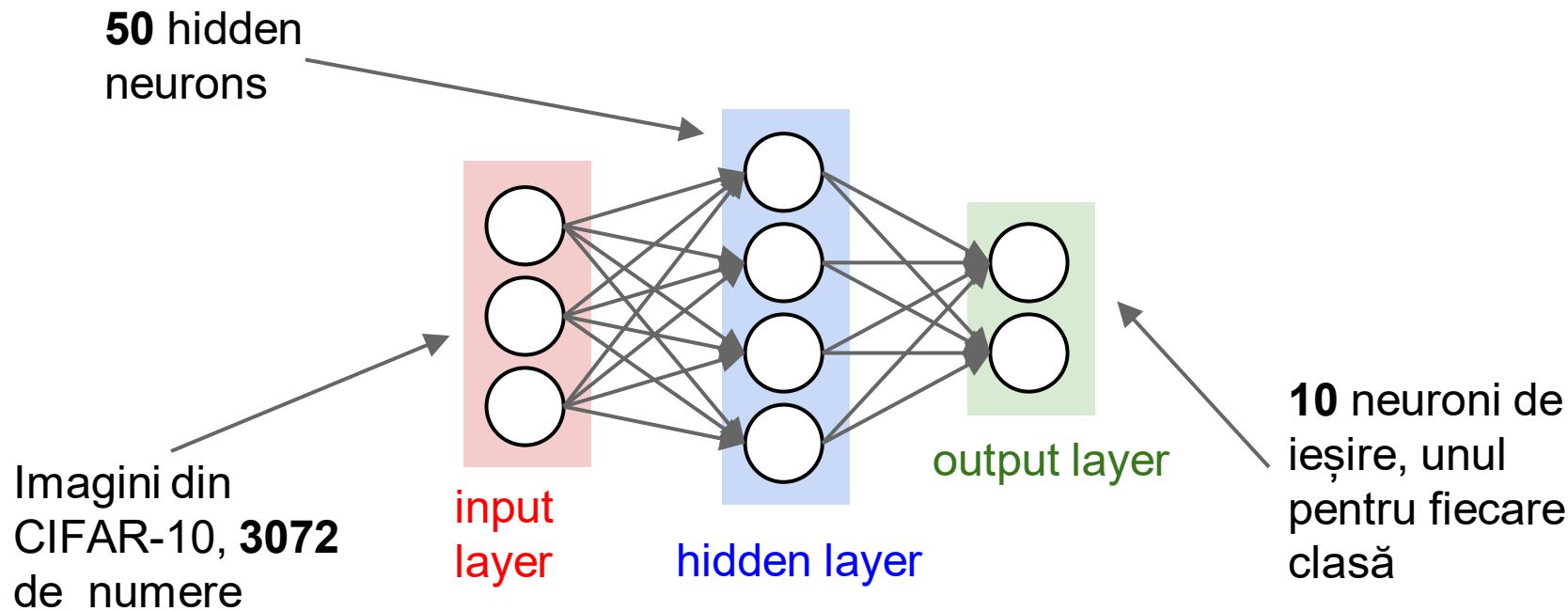
$$\hat{x}^{(k)} = \frac{x^{(k)} - \text{E}[x^{(k)}]}{\sqrt{\text{Var}[x^{(k)}]}}$$

[Ioffe and Szegedy, 2015]

# Asistarea procesului de învățare

# Alegerea arhitecturii potrivite

Începem cu un strat ascuns de 50 de neuroni, apoi mărim gradual capacitatea rețelei

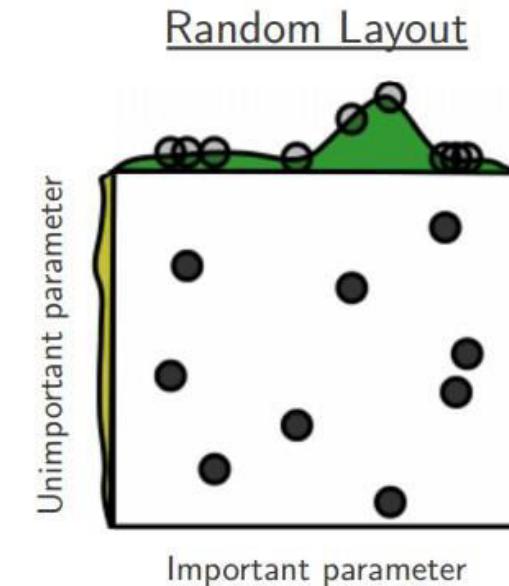
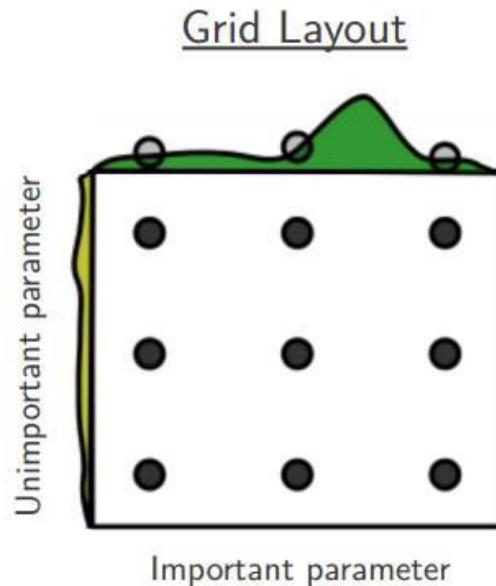


# Sfaturi practice

1. Dezactivăm regularizarea și verificăm dacă valoarea funcției de pierdere este rezonabilă (~2.5 pentru 10 clase este ok)
2. Când adăugăm regularizare, valoarea funcției de pierdere ar trebui să crească, e.g. 3.2
3. Ne asigurăm că putem face overfitting pe o parte mică din setul de antrenare (e.g. 20 de exemple)

# Optimizarea hiperparametrilor

# Strategii de căutare: aleator versus grid



*Random Search for Hyper-Parameter Optimization*  
Bergstra and Bengio, 2012

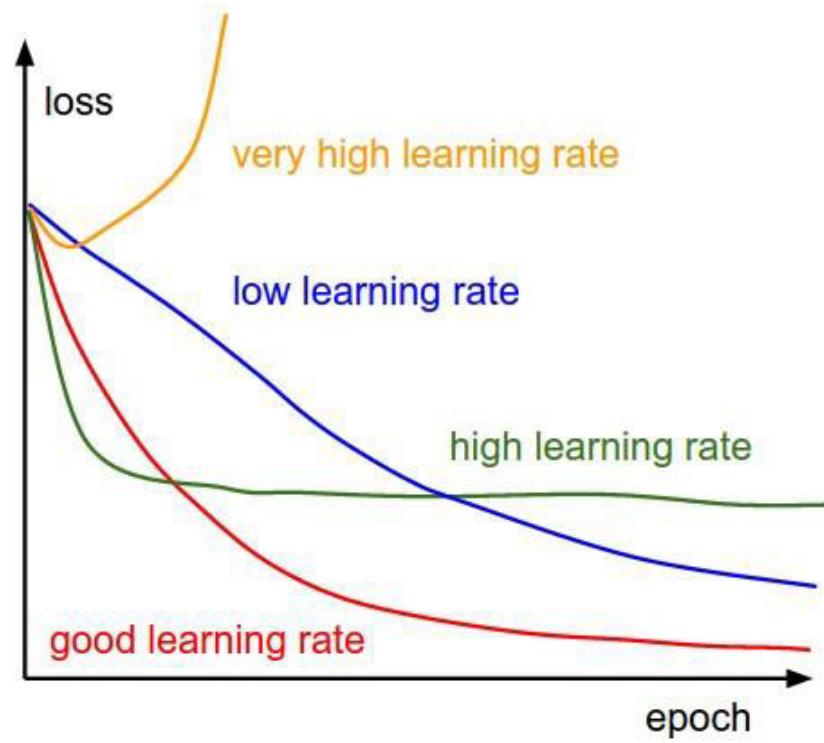
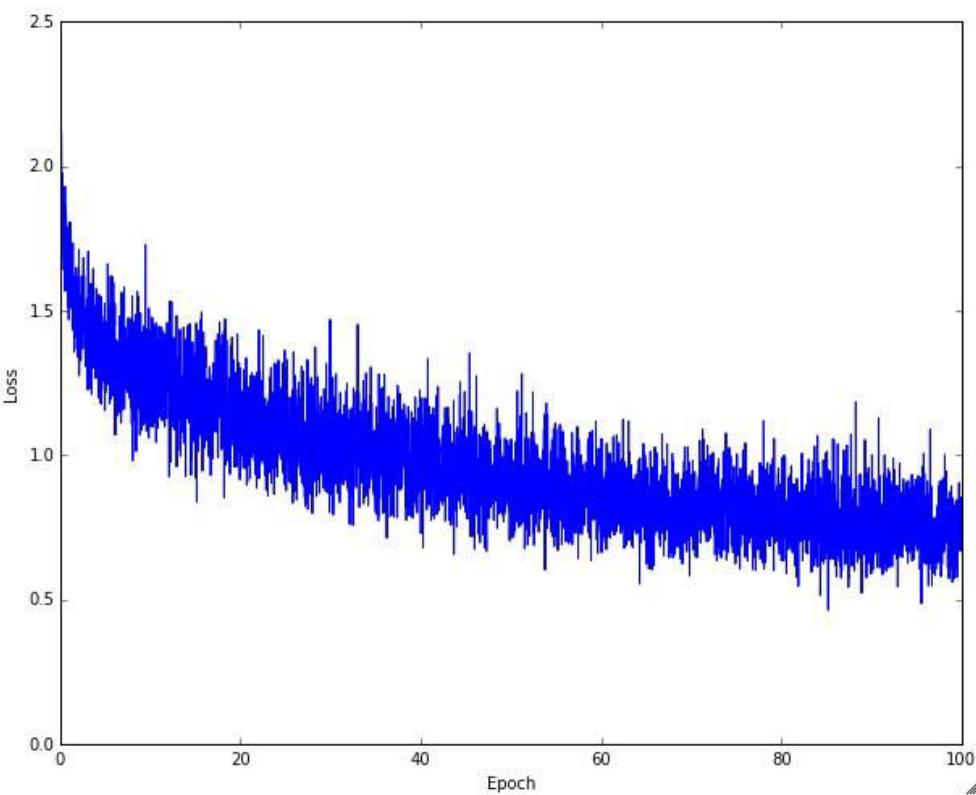
# Hiperparametrii care pot fi optimizați

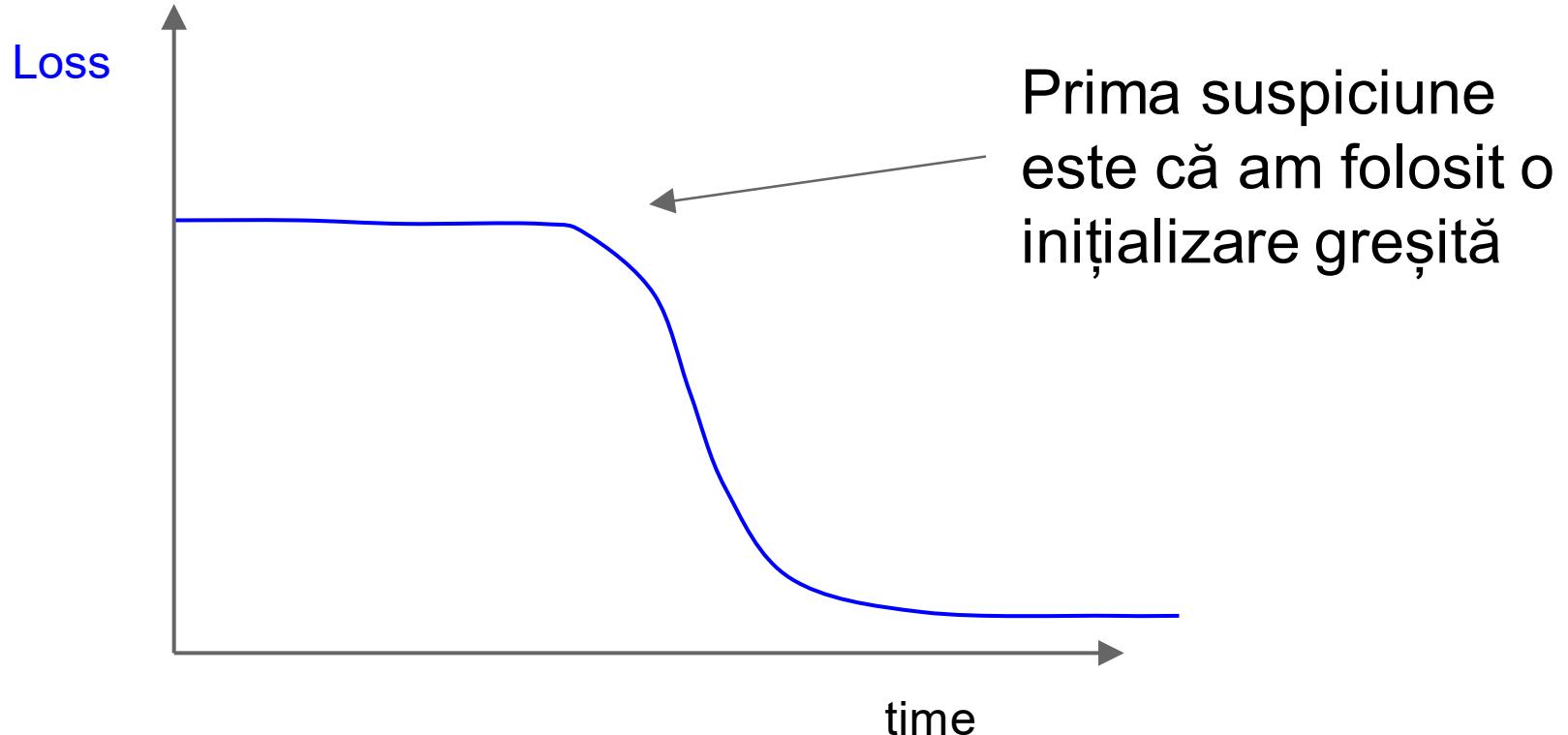
- Arhitectura rețelei
- Rata de învățare, cum se degradează rata (decay)
- Algoritmul de învățare: SGD, SGD cu moment, etc.
- Regularizarea (L2 / Dropout)

Lucrul cu rețelele neuronale  
(muzica = funcția de pierdere)

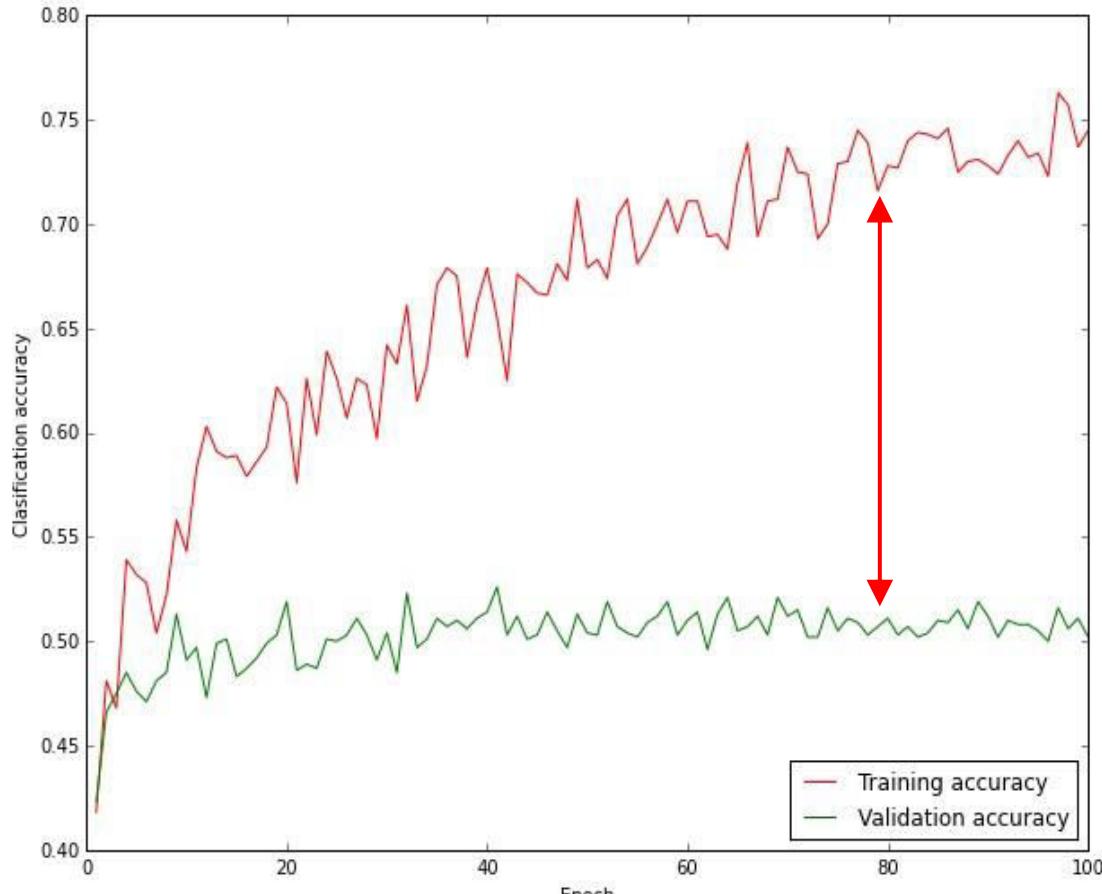


# Monitorizăm evoluția funcției de pierdere





# Monitorizăm evoluția acurateții



distanță mare = overfitting  
=> Creștem regularizarea?

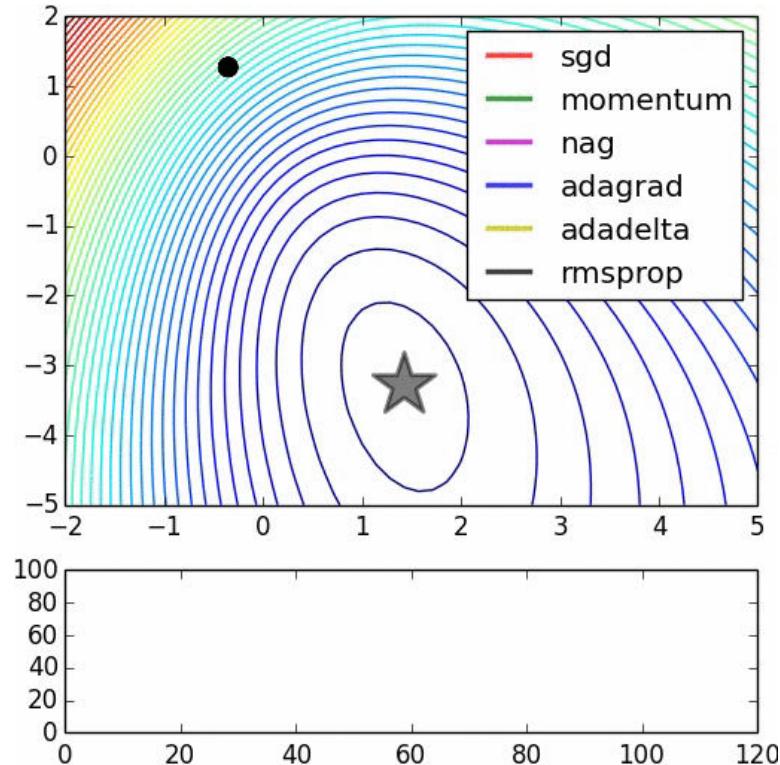
distanță foarte mică  
=> Creștem capacitatea modelului?

# Sfaturi practice (până acum)

- Funcții de activare (**folosim ReLU**)
- Preprocesarea datelor (**imagini: scădem media**)
- Inițializarea ponderilor (**folosim Xavier**)
- Batch Normalization (**folosim**)
- Asistarea procesului de învățare
- Optimizarea hiperparametrilor (**încercări aleatoare**)

# Algoritmul de optimizare

# Există diverse variante ale algoritmului de antrenare



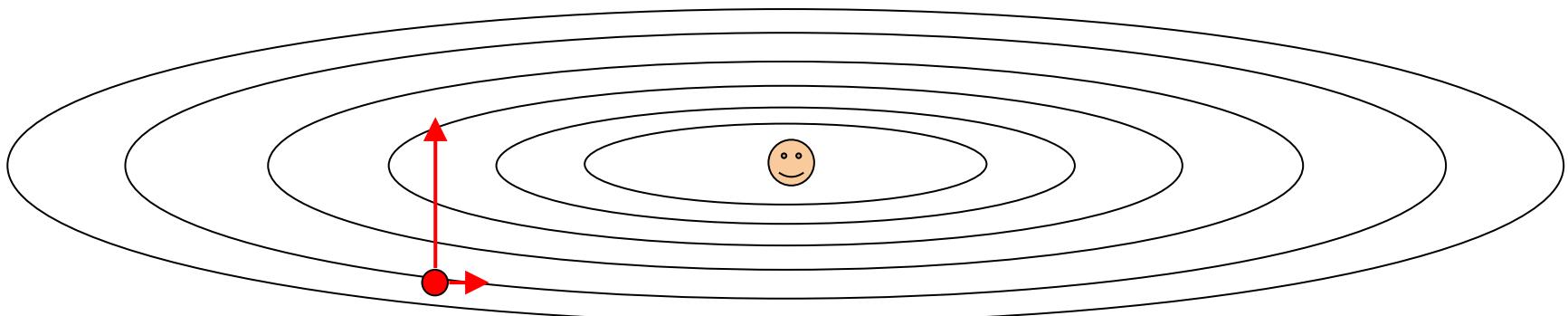
Imagine de Alec Radford

# Algorimtul coborârii pe gradient (Python)

```
def GD(W0, X, goal, learningRate):
    perfGoalNotMet = true
    W = W0

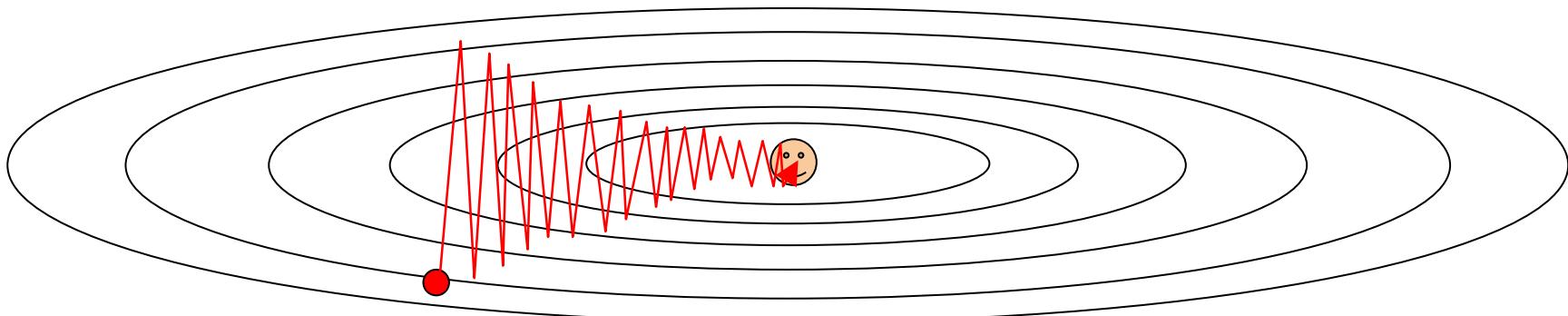
    while perfGoalNotMet:
        gradient = eval_gradient(X, W)
        W_old = W
        W = W - learningRate * gradient
        perfGoalNotMet = sum(abs(W - W_old)) > goal
```

Dacă funcție este abruptă pe verticală, dar lină pe orizontală:



Q: Care este traекторia de-a lungul căreia algoritmul SGD converge către minim?

Dacă funcție este abruptă pe verticală, dar lină pe orizontală:



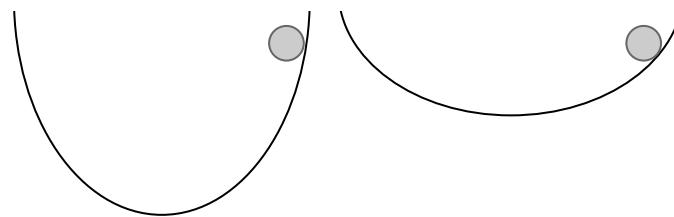
Q: Care este traекторia de-a lungul căreia algoritmul SGD converge către minim? Progres încet pe direcția cu pantă lină, zig zag pe direcția abruptă

# Algoritmul SGD cu moment (Python)

```
def GD(W0, X, goal, learningRate, mu):
    perfGoalNotMet = true
    W = W0
    V = 0
    while perfGoalNotMet:
        gradient = eval_gradient(X, W)
        W_old = W
        V = mu * V - learningRate * gradient
        W = W + V
        perfGoalNotMet = sum(abs(W - W_old)) > goal
```

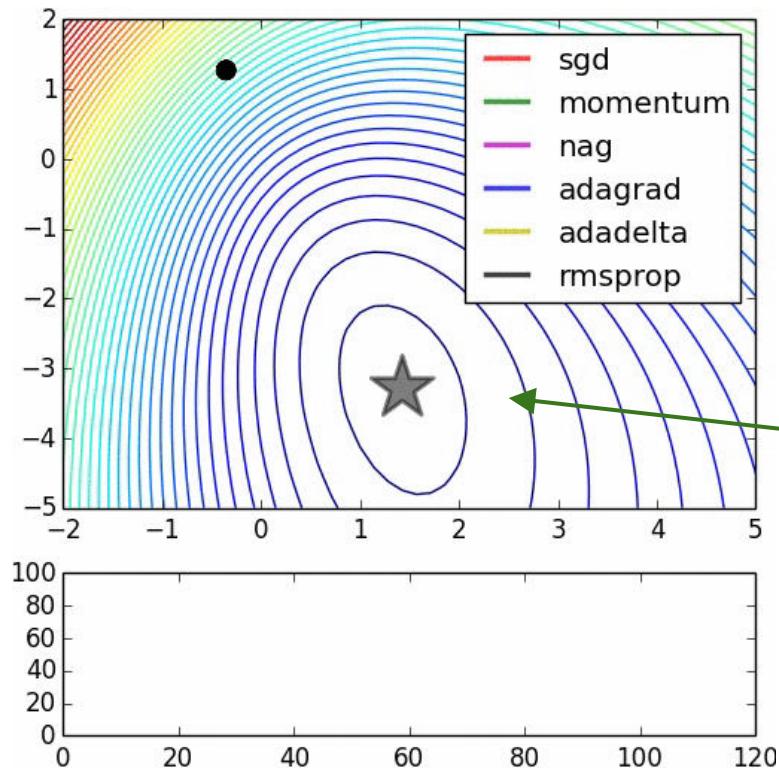
# Algorimtul SGD cu moment

- Interpretarea fizică a unei mingi care se rostogolește pe funcția de pierdere
- Forța de frecare este dată de coeficientul  $\mu$
- $\mu$  = deobicei în jur de ~0.9, 0.95 sau 0.99 (câteodată se modifică în timp, e.g. de la 0.5 către 0.99)



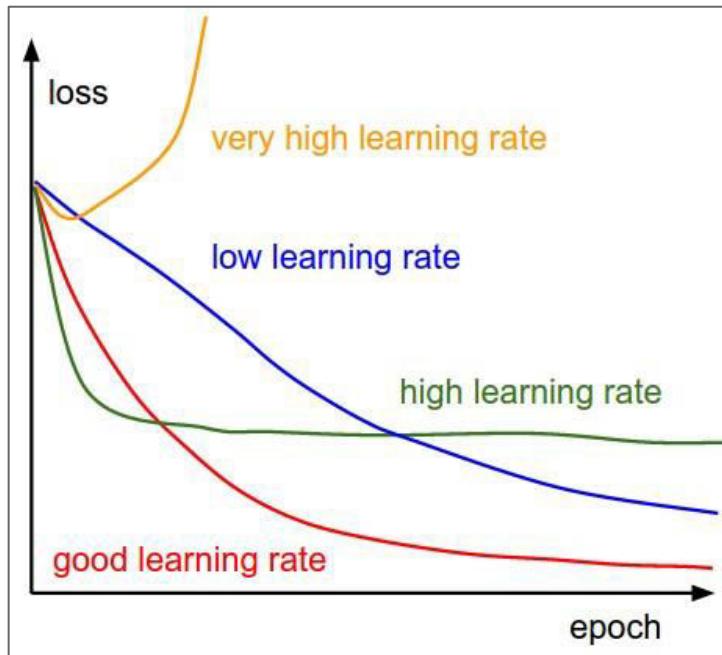
- Permite acumularea vitezei de-a lungul direcțiilor cu pantă lină
- Viteza se amortizează de-a lungul direcției abrupte din cauza schimbării dese a semnului / direcției de coborâre

# SGD vs SGD cu moment



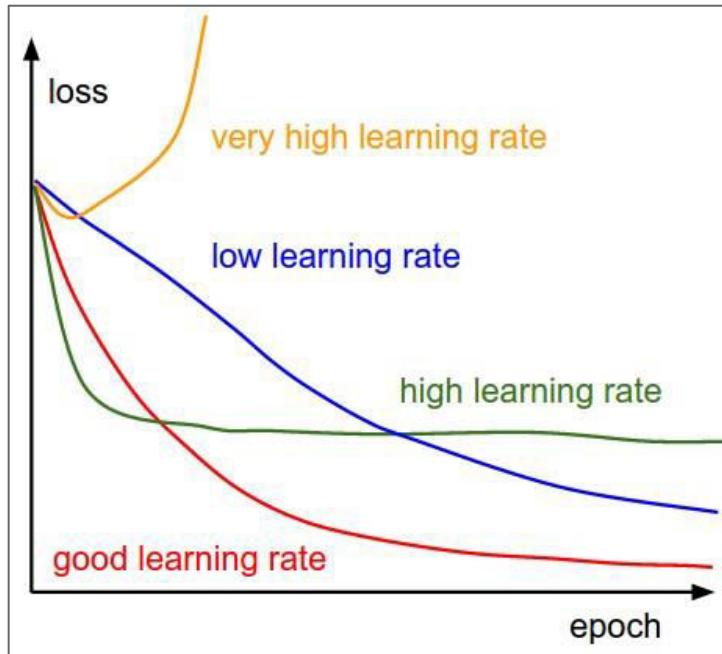
Observăm cum SGD cu moment depășește ținta, dar per total ajunge la minimul local mult mai rapid

# Rata de învățare este un hiperparametru al SGD / SGD cu moment



Q: Care din aceste rate de  
învățare este mai potrivită?

# Rata de învățare este un hiperparametru al SGD / SGD cu moment



=> Declinul ratei de învățare în timp

**step decay:**

e.g. rata de învățare se înjumătățește după fiecare câteva epoci

**exponential decay:**

$$\alpha = \alpha_0 e^{-kt}$$

**1/t decay:**

$$\alpha = \alpha_0 / (1 + kt)$$

Evaluare:  
Ansamble de modele

# Ansamble de modele

1. Antrenăm independent mai multe modele
2. La testare, calculăm media predicțiilor

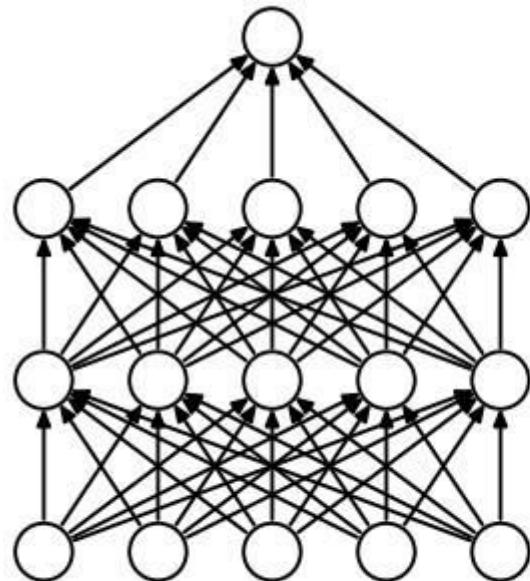
De obicei, acuratețea crește cu ~2%

Sfat practic: o mică îmbunătățire se poate obține și prin calcularea mediei predicțiilor date de un singur model, salvat la momente de timp diferite în timpul antrenării

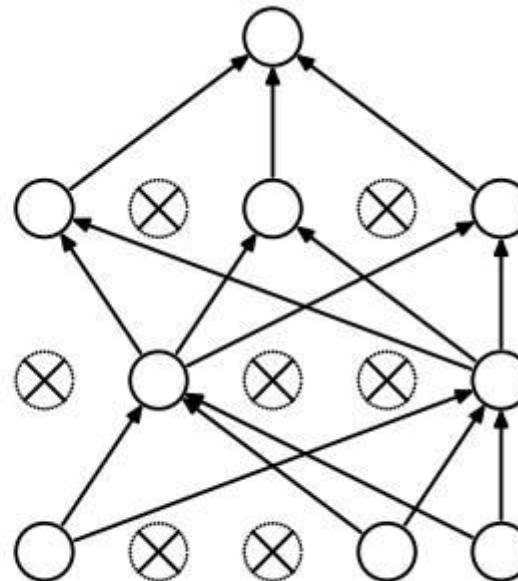
Regularizare folosind  
**Dropout**

# Regularizarea folosind Dropout

Atribuim în mod aleator ponderi egale cu zero pentru o parte din neuroni (echivalent cu a deconecta o parte din neuroni)



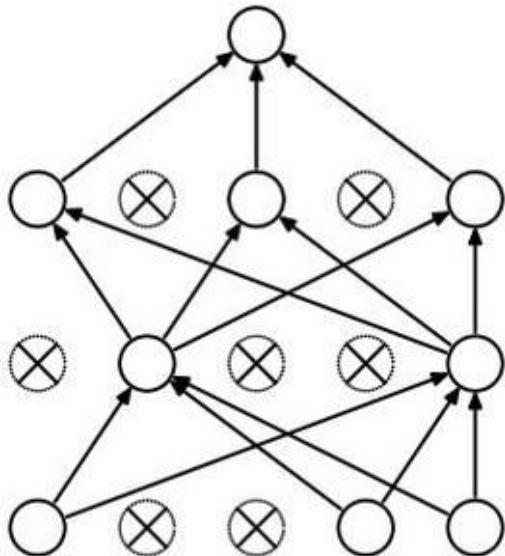
(a) Standard Neural Net



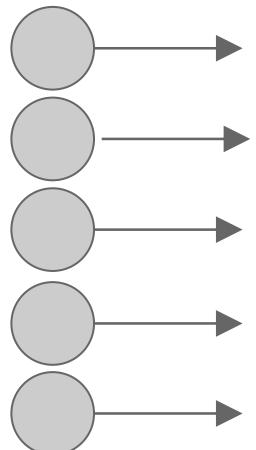
(b) After applying dropout.

[Srivastava et al., 2014]

# Cum ar putea fi asta o idee bună?



Forțează rețeaua să producă o reprezentare redundantă



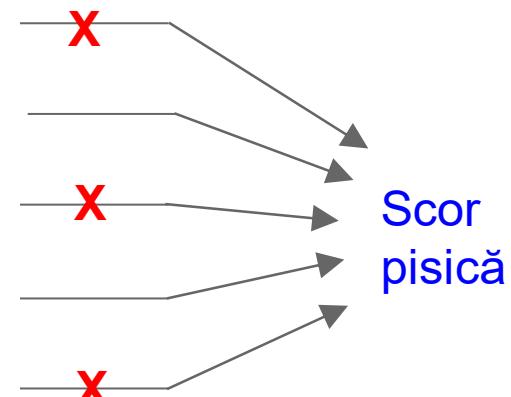
Are ureche

Are coadă

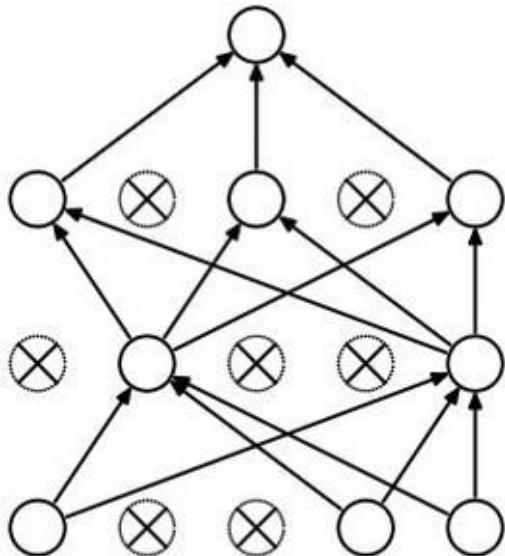
E blănos

Are gheare

Arată înfriicoșător



# Cum ar putea fi asta o idee bună?

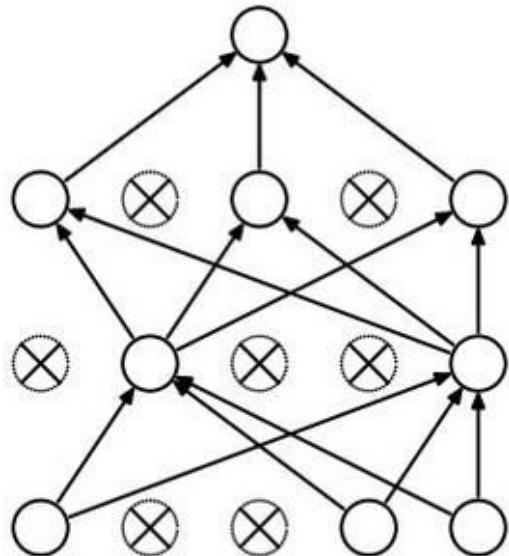


O altă interpretare:

Dropout este echivalent cu antrenarea unui ansamblu de multe modele (care au în comun parametrii)

Fiecare mască binară produce un model care se antrenează pe un exemplu / mini-batch

# La testare...



**Ideal:**

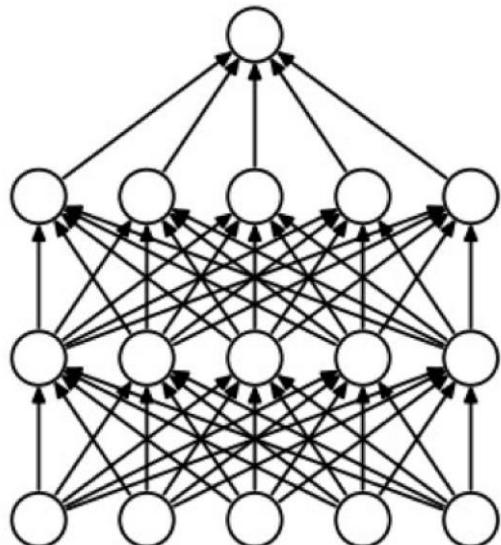
Vrem să eliminăm tot zgomotul

**Aproximare Monte Carlo:**

Facem mai multe treceri prin rețea folosind diverse măști de dropout, calculând apoi media predictiilor

# La testare...

Putem face totul printr-o singură trecere! (aproximativ)



Activăm toți neuronii (fără dropout)  
(acestă variantă poate fi  
interpretată ca o aproximare a  
întregului ansamblu)

# Rețele neuronale convoluționale. Straturi speciale și arhitecturi.

Prof. Dr. Radu Ionescu

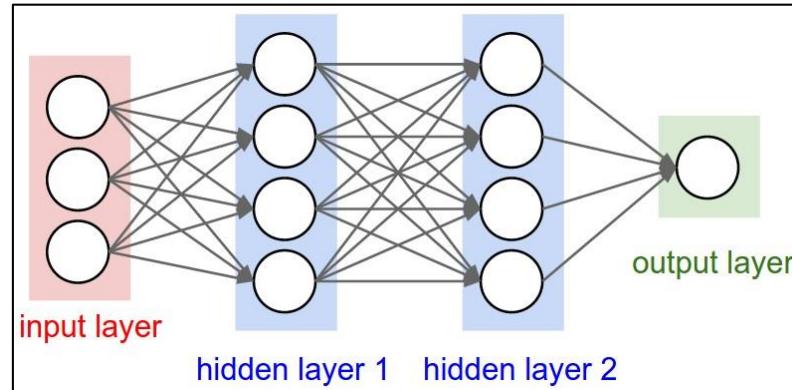
[raducu.ionescu@gmail.com](mailto:raducu.ionescu@gmail.com)

Facultatea de Matematică și Informatică  
Universitatea din București

# SGD cu mini-batch

Repetă:

1. Selectăm un mini-batch de exemple
2. Propagăm **înainte** prin graf pentru a obține pierderea
3. Propagăm **înapoi** pentru a calcula gradienții
4. Actualizăm ponderile pe baza gradienților calculați



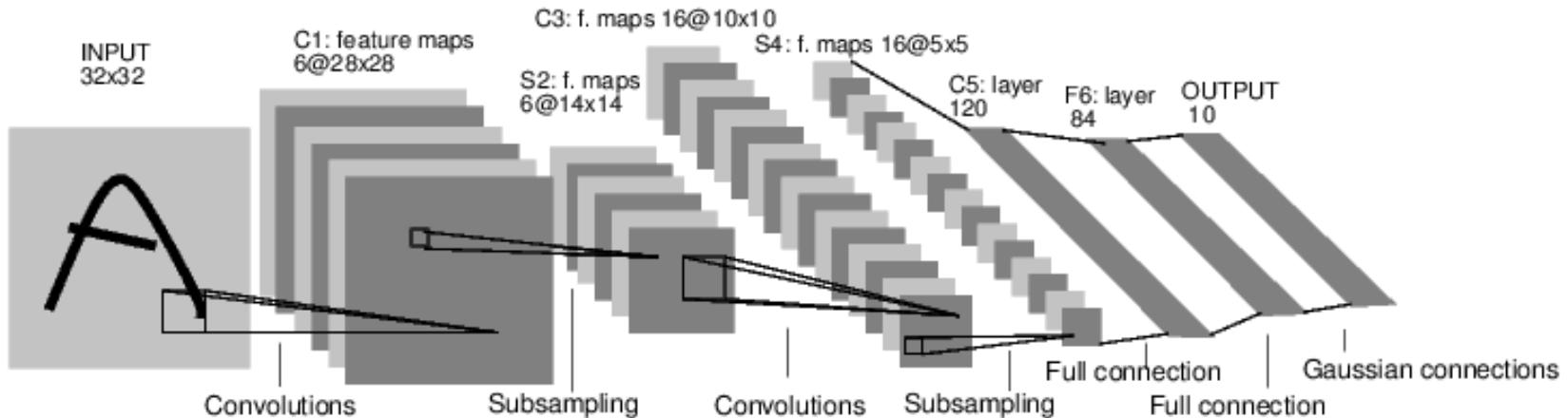
# Rețele neuronale sunt funcții universale de aproximare

- **Teorema Aproximării Universale:**

O rețea neuronală de tip feed-forward cu un strat ascuns având un număr finit de neuroni poate aproxima orice funcție continuă definită pe un subset compact din  $\mathbb{R}^n$ .

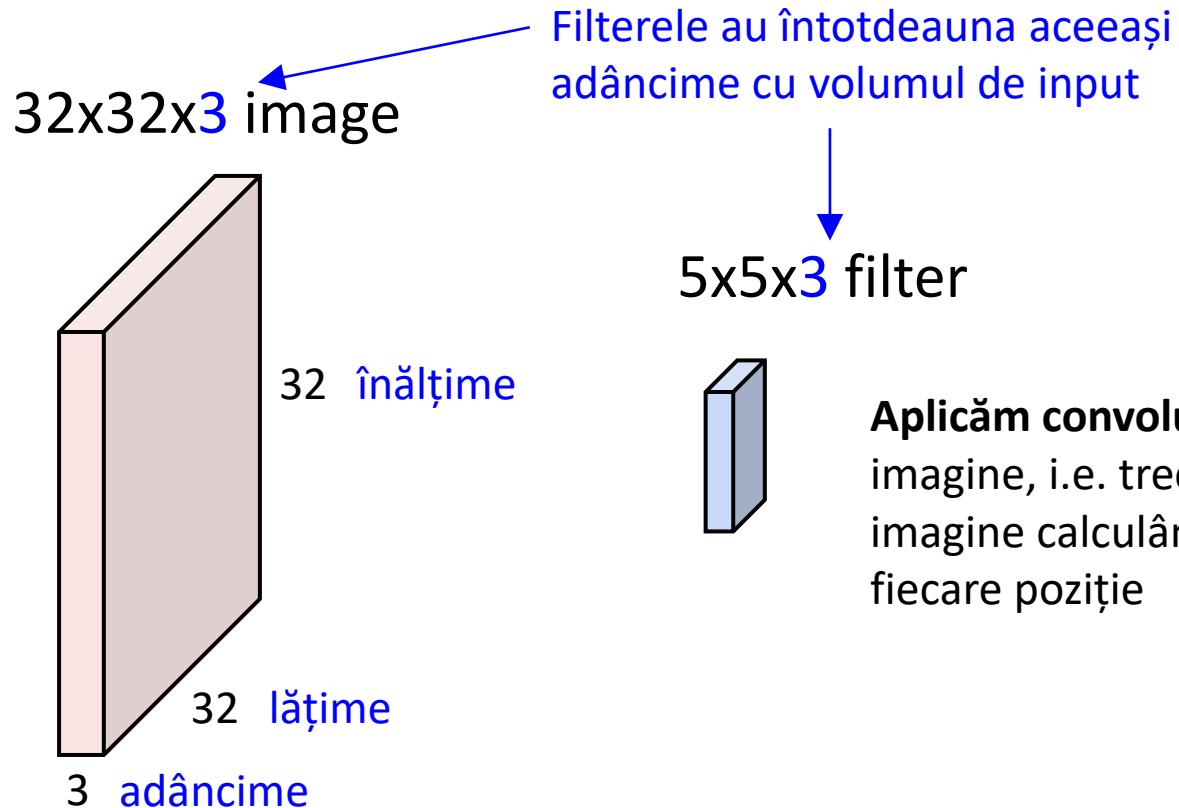
- Deși rețelele neuronale cu două straturi (un strat ascuns) sunt funcții universale de aproximare, lățimea (numărul de perceptri) acestor rețele poate fi exponențial de mare.
- **În practică, preferăm rețelele mai adânci (cu mai multe straturi)**

# Retele neuronale convolutionale

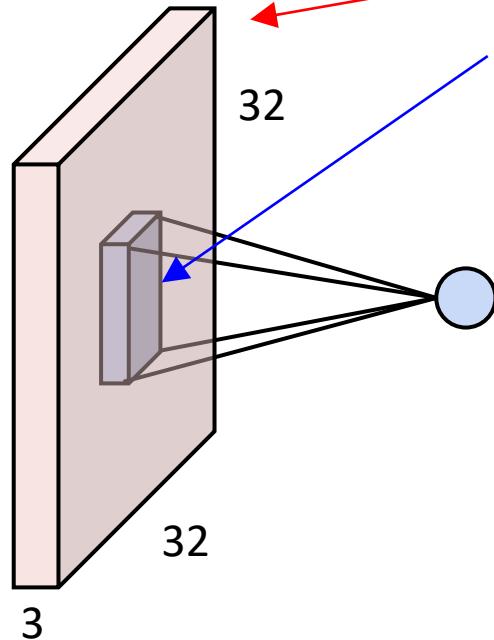


[LeNet-5, LeCun 1998]

# Stratul conoluțional



# Stratul conveoluțional



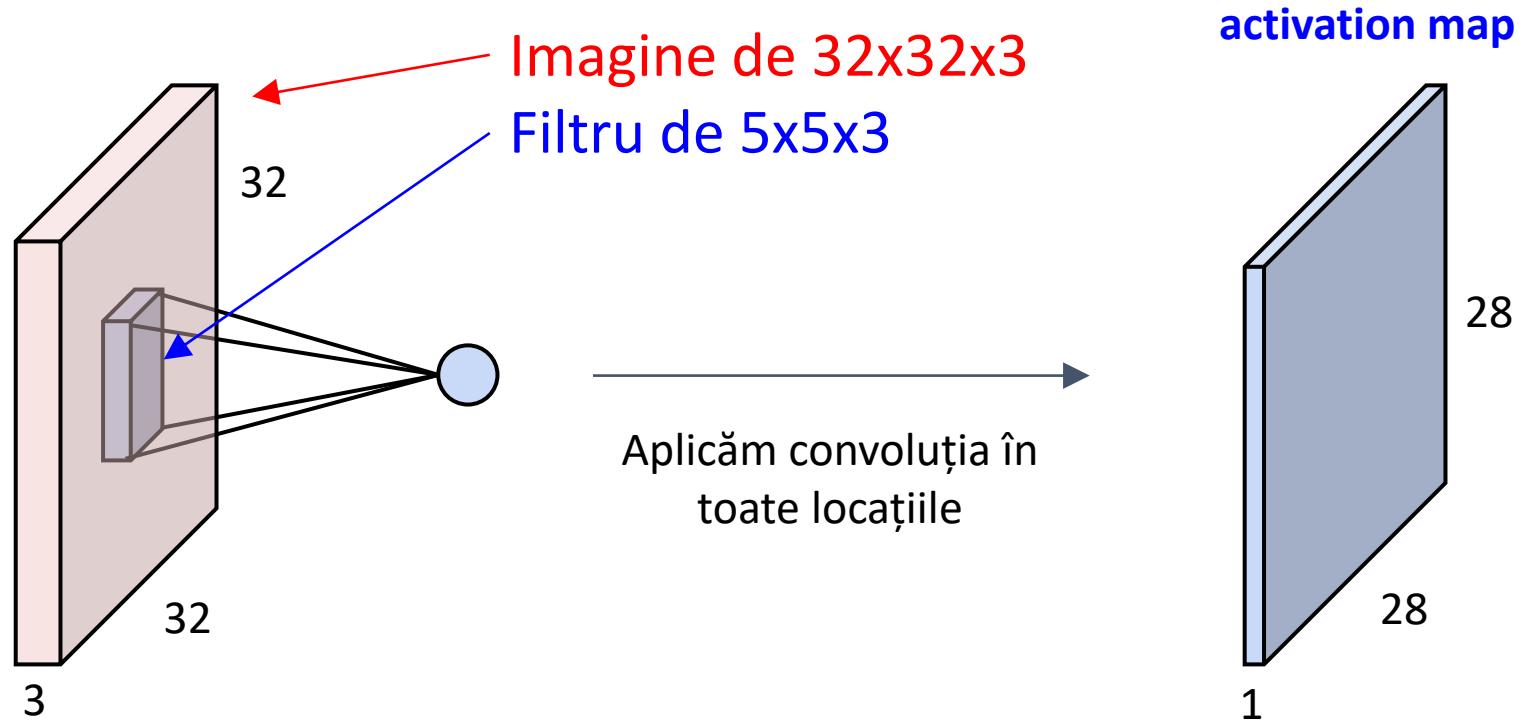
Imagine de  $32 \times 32 \times 3$   
Filtru de  $5 \times 5 \times 3$   $w$

un număr:

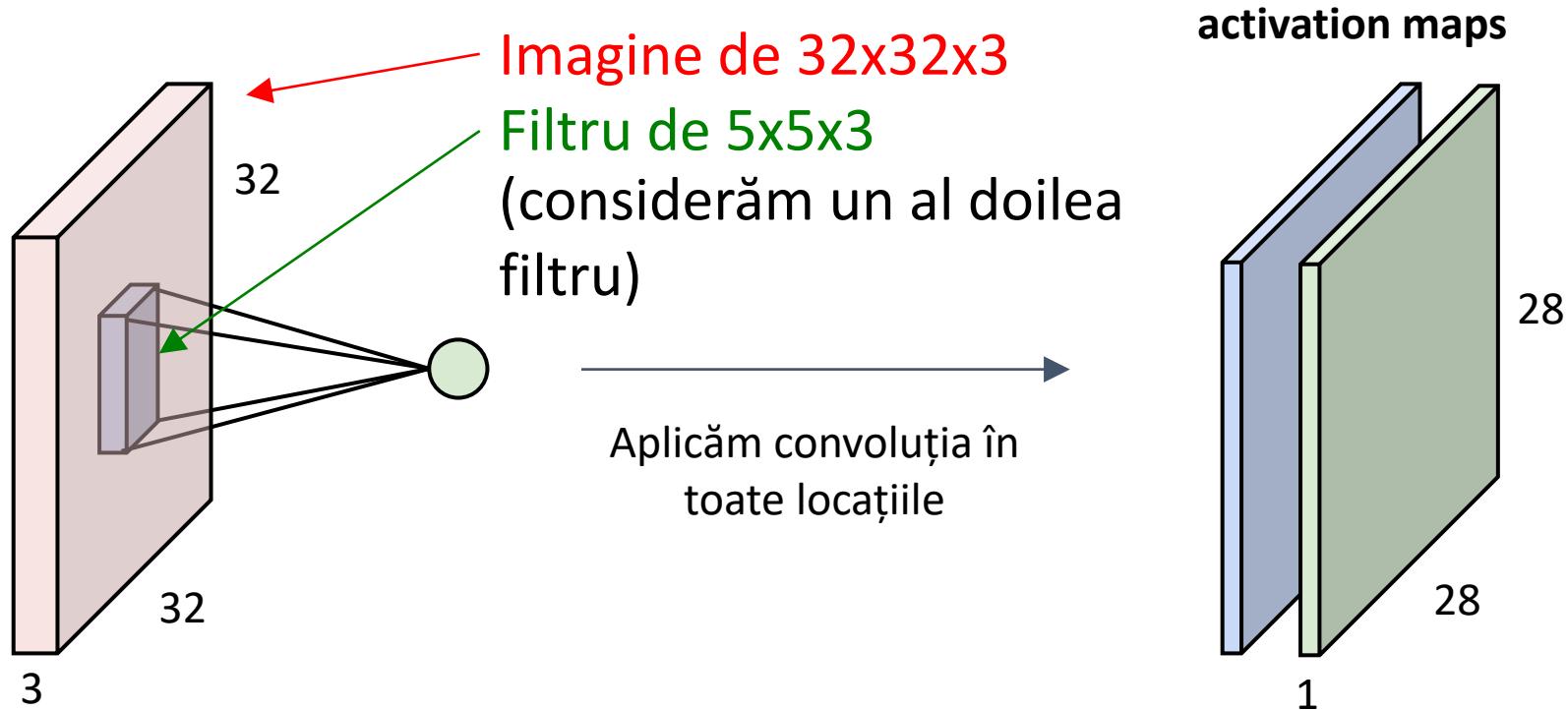
Rezultatul produsului scalar dintre filtru și o  
subimagine de  $5 \times 5 \times 3$  pixeli  
(i.e.  $5 \times 5 \times 3 =$  produs scalar pe 75 de componente +  
bias)

$$w^T x + b$$

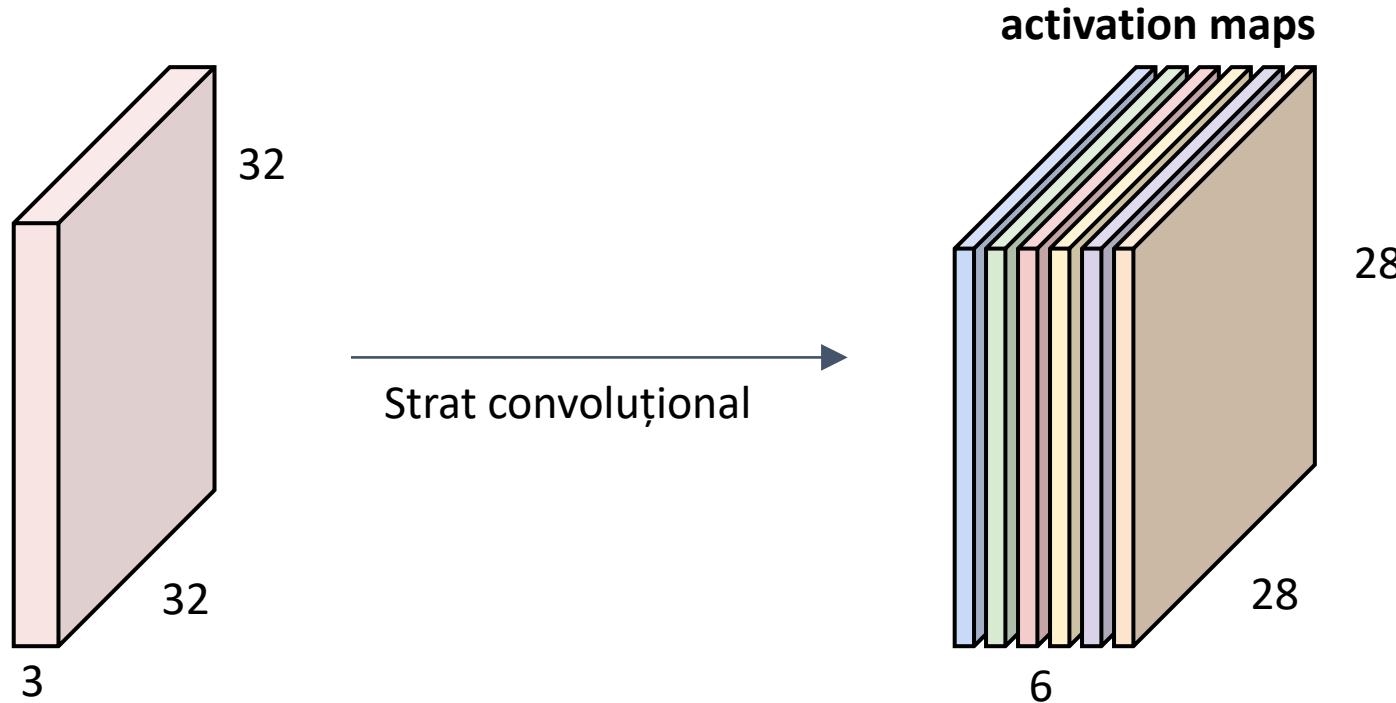
# Stratul conoluțional



# Stratul conoluțional

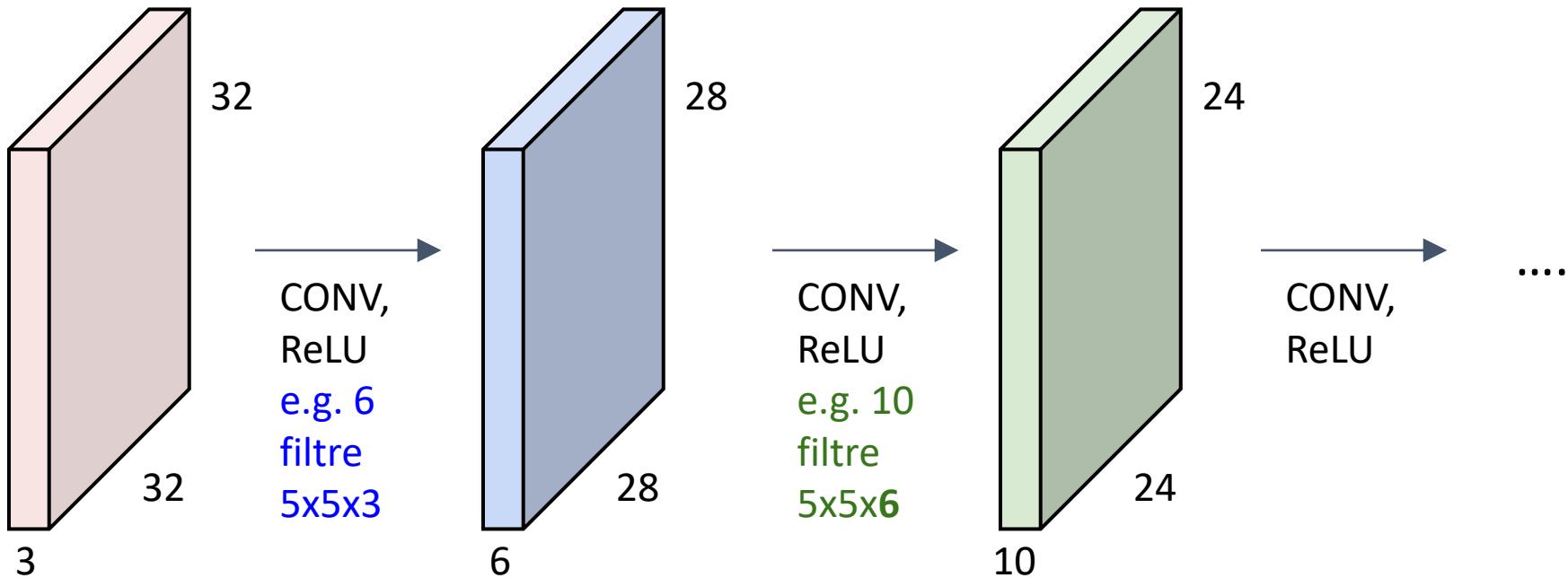


Un strat conoluțional este format din mai multe filtre (de exemplu 6)

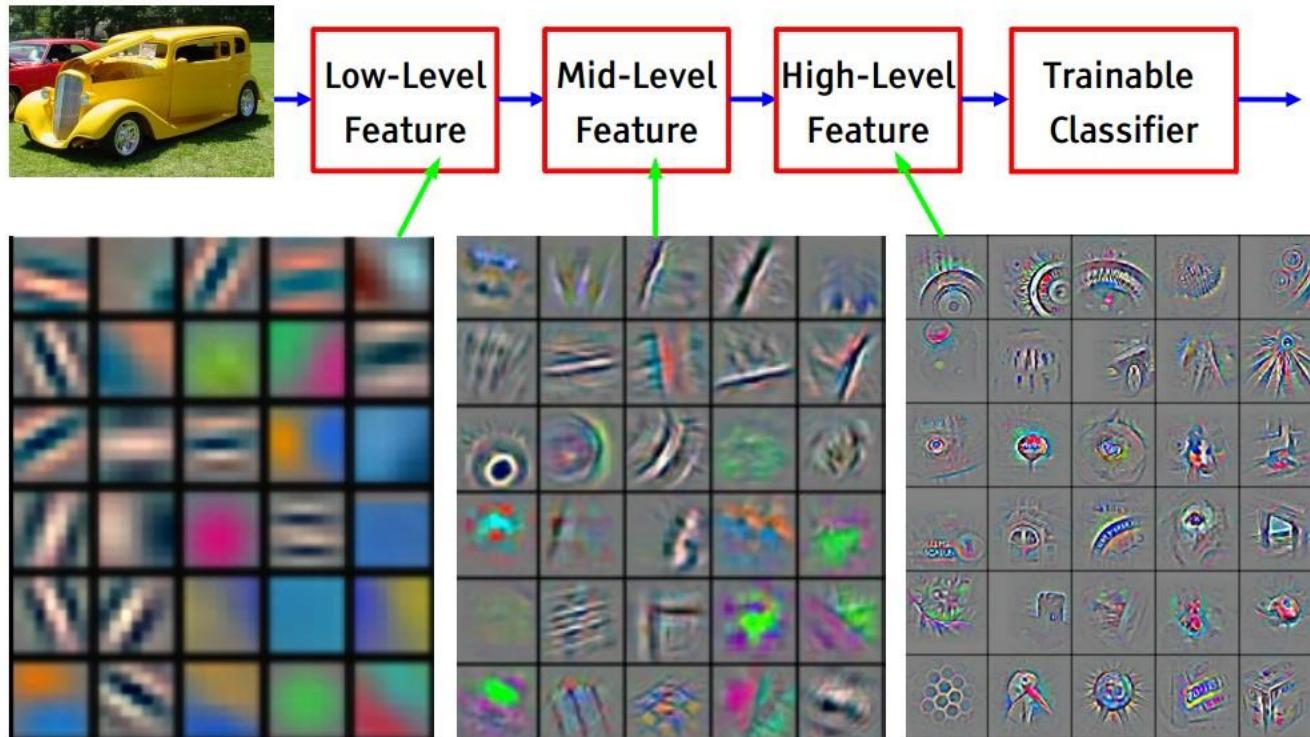


Concatenăm activările pentru a obține o nouă “imagine” de  $28 \times 28 \times 6$

În esență o rețea conoluțională (CNN sau ConvNet) este o formată dintr-o secvență de straturi conoluționale, între care interpunem funcții de activare

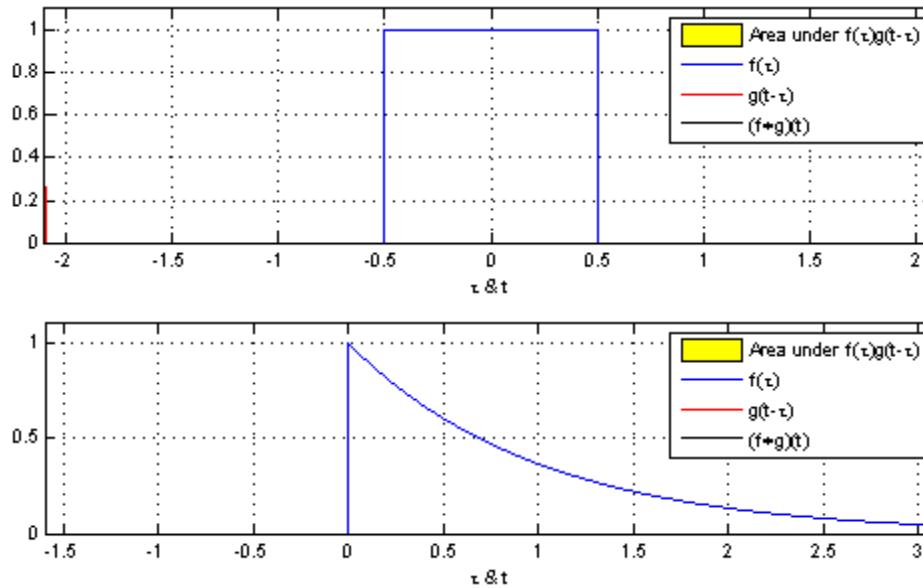


Filtrele corespond unor un trăsături ale obiectelor



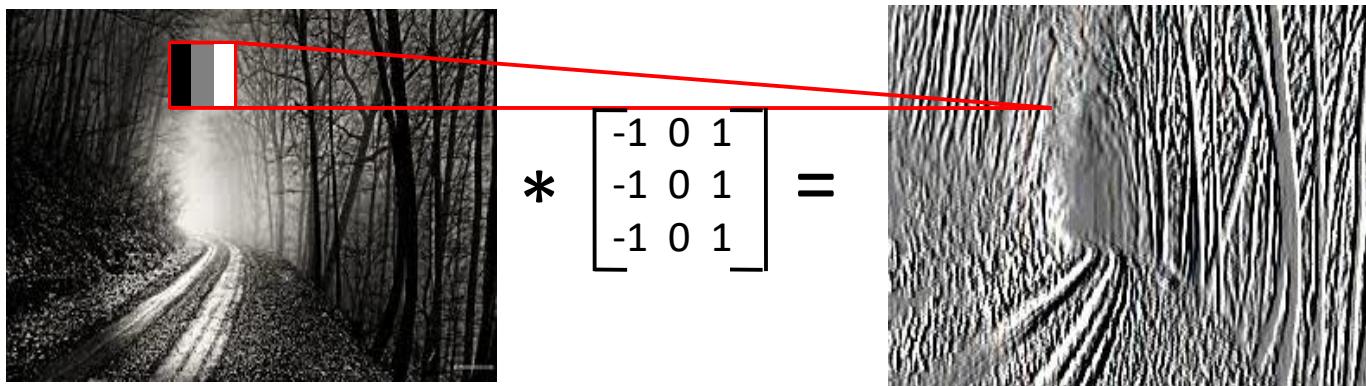
Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

# Activare maximă = răspunsul cel mai mare



[https://commons.wikimedia.org/wiki/File:Convolution\\_of\\_box\\_signal\\_with\\_itself.gif](https://commons.wikimedia.org/wiki/File:Convolution_of_box_signal_with_itself.gif)

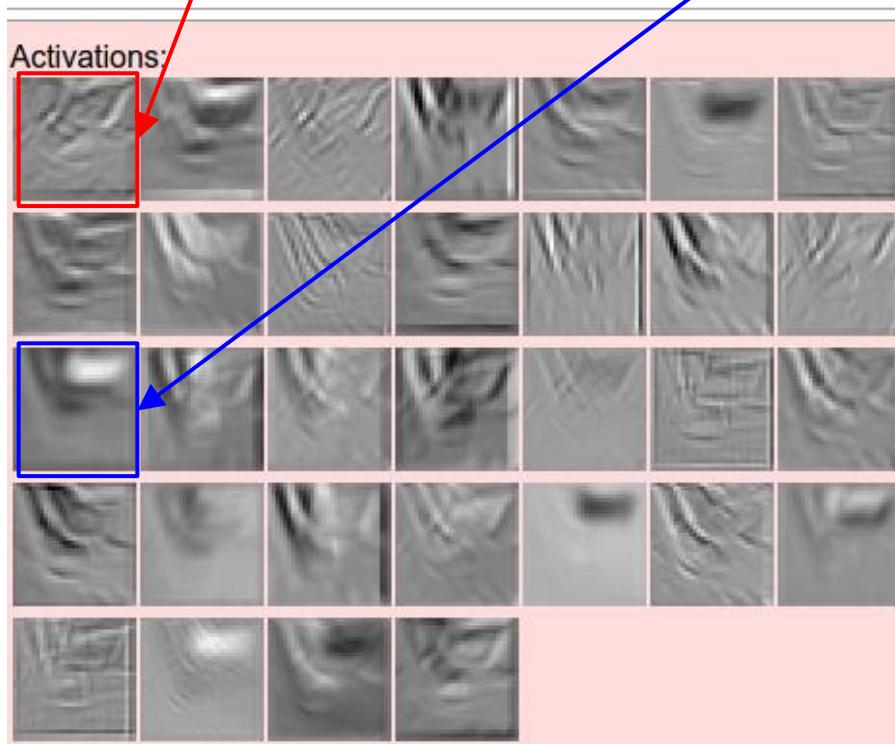
# Activare maximă = răspunsul cel mai mare





un filtru =>  
un activation map

exemplu cu 32 de filtre 5x5



# Vizualizarea filterelor învățate

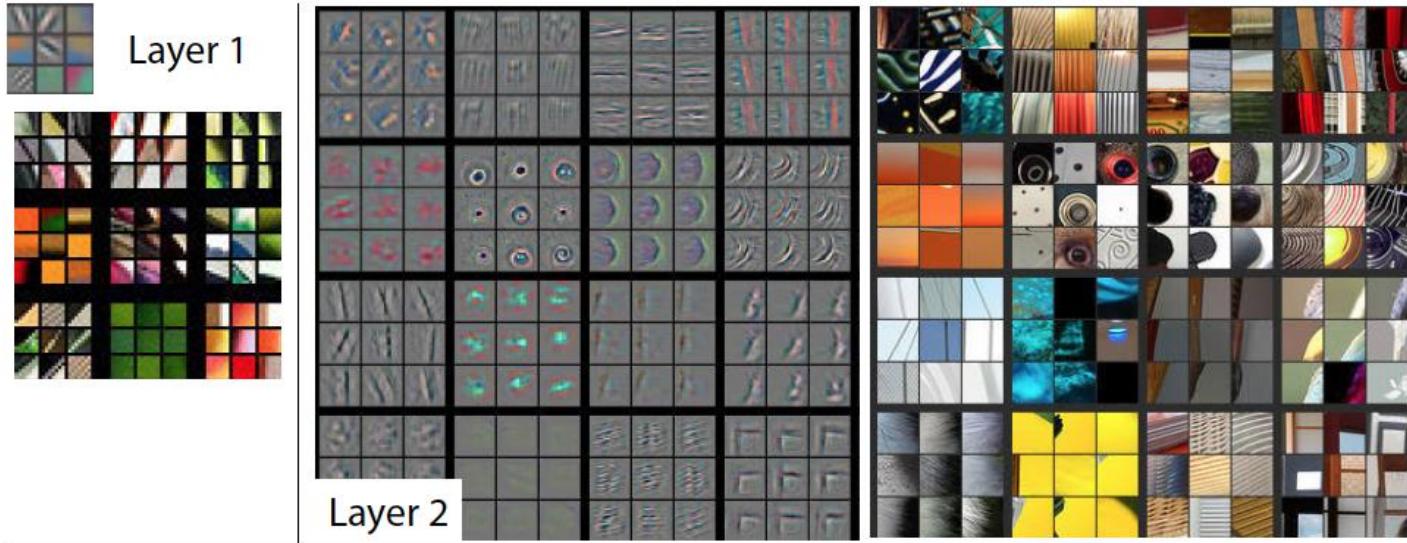


Figure Credit: [Zeiler & Fergus ECCV14]

# Vizualizarea filterelor învățate

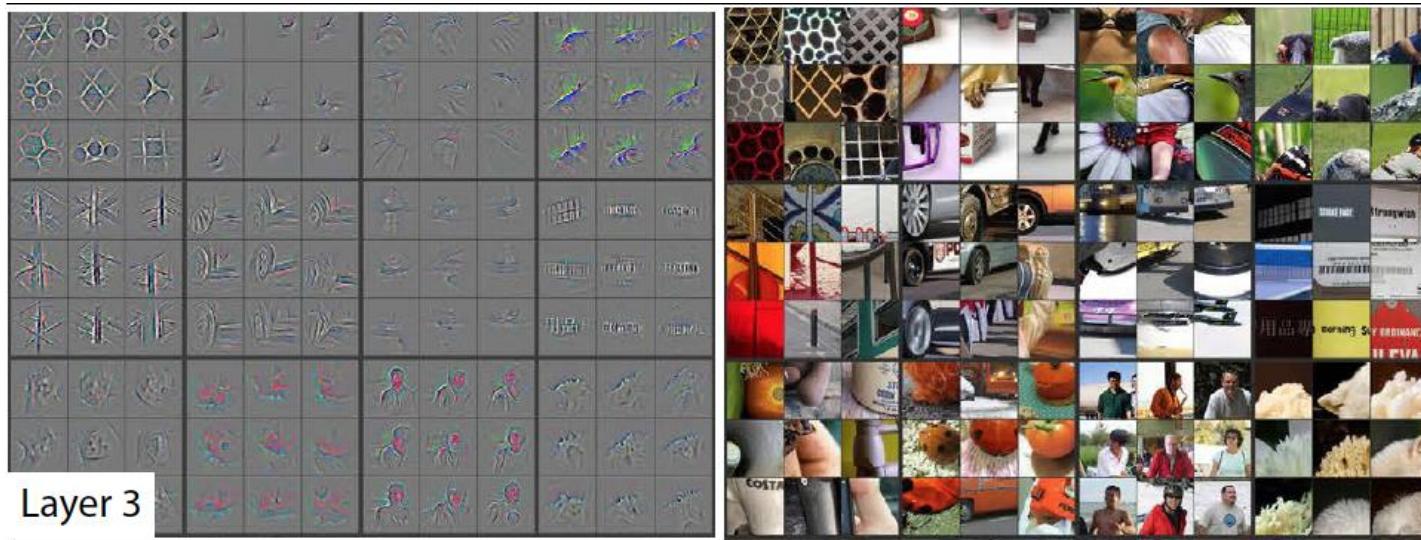


Figure Credit: [Zeiler & Fergus ECCV14]

# Vizualizarea filterelor învățate

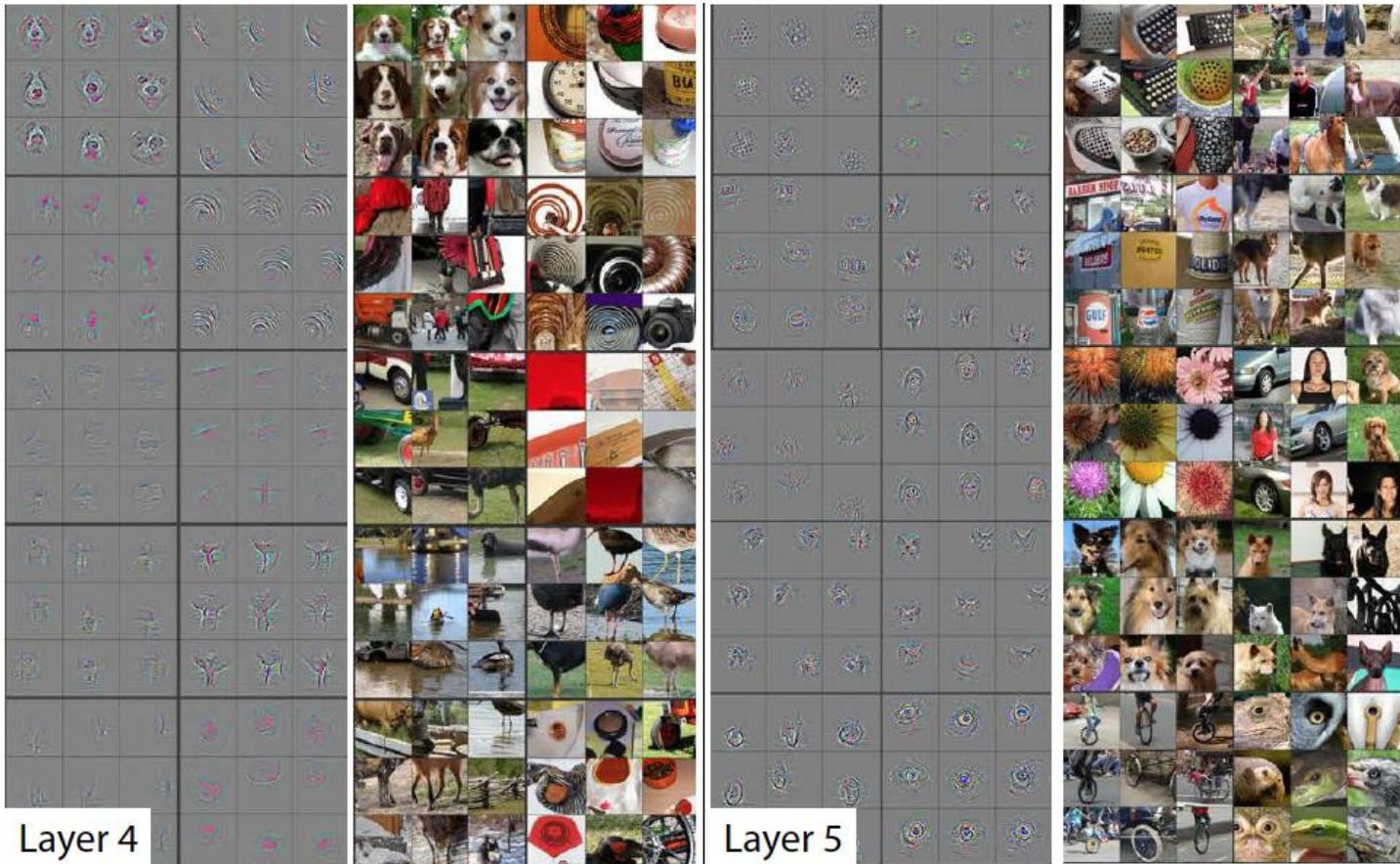
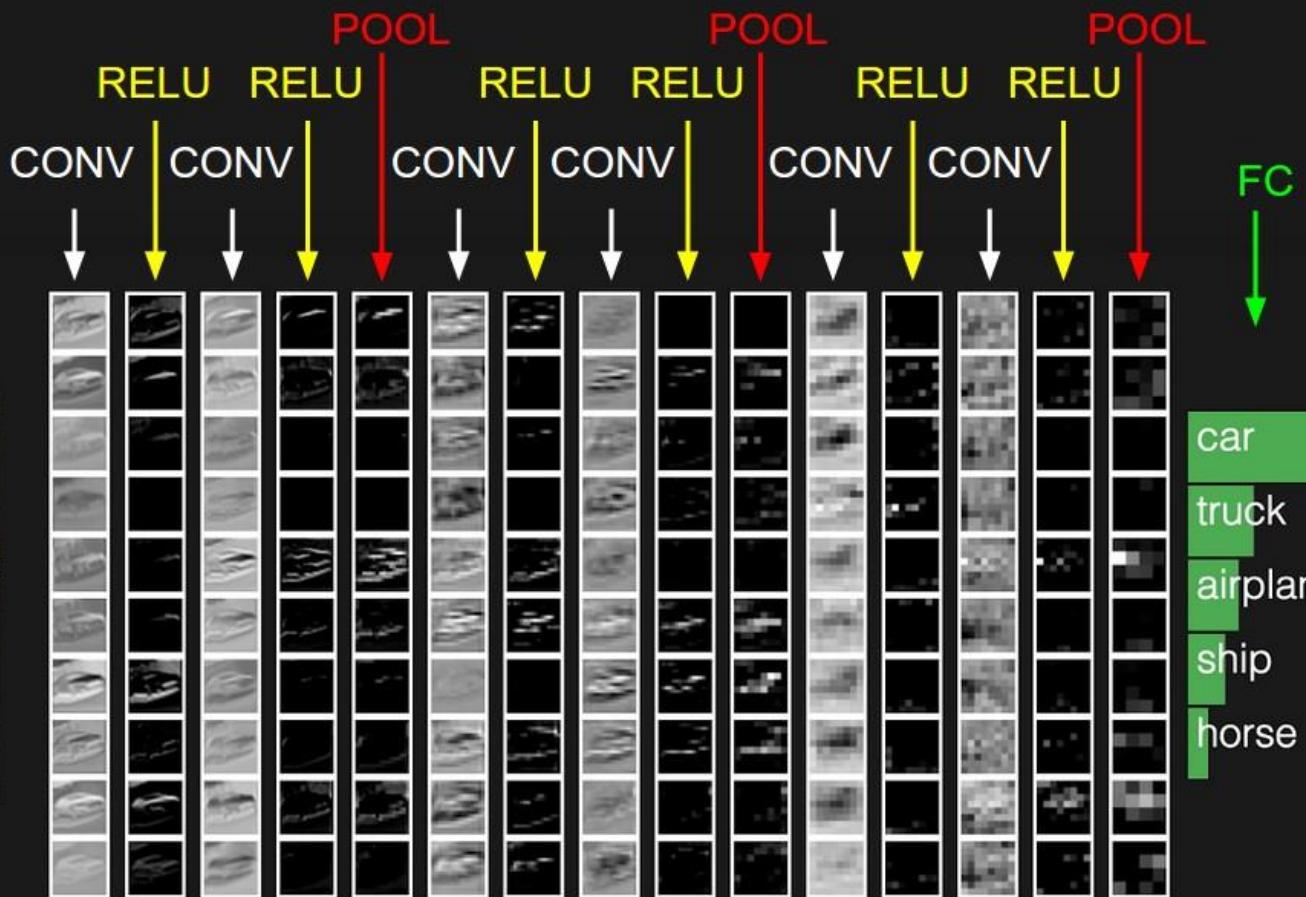
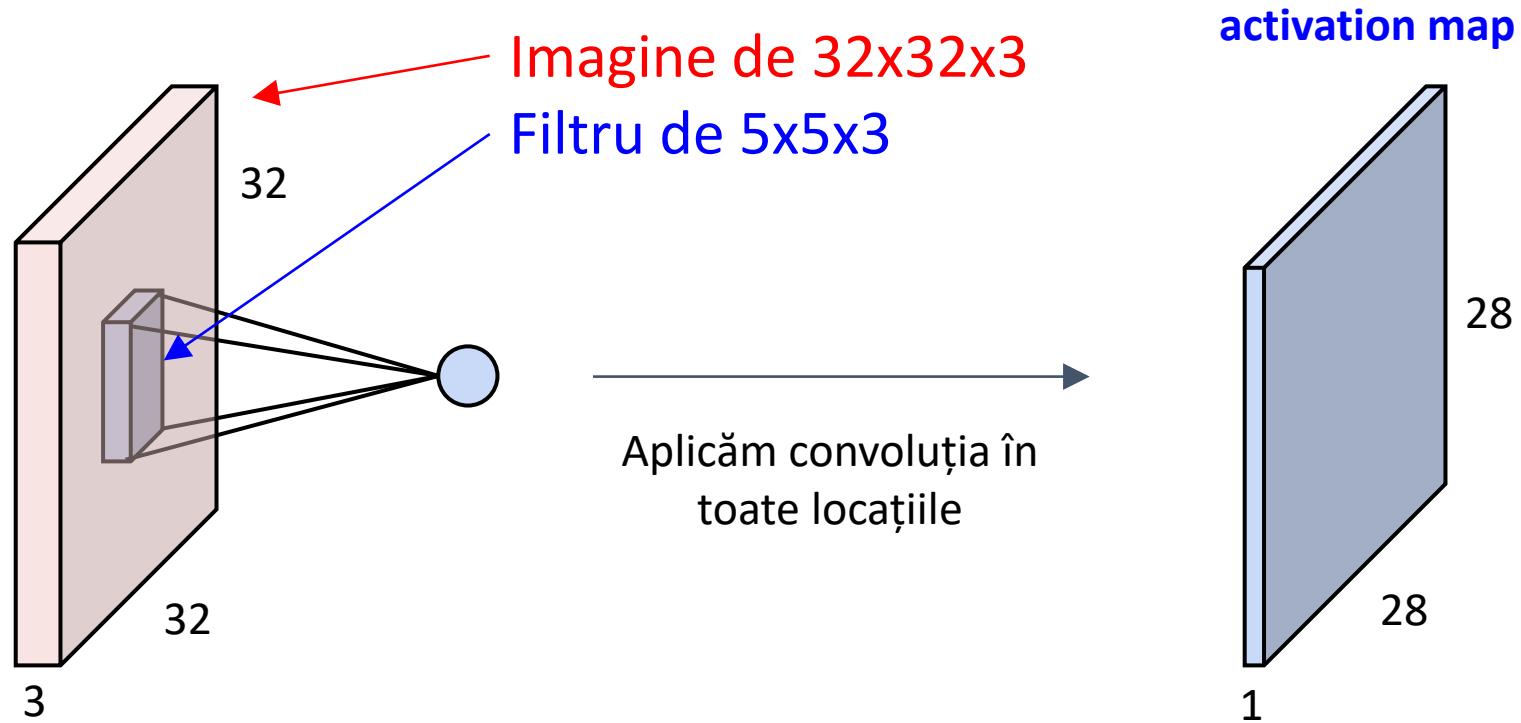


Figure Credit: [Zeiler & Fergus ECCV14]

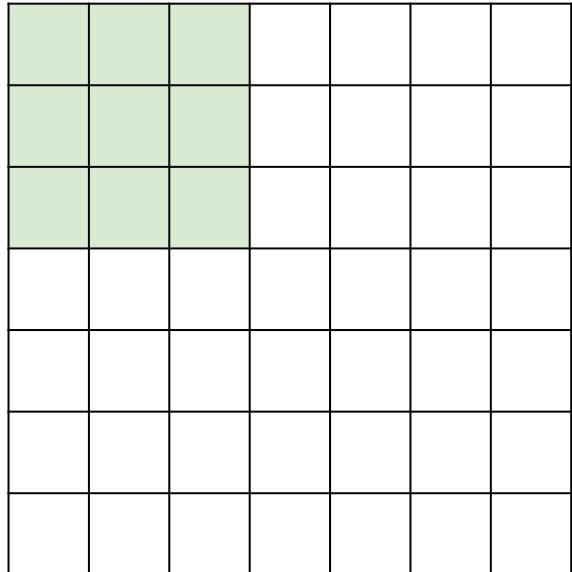


# Dimensiunea spațială a unei activări



# Dimensiunea spațială a unei activări

7

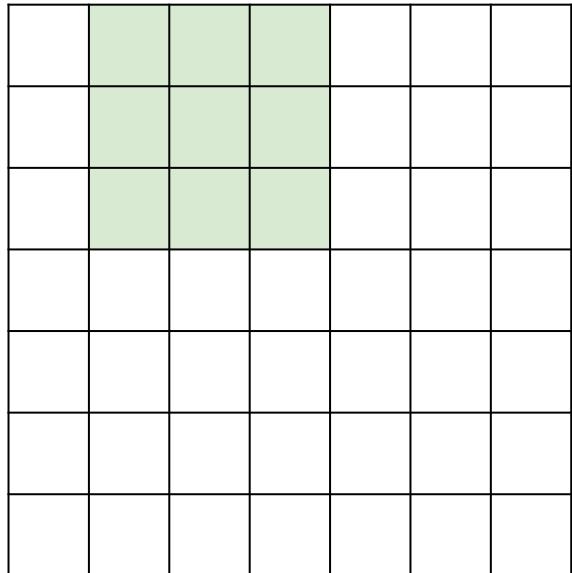


Imagine de  $7 \times 7$  (fără adâncime)  
Filtru de  $3 \times 3$

7

# Dimensiunea spațială a unei activări

7

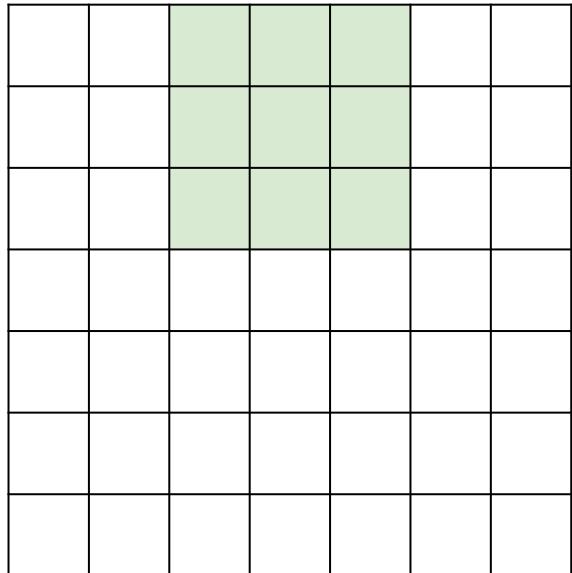


Imagine de  $7 \times 7$  (fără adâncime)  
Filtru de  $3 \times 3$

7

# Dimensiunea spațială a unei activări

7

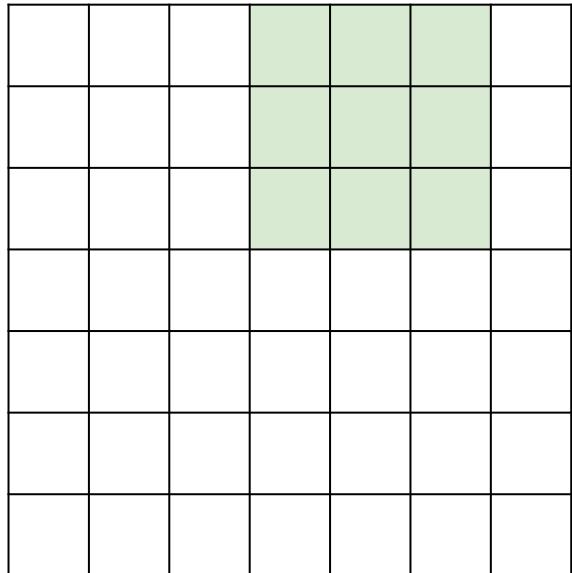


Imagine de  $7 \times 7$  (fără adâncime)  
Filtru de  $3 \times 3$

7

# Dimensiunea spațială a unei activări

7

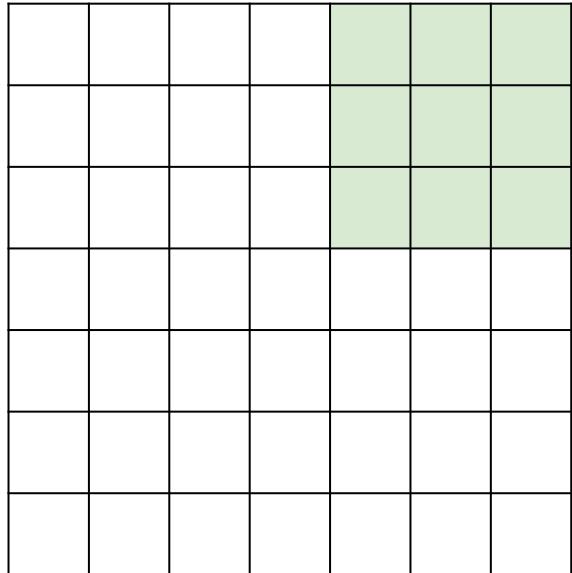


Imagine de  $7 \times 7$  (fără adâncime)  
Filtru de  $3 \times 3$

7

# Dimensiunea spațială a unei activări

7

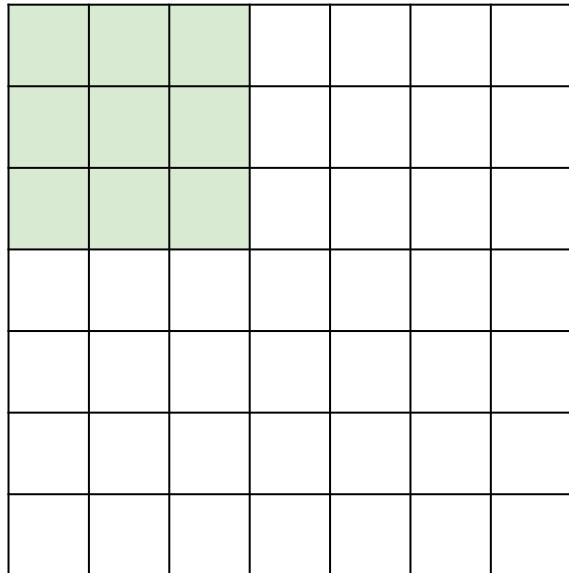


Imagine de  $7 \times 7$  (fără adâncime)  
Filtru de  $3 \times 3$

=> Output de  $5 \times 5$

# Dimensiunea spațială a unei activări

7

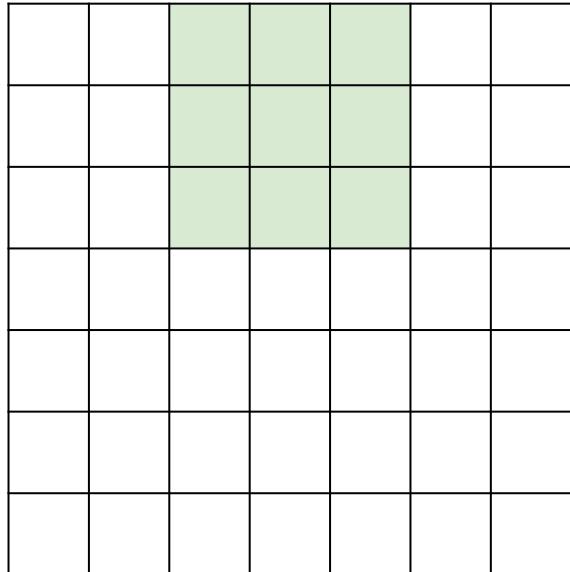


Imagine de  $7 \times 7$  (fără adâncime)  
Filtru de  $3 \times 3$  aplicat cu **stride 2**

7

# Dimensiunea spațială a unei activări

7

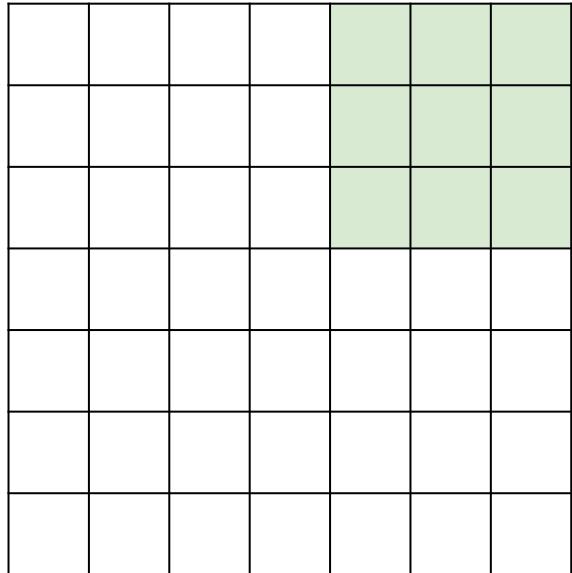


Imagine de  $7 \times 7$  (fără adâncime)  
Filtru de  $3 \times 3$  aplicat cu **stride 2**

7

# Dimensiunea spațială a unei activări

7

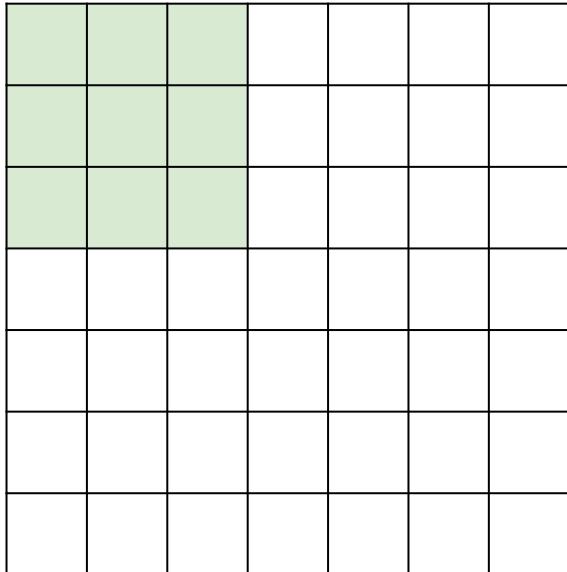


Imagine de  $7 \times 7$  (fără adâncime)  
Filtru de  $3 \times 3$  aplicat cu **stride 2**

=> Output de  $3 \times 3$

# Dimensiunea spațială a unei activări

7



7

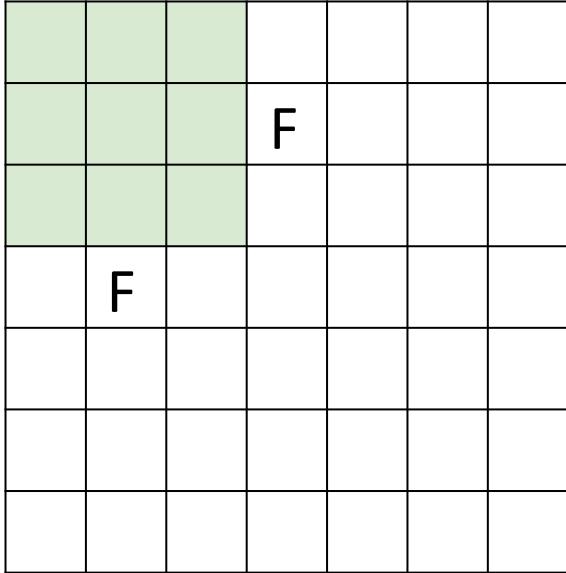
Imagine de  $7 \times 7$  (fără adâncime)  
Filtru de  $3 \times 3$  aplicat cu **stride 3**?

Nu se potrivește!

Nu putem aplica un filtru de  $3 \times 3$   
pe o imagine de  $7 \times 7$  folosind stride  
3

N

---



Mărimea activării:  
 **$(N - F) / \text{stride} + 1$**

N

e.g.  $N = 7, F = 3$ :

$$\text{stride } 1 \Rightarrow (7 - 3)/1 + 1 = 5$$

$$\text{stride } 2 \Rightarrow (7 - 3)/2 + 1 = 3$$

$$\text{stride } 3 \Rightarrow (7 - 3)/3 + 1 = 2.33 :($$

# În practică: deobicei imaginea se bordează cu 0

0	0	0	0	0	0			
0								
0								
0								
0								

e.g. imagine de 7x7

filtru **3x3**, aplicat cu **stride 1**

**bordăm cu 1 pixel** de valoare 0

Q: Cum arată activarea?

=> **Output de 7x7**

În general, se folosesc straturi conveționale cu stride 1, cu filtre de dimensiune  $F \times F$ , și bordură de  $(F-1)/2$ . (menține dimensiunea imaginii de input)

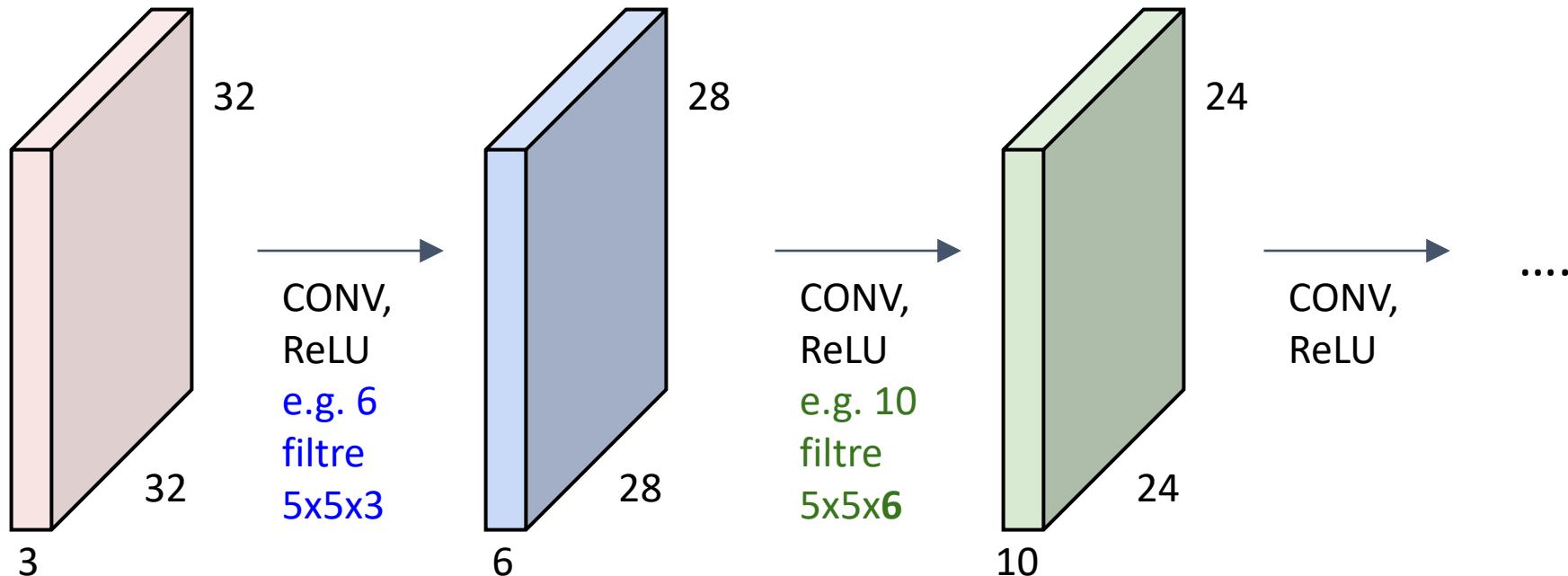
e.g.  $F = 3 \Rightarrow$  bordăm cu 1 pixel (valoare 0)

$F = 5 \Rightarrow$  bordăm cu 2 pixeli (valoare 0)

$F = 7 \Rightarrow$  bordăm cu 3 pixeli (valoare 0)

Aplicând convoluții cu filtre de  $5 \times 5$  în mod repetat pe un input de  $32 \times 32$  micșorează volumul ( $32 \Rightarrow 28 \Rightarrow 24 \dots$ )

Micșorarea prea rapidă a volumului nu produce rezultate optime

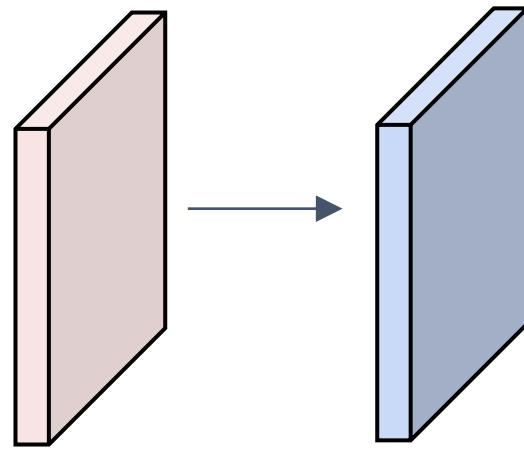


Exemplu:

Volum de input: **32x32x3**

10 filtre de 5x5 cu stride 1, bordură 2

Mărimea volumului de output: ?



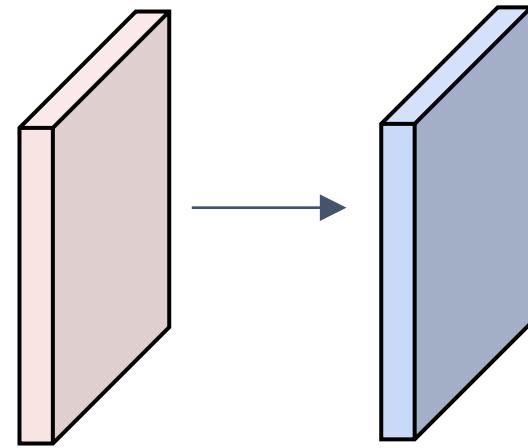
Exemplu:

Volum de input: **32x32x3**

10 filtre de **5x5** cu stride **1**, bordură **2**

Mărimea volumului de output:

$(32+2*2-5)/1+1 = 32$ , deci volumul  
este **32x32x10**

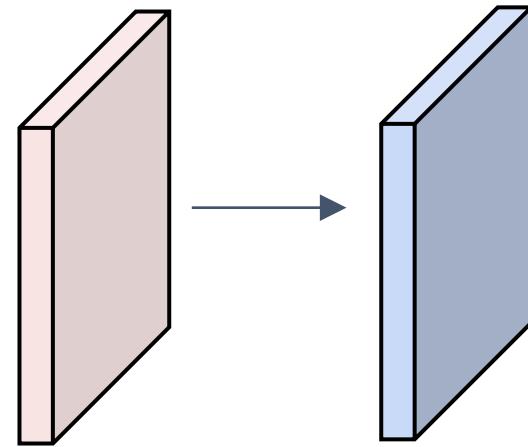


Exemplu:

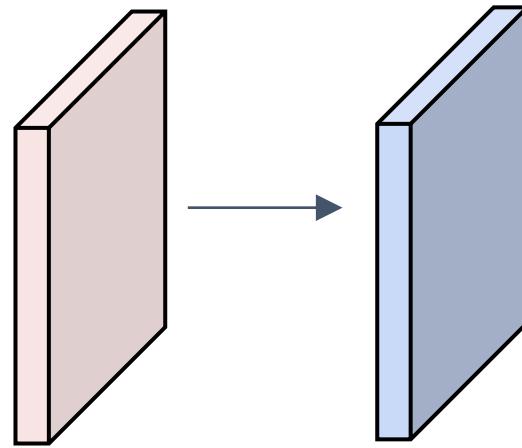
Volum de input: **32x32x3**

10 filtre de 5x5 cu stride 1, bordură 2

Numărul de parametrii al acestui strat: ?



## Exemplu:



Volum de input: **32x32x3**

**10** filtre de **5x5** cu stride 1, bordură 2

Numărul de parametrii al acestui strat: ?

Fiecare filtru are  $5 \times 5 \times 3 + 1 = 76$

**parametrii** (+1 pentru bias)

$$\Rightarrow 76 * 10 = 760$$

## Setări comune:

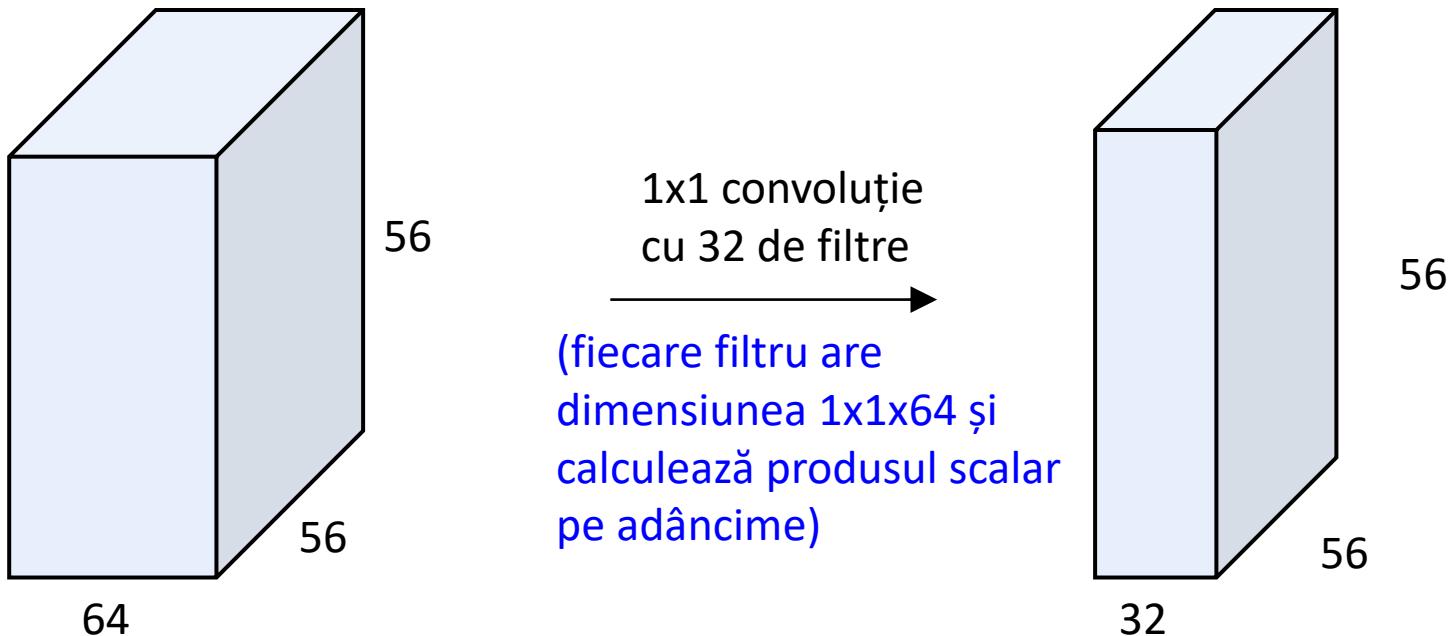
**Summary.** To summarize, the Conv Layer:

- Accepts a volume of size  $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters  $K$ ,
  - their spatial extent  $F$ ,
  - the stride  $S$ ,
  - the amount of zero padding  $P$ .
- Produces a volume of size  $W_2 \times H_2 \times D_2$  where:
  - $W_2 = (W_1 - F + 2P)/S + 1$
  - $H_2 = (H_1 - F + 2P)/S + 1$  (i.e. width and height are computed equally by symmetry)
  - $D_2 = K$
- With parameter sharing, it introduces  $F \cdot F \cdot D_1$  weights per filter, for a total of  $(F \cdot F \cdot D_1) \cdot K$  weights and  $K$  biases.
- In the output volume, the  $d$ -th depth slice (of size  $W_2 \times H_2$ ) is the result of performing a valid convolution of the  $d$ -th filter over the input volume with a stride of  $S$ , and then offset by  $d$ -th bias.

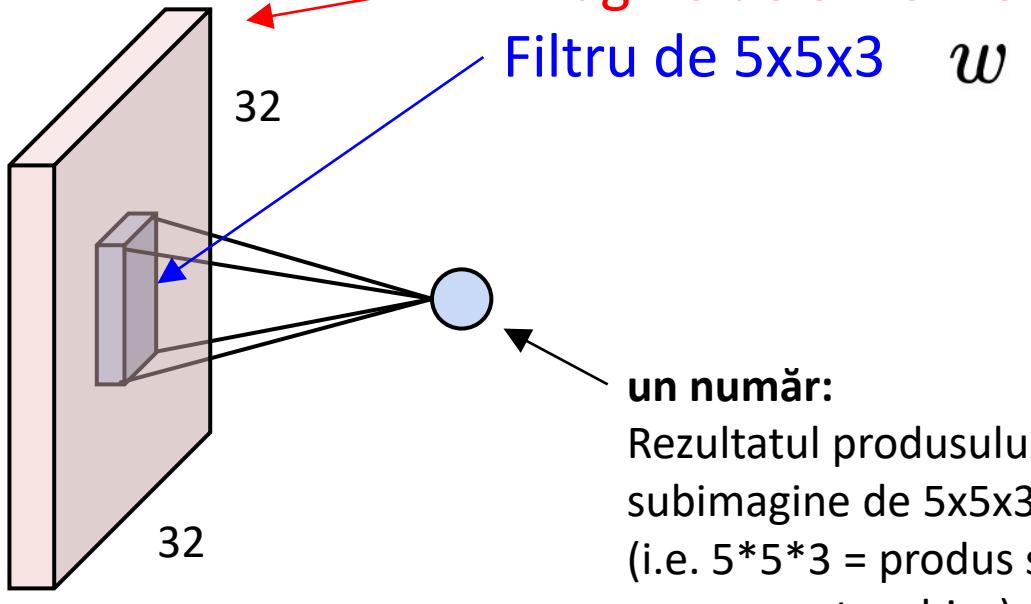
$K = (\text{puteri ale lui } 2, \text{ e.g. } 32, 64, 128, 512)$

- $F = 3, S = 1, P = 1$
- $F = 5, S = 1, P = 2$
- $F = 5, S = 2, P = ?$  (orice se încadrează)
- $F = 1, S = 1, P = 0$

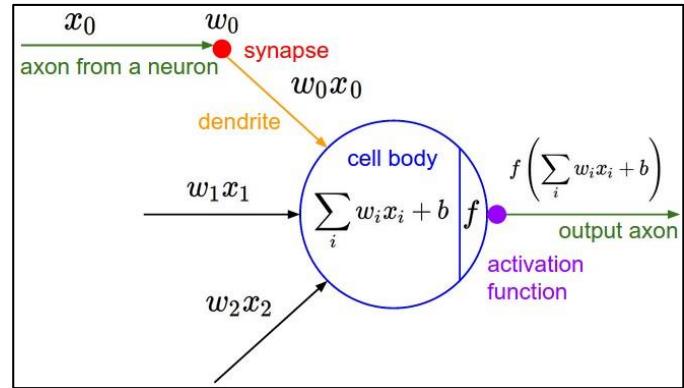
## Are sens să folosim chiar filtre de 1x1



# Stratul convolutional din punct de vedere biologic

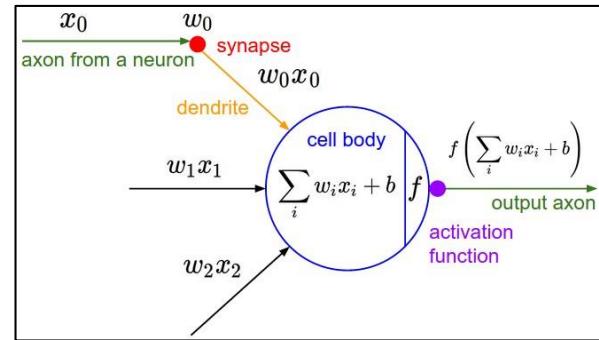
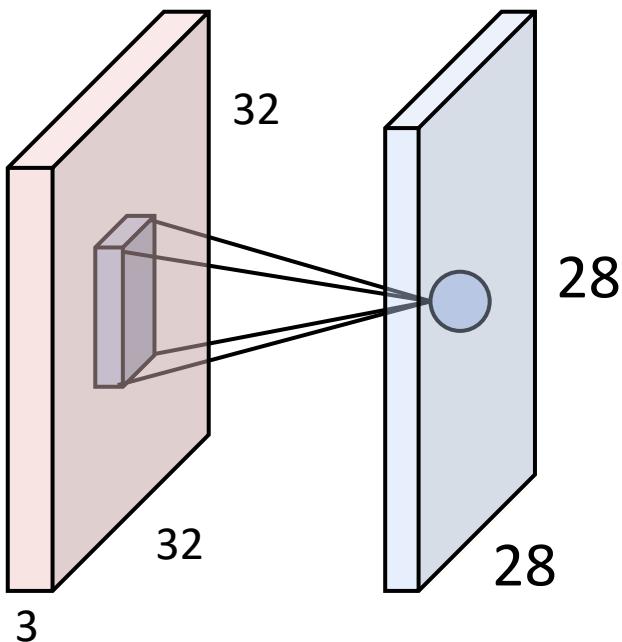


$$w^T x + b$$



...este doar un neuron cu conexiuni locale

# Stratul convolutional din punct de vedere biologic

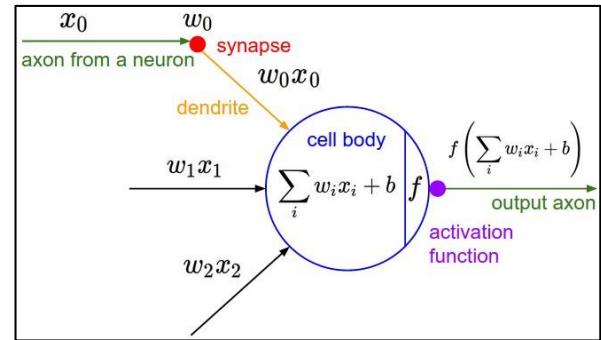
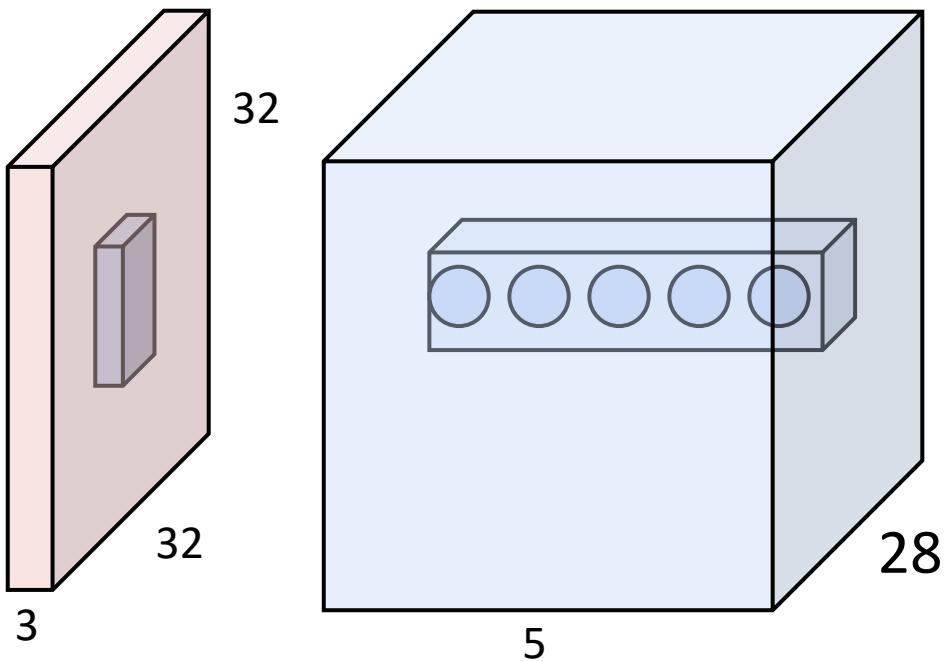


Un activation map este matrice cu output-urile a  $28 \times 28$  neuroni:

1. Fiecare este conectat la o regiune mică din input
2. Toții neuroni au aceeși parametrii

filtru  $5 \times 5$  = câmp receptiv de  $5 \times 5$  pentru fiecare neuron

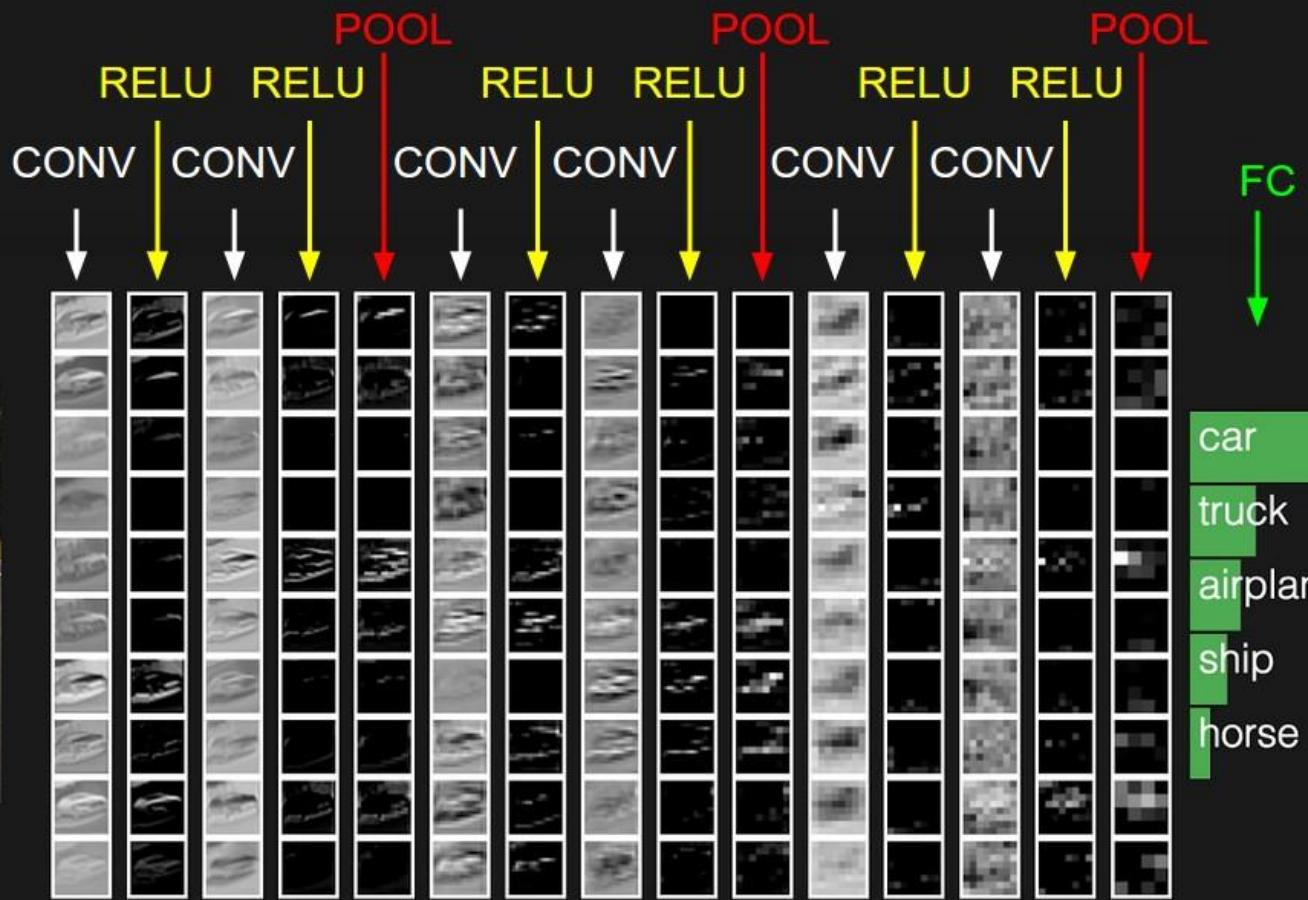
# Stratul convolutional din punct de vedere biologic



E.g. cu 5 filtre,  
stratul convolutional este format din  
neuroni aranjați într-un grid  
( $28 \times 28 \times 5$ )

Pentru o regiune vor fi 5 neuroni  
diferiți (toți primesc același input)

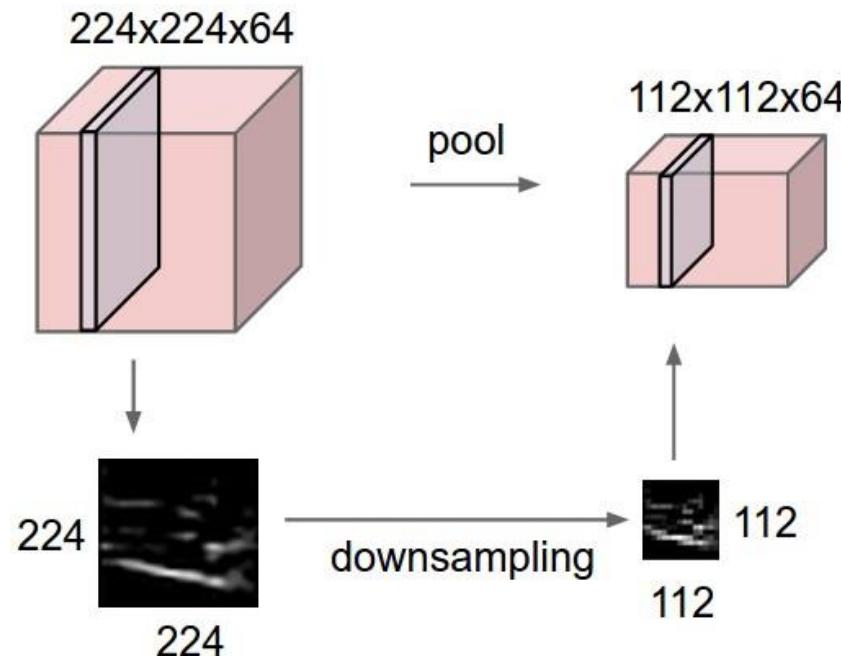
Mai avem de discutat despre două tipuri de straturi: POOL, FC



car  
truck  
airplane  
ship  
horse

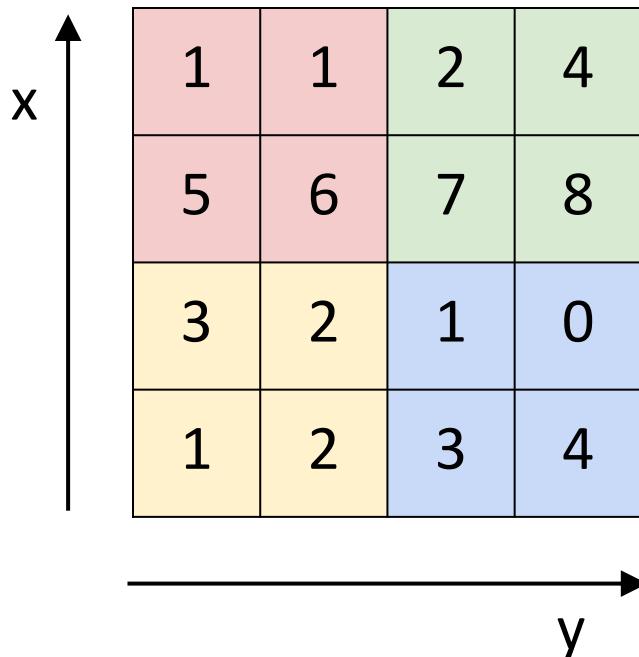
# Stratul de pooling

- Reduce reprezentarea, permite o utilizare mai ușoară
- Operează pe fiecare activation map în parte:



# MAX Pooling

Un activation map



max pooling cu filtru  
de 2x2 și stride 2

The result of the max pooling operation is a 2x2 matrix:

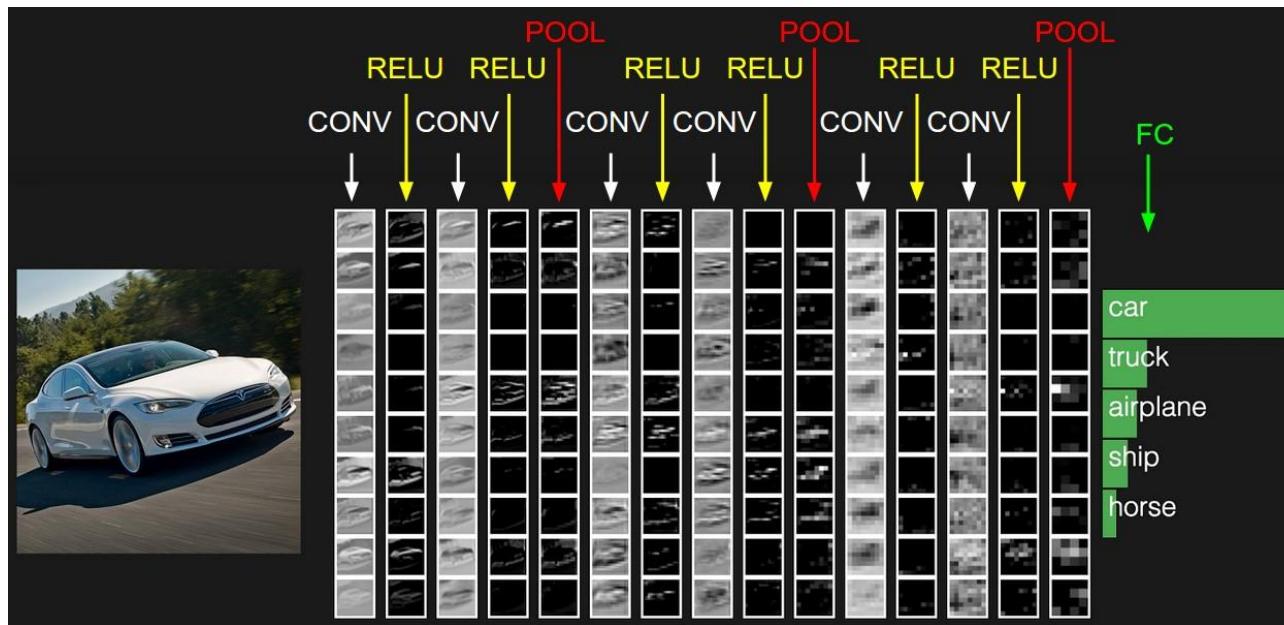
6	8
3	4

Setări comune:

- Accepts a volume of size  $W_1 \times H_1 \times D_1$   $F = 2, S = 2$
- Requires three hyperparameters:
  - their spatial extent  $F$ ,
  - the stride  $S$ ,
- Produces a volume of size  $W_2 \times H_2 \times D_2$  where:
  - $W_2 = (W_1 - F)/S + 1$
  - $H_2 = (H_1 - F)/S + 1$
  - $D_2 = D_1$
- Introduces zero parameters since it computes a fixed function of the input
- Note that it is not common to use zero-padding for Pooling layers

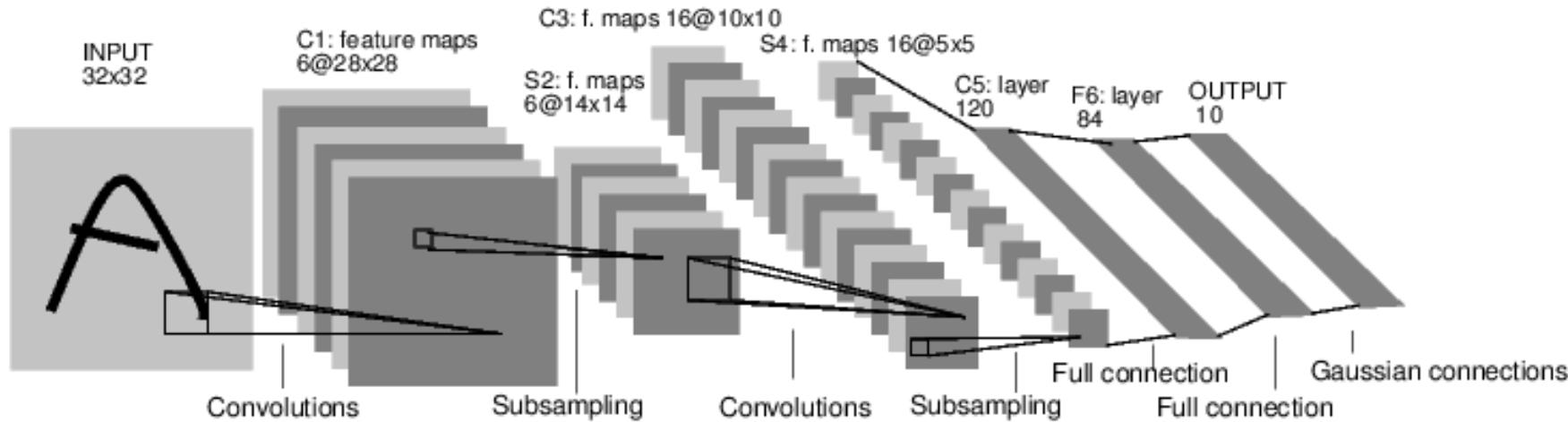
# Straturi cu conexiuni complete (FC layer)

- Conține neuroni conectați cu întregul volum de input, ca în rețelele neuronale obișnuite



# Studiu de caz: LeNet-5

[LeCun et al., 1998]



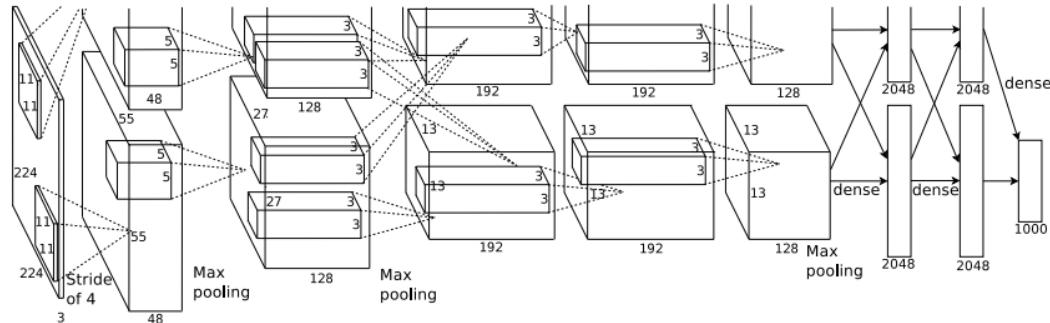
Straturi conveționale cu filtru de 5x5, aplicate cu stride 1

Straturi de pooling cu filtru de 2x2, aplicate cu stride 2

Arhitectura este [CONV-POOL-CONV-POOL-CONV-FC]

# Studiu de caz: AlexNet

[Krizhevsky et al. 2012]



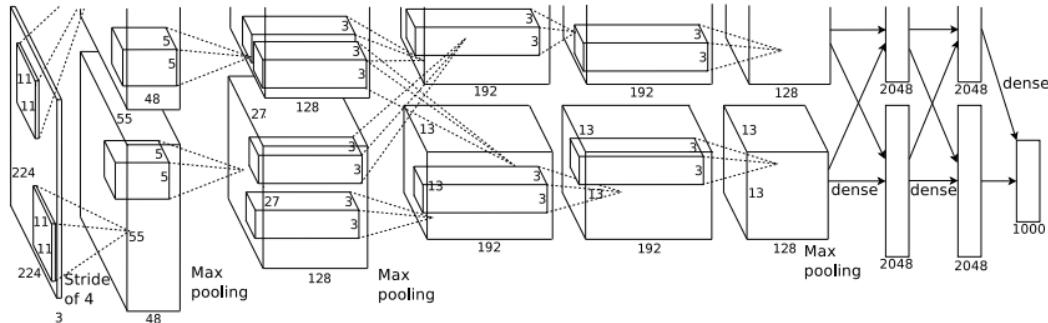
Input: imagini de 227x227x3

**Primul strat (CONV1):** 96 filtre de 11x11, aplicate cu stride 4

Q: Care este mărimea volumului de output? Hint:  $(227-11) / 4 + 1 = 55$

# Studiu de caz: AlexNet

[Krizhevsky et al. 2012]



Input: imagini de 227x227x3

**Primul strat (CONV1):** 96 filtre de 11x11, aplicate cu stride 4

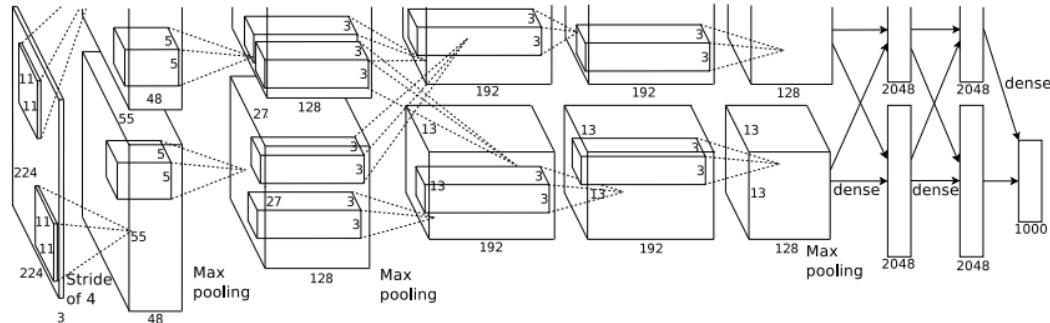
=>

Volumul de output este **[55x55x96]**

Q: Care este numărul parametrilor din acest strat?

# Studiu de caz: AlexNet

[Krizhevsky et al. 2012]



Input: imagini de 227x227x3

**Primul strat (CONV1):** 96 filtre de 11x11, aplicate cu stride 4

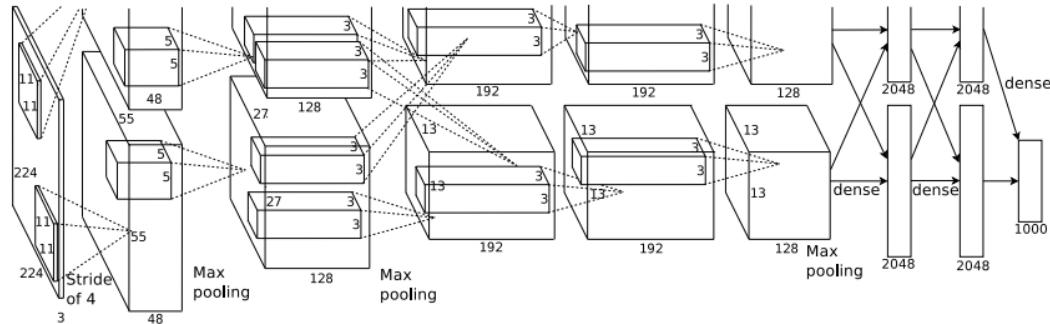
=>

Volumul de output este **[55x55x96]**

Parametrii:  $(11 \times 11 \times 3) \times 96 = 35\text{K}$

# Studiu de caz: AlexNet

[Krizhevsky et al. 2012]



Input: imagini de 227x227x3

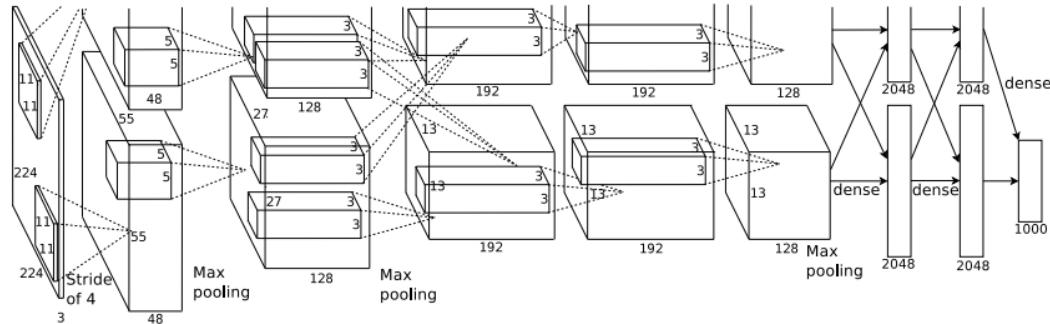
CONV1: 55x55x96

Al doilea strat (POOL1): filtru de 3x3 aplicate cu stride 2

Q: Care este mărimea volumului de output? Hint:  $(55-3) / 2 + 1 = 27$

# Studiu de caz: AlexNet

[Krizhevsky et al. 2012]



Input: imagini de 227x227x3

CONV1: 55x55x96

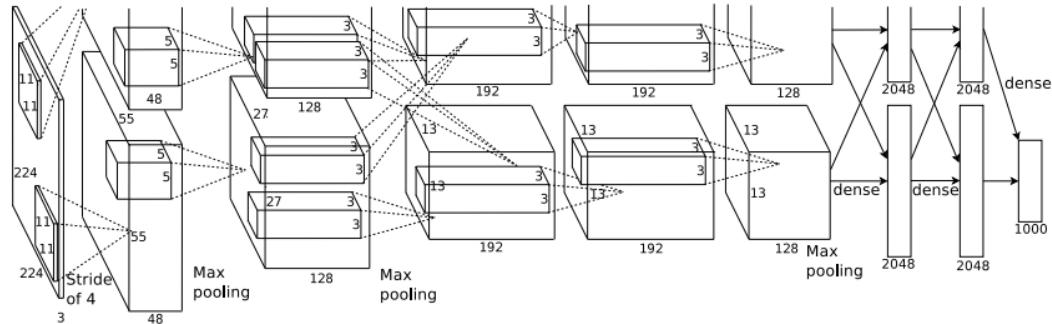
Al doilea strat (POOL1): filtre de 3x3 aplicate cu stride 2

Volumul de output este **[27x27x96]**

Q: Care este numărul parametrilor din acest strat?

# Studiu de caz: AlexNet

[Krizhevsky et al. 2012]



Input: imagini de 227x227x3

CONV1: 55x55x96

Al doilea strat (POOL1): filtru de 3x3 aplicate cu stride 2

Volumul de output este **[27x27x96]**

Parametrii: **0**

# Studiu de caz: AlexNet

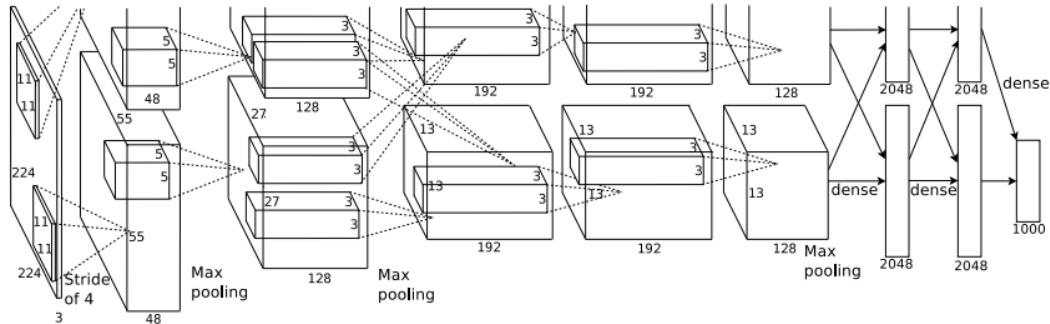
[Krizhevsky et al. 2012]

Input: imagini de 227x227x3

CONV1: 55x55x96

POOL1: 27x27x96

...



# Studiu de caz: AlexNet

[Krizhevsky et al. 2012]

Arhitectura completă AlexNet:

[227x227x3] INPUT

[55x55x96] CONV1: 96 filtre de 11x11 cu stride 4, fără bordură

[27x27x96] MAX POOL1: filtre de 3x3 cu stride 2

[27x27x96] NORM1: strat de normalizare

[27x27x256] CONV2: 256 filtre de 5x5 cu stride 1, bordură 2

[13x13x256] MAX POOL2: filtre de 3x3 cu stride 2

[13x13x256] NORM2: strat de normalizare

[13x13x384] CONV3: 384 filtre de 3x3 cu stride 1, bordură 1

[13x13x384] CONV4: 384 filtre de 3x3 cu stride 1, bordură 1

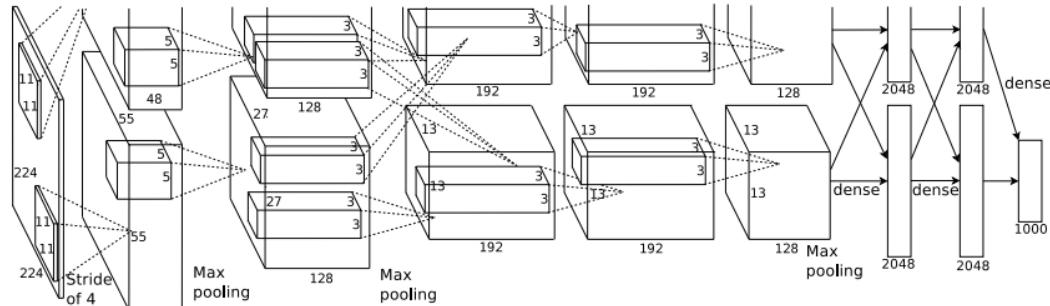
[13x13x256] CONV5: 256 filtre de 3x3 cu stride 1, bordură 1

[6x6x256] MAX POOL3: filtre de 3x3 cu stride 2

[4096] FC6: 4096 neuroni

[4096] FC7: 4096 neuroni

[1000] FC8: 1000 neuroni (scoruri pentru 1000 de clase)



## Detalii:

- Prima utilizare a ReLU
- Straturi de normalizare (nu se mai folosesc)
- Augmentarea datelor
- Dropout 0.5
- Mărimea batch-ului 128
- SGD cu moment 0.9
- Rata de învățare 0.01, împărțită la 10 atunci când acuratețea de validare atinge un platou
- Ansamblu de 7 CNN-uri: 18.2% => 15.4%

# Studiu de caz: VGGNet

[Simonyan and Zisserman, 2014]

Doar CONV de 3x3 cu stride 1, bordură 1  
și MAX POOL de 2x2 cu stride 2

Cel mai bun model

11.2% eroare top 5 la ILSVRC 2013

=>

7.3% eroare top 5

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 LRN	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 conv3-256 conv3-256
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 2: Number of parameters (in millions).

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

(fără a număra bias-urile)

INPUT: [224x224x3] memoria:  $224 \times 224 \times 3 = 150K$  parametrii: 0

CONV3-64: [224x224x64] memoria:  $224 \times 224 \times 64 = 3.2M$  parametrii:  $(3 \times 3 \times 3) \times 64 = 1,728$

CONV3-64: [224x224x64] memoria:  $224 \times 224 \times 64 = 3.2M$  parametrii:  $(3 \times 3 \times 64) \times 64 = 36,864$

POOL2: [112x112x64] memoria:  $112 \times 112 \times 64 = 800K$  parametrii: 0

CONV3-128: [112x112x128] memoria:  $112 \times 112 \times 128 = 1.6M$  parametrii:  $(3 \times 3 \times 64) \times 128 = 73,728$

CONV3-128: [112x112x128] memoria:  $112 \times 112 \times 128 = 1.6M$  parametrii:  $(3 \times 3 \times 128) \times 128 = 147,456$

POOL2: [56x56x128] memoria:  $56 \times 56 \times 128 = 400K$  parametrii: 0

CONV3-256: [56x56x256] memoria:  $56 \times 56 \times 256 = 800K$  parametrii:  $(3 \times 3 \times 128) \times 256 = 294,912$

CONV3-256: [56x56x256] memoria:  $56 \times 56 \times 256 = 800K$  parametrii:  $(3 \times 3 \times 256) \times 256 = 589,824$

CONV3-256: [56x56x256] memoria:  $56 \times 56 \times 256 = 800K$  parametrii:  $(3 \times 3 \times 256) \times 256 = 589,824$

POOL2: [28x28x256] memoria:  $28 \times 28 \times 256 = 200K$  parametrii: 0

CONV3-512: [28x28x512] memoria:  $28 \times 28 \times 512 = 400K$  parametrii:  $(3 \times 3 \times 256) \times 512 = 1,179,648$

CONV3-512: [28x28x512] memoria:  $28 \times 28 \times 512 = 400K$  parametrii:  $(3 \times 3 \times 512) \times 512 = 2,359,296$

CONV3-512: [28x28x512] memoria:  $28 \times 28 \times 512 = 400K$  parametrii:  $(3 \times 3 \times 512) \times 512 = 2,359,296$

POOL2: [14x14x512] memoria:  $14 \times 14 \times 512 = 100K$  parametrii: 0

CONV3-512: [14x14x512] memoria:  $14 \times 14 \times 512 = 100K$  parametrii:  $(3 \times 3 \times 512) \times 512 = 2,359,296$

CONV3-512: [14x14x512] memoria:  $14 \times 14 \times 512 = 100K$  parametrii:  $(3 \times 3 \times 512) \times 512 = 2,359,296$

CONV3-512: [14x14x512] memoria:  $14 \times 14 \times 512 = 100K$  parametrii:  $(3 \times 3 \times 512) \times 512 = 2,359,296$

POOL2: [7x7x512] memoria:  $7 \times 7 \times 512 = 25K$  parametrii: 0

FC: [1x1x4096] memoria: 4096 parametrii:  $7 \times 7 \times 512 \times 4096 = 102,760,448$

FC: [1x1x4096] memoria: 4096 parametrii:  $4096 \times 4096 = 16,777,216$

FC: [1x1x1000] memoria: 1000 parametrii:  $4096 \times 1000 = 4,096,000$

**Memorie totală: 24M \* 4 bytes ≈ 93MB / imagine** (doar propagarea înainte, ~x2 înapoi)

**Numărul total de parametrii: 138M**

ConvNet Configuration			
B	C	D	E
13 weight layers	16 weight layers	16 weight layers	19 weight layers
put (224 × 224 RGB image)			
conv3-64	conv3-64	conv3-64	conv3-64
<b>conv3-64</b>	conv3-64	conv3-64	conv3-64
maxpool			
conv3-128	conv3-128	conv3-128	conv3-128
<b>conv3-128</b>	conv3-128	conv3-128	conv3-128
maxpool			
conv3-256	conv3-256	conv3-256	conv3-256
conv3-256	conv3-256	conv3-256	conv3-256
	<b>conv1-256</b>	<b>conv3-256</b>	conv3-256
maxpool			
conv3-512	conv3-512	conv3-512	conv3-512
conv3-512	conv3-512	conv3-512	conv3-512
	<b>conv1-512</b>	<b>conv3-512</b>	conv3-512
maxpool			
conv3-512	conv3-512	conv3-512	conv3-512
conv3-512	conv3-512	conv3-512	conv3-512
	<b>conv1-512</b>	<b>conv3-512</b>	conv3-512
maxpool			
FC-4096	FC-4096	FC-4096	FC-4096
FC-4096	FC-4096	FC-4096	FC-4096
FC-1000	FC-1000	FC-1000	FC-1000
soft-max			

INPUT: [224x224x3] memoria:  $224 \times 224 \times 3 = 150\text{K}$  parametrii: 0

CONV3-64: [224x224x64] memoria:  $224 \times 224 \times 64 = 3.2\text{M}$  parametrii:  $(3 \times 3 \times 3) \times 64 = 1,728$

CONV3-64: [224x224x64] memoria:  $224 \times 224 \times 64 = 3.2\text{M}$  parametrii:  $(3 \times 3 \times 64) \times 64 = 36,864$

POOL2: [112x112x64] memoria:  $112 \times 112 \times 64 = 800\text{K}$  parametrii: 0

CONV3-128: [112x112x128] memoria:  $112 \times 112 \times 128 = 1.6\text{M}$  parametrii:  $(3 \times 3 \times 64) \times 128 = 73,728$

CONV3-128: [112x112x128] memoria:  $112 \times 112 \times 128 = 1.6\text{M}$  parametrii:  $(3 \times 3 \times 128) \times 128 = 147,456$

POOL2: [56x56x128] memoria:  $56 \times 56 \times 128 = 400\text{K}$  parametrii: 0

CONV3-256: [56x56x256] memoria:  $56 \times 56 \times 256 = 800\text{K}$  parametrii:  $(3 \times 3 \times 128) \times 256 = 294,912$

CONV3-256: [56x56x256] memoria:  $56 \times 56 \times 256 = 800\text{K}$  parametrii:  $(3 \times 3 \times 256) \times 256 = 589,824$

CONV3-256: [56x56x256] memoria:  $56 \times 56 \times 256 = 800\text{K}$  parametrii:  $(3 \times 3 \times 256) \times 256 = 589,824$

POOL2: [28x28x256] memoria:  $28 \times 28 \times 256 = 200\text{K}$  parametrii: 0

CONV3-512: [28x28x512] memoria:  $28 \times 28 \times 512 = 400\text{K}$  parametrii:  $(3 \times 3 \times 256) \times 512 = 1,179,648$

CONV3-512: [28x28x512] memoria:  $28 \times 28 \times 512 = 400\text{K}$  parametrii:  $(3 \times 3 \times 512) \times 512 = 2,359,296$

CONV3-512: [28x28x512] memoria:  $28 \times 28 \times 512 = 400\text{K}$  parametrii:  $(3 \times 3 \times 512) \times 512 = 2,359,296$

POOL2: [14x14x512] memoria:  $14 \times 14 \times 512 = 100\text{K}$  parametrii: 0

CONV3-512: [14x14x512] memoria:  $14 \times 14 \times 512 = 100\text{K}$  parametrii:  $(3 \times 3 \times 512) \times 512 = 2,359,296$

CONV3-512: [14x14x512] memoria:  $14 \times 14 \times 512 = 100\text{K}$  parametrii:  $(3 \times 3 \times 512) \times 512 = 2,359,296$

CONV3-512: [14x14x512] memoria:  $14 \times 14 \times 512 = 100\text{K}$  parametrii:  $(3 \times 3 \times 512) \times 512 = 2,359,296$

POOL2: [7x7x512] memoria:  $7 \times 7 \times 512 = 25\text{K}$  parametrii: 0

FC: [1x1x4096] memoria: 4096 parametrii:  $7 \times 7 \times 512 \times 4096 = 102,760,448$

FC: [1x1x4096] memoria: 4096 parametrii:  $4096 \times 4096 = 16,777,216$

FC: [1x1x1000] memoria: 1000 parametrii:  $4096 \times 1000 = 4,096,000$

**Memorie totală: 24M \* 4 bytes ~ 93MB / imagine** (doar propagarea înainte, ~x2 înapoi)

**Numărul total de parametrii: 138M**

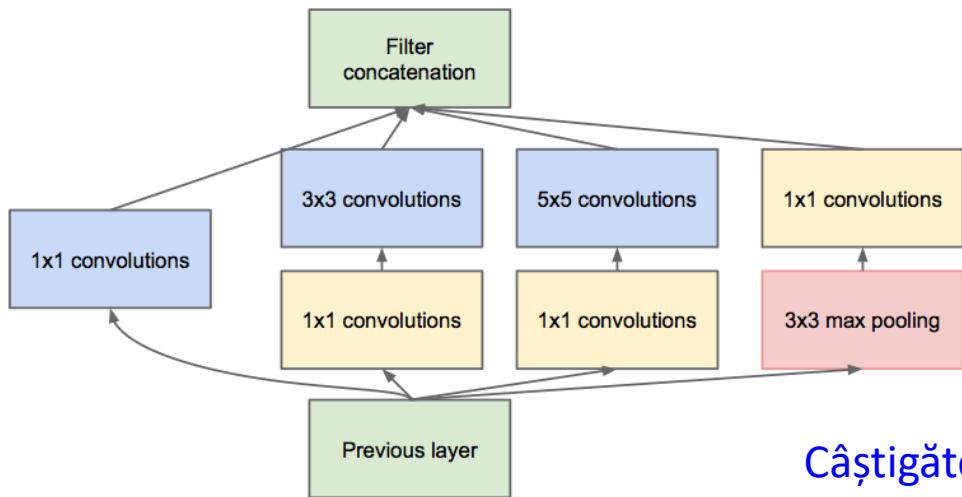
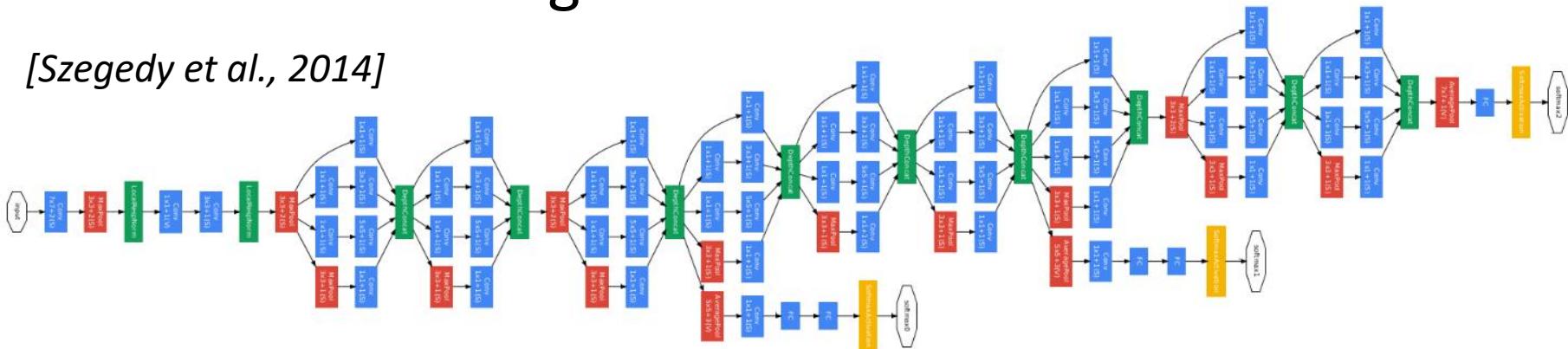
Observații:

Cea mai multă memoria este consumată în primele straturi CONV

Cei mai mulți parametrii în ultimele straturi FC

# Studiu de caz: GoogLeNet

[Szegedy et al., 2014]



Modulul Inception

Câștigătorul ILSVRC 2014 (6.7% eroarea top 5)

# Studiu de caz: GoogLeNet

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

Avantaje interesante:

- Doar 5 milioane de parametrii! (elimină straturile FC)

Comparat cu AlexNet:

- 12x mai puțini parametrii
- 2x mai multe calcule
- 6.67% (vs. 16.4%)

# Studiu de caz: ResNet

[He et al., 2015]

Câștigătorul ILSVRC 2015 (3.6% eroare top 5)



## MSRA @ ILSVRC & COCO 2015 Competitions

- **1st places in all five main tracks**

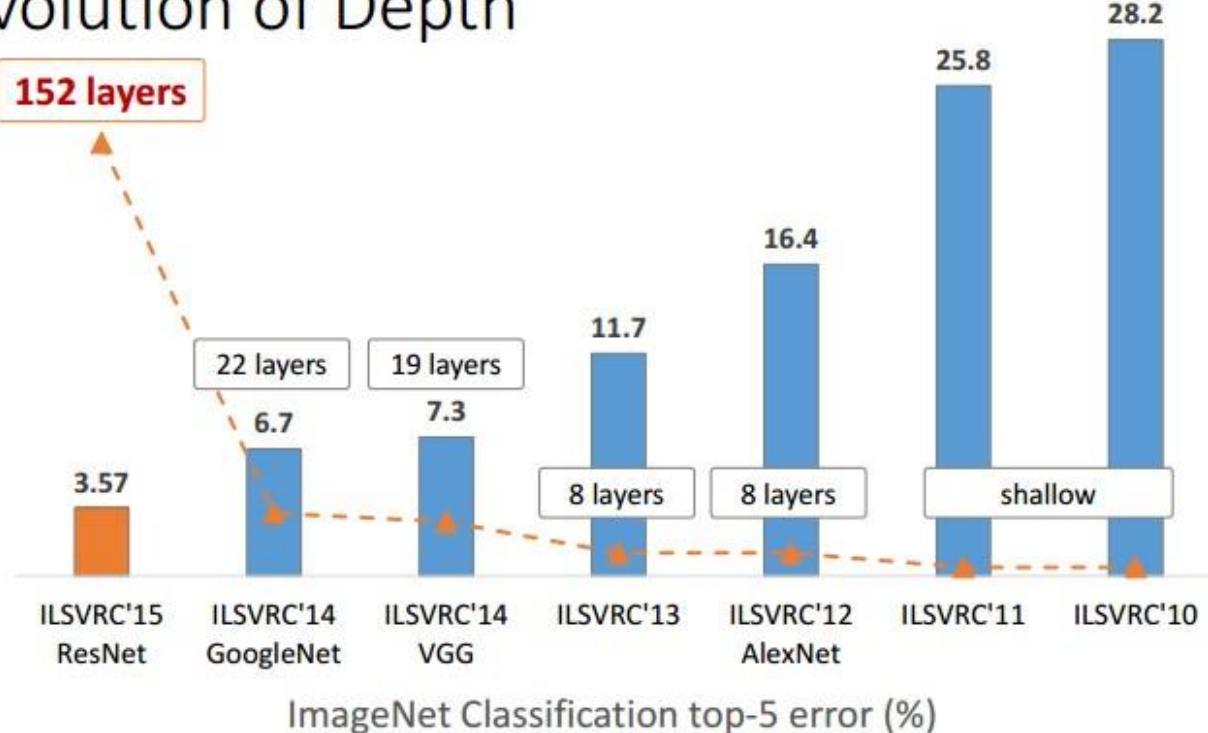
- ImageNet Classification: “Ultra-deep” (quote Yann) **152-layer nets**
- ImageNet Detection: **16%** better than 2nd
- ImageNet Localization: **27%** better than 2nd
- COCO Detection: **11%** better than 2nd
- COCO Segmentation: **12%** better than 2nd

\*improvements are relative numbers

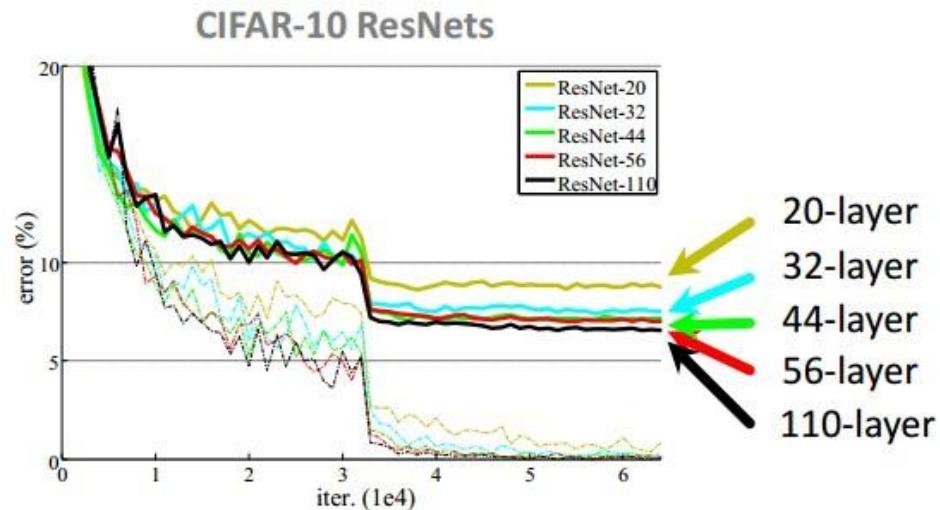
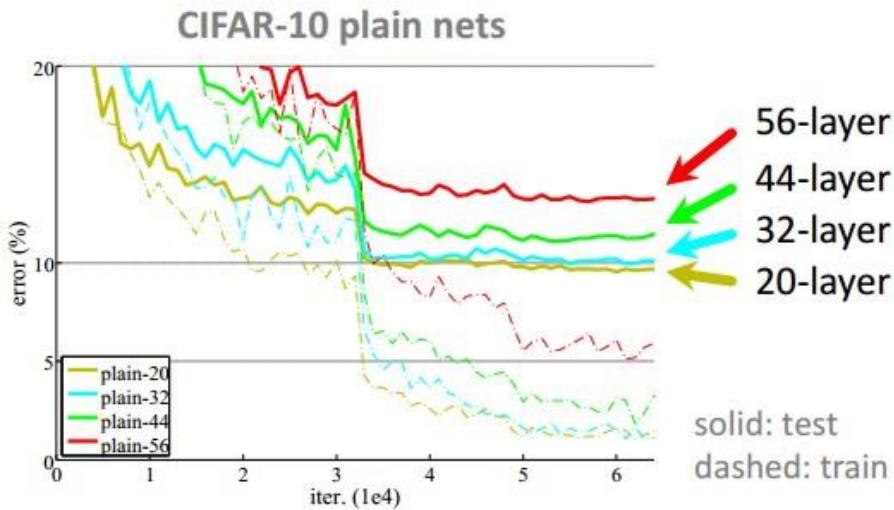


Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". arXiv 2015.

# Revolution of Depth



# CIFAR-10 experiments



# Studiu de caz: ResNet

[He et al., 2015]

Câștigătorul ILSVRC 2015 (3.6% eroare top 5)

## Revolution of Depth

AlexNet, 8 layers  
(ILSVRC 2012)



VGG, 19 layers  
(ILSVRC 2014)



ResNet, 152 layers  
(ILSVRC 2015)

Microsoft  
Research

2-3 săptămâni pentru antrenare pe un server cu 8 plăci GPU

la runtime: mai rapid ca VGGNet! (deși are de 8x mai multe straturi)



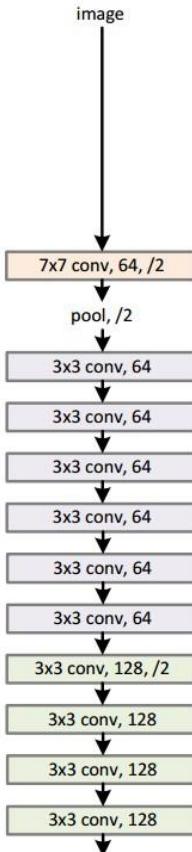
Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun, "Deep Residual Learning for Image Recognition". arXiv 2015.

Slide după Kaiming He

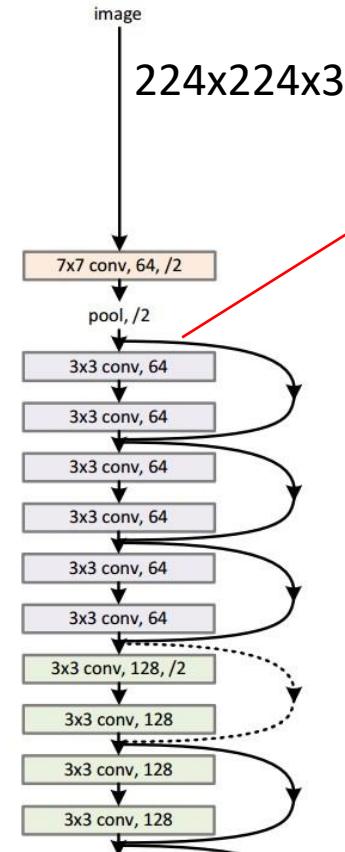
# Studiu de caz: ResNet

[He et al., 2015]

34-layer plain



34-layer residual

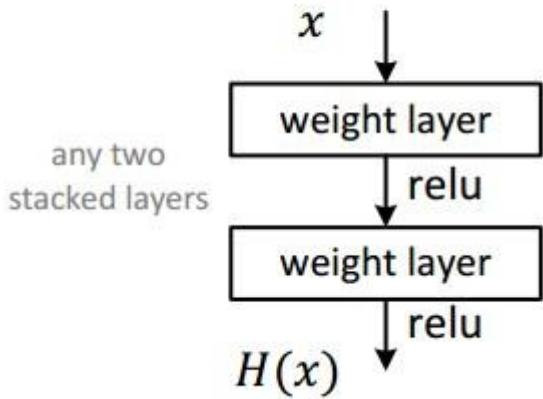


Dimensiunea spațială  
de 56x56!

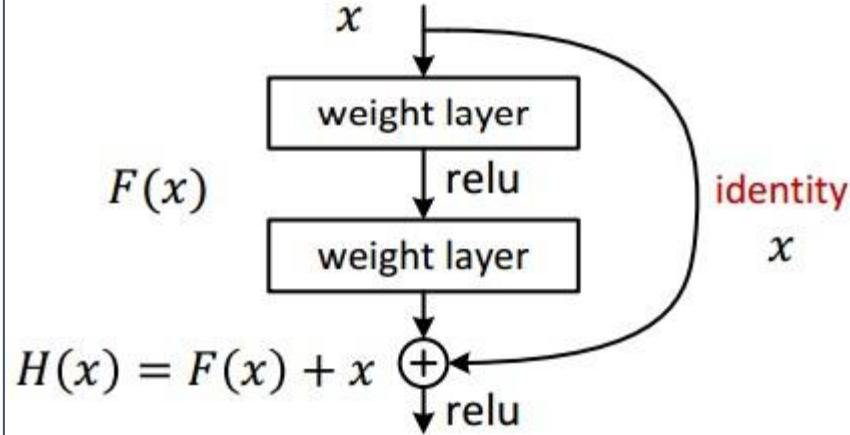
# Studiu de caz: ResNet

[He et al., 2015]

- Plain net



- Residual net



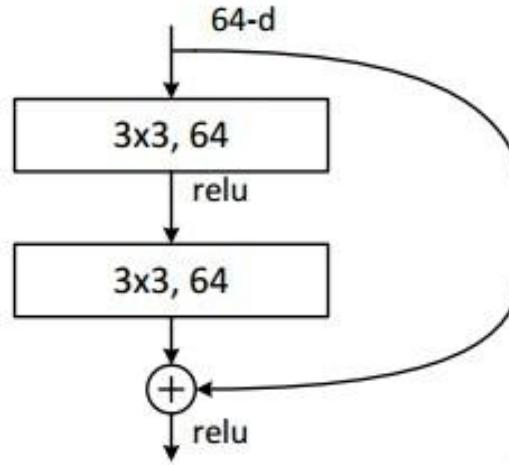
# Studiu de caz: ResNet

[He et al., 2015]

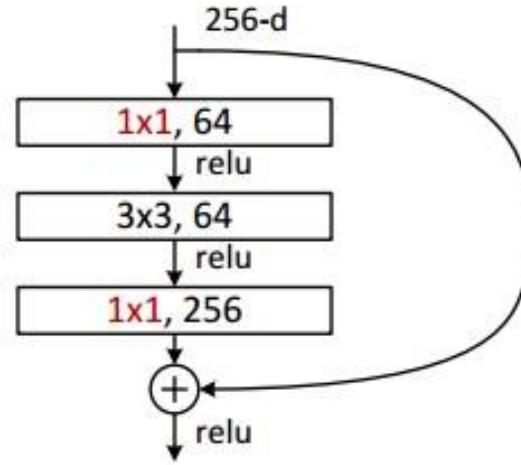
- Normalizare batch după fiecare strat CONV
- Inițializare Xavier / 2
- SGD cu moment 0.9
- Rată de învățare: 0.1, împărțită la 10 când eroare pe validare atinge un platou
- Mărimea unui mini-batch 256
- Degradarea ponderilor cu  $10^{-5}$
- Fără dropout!

# Studiu de caz: ResNet

[He et al., 2015]



all-3x3

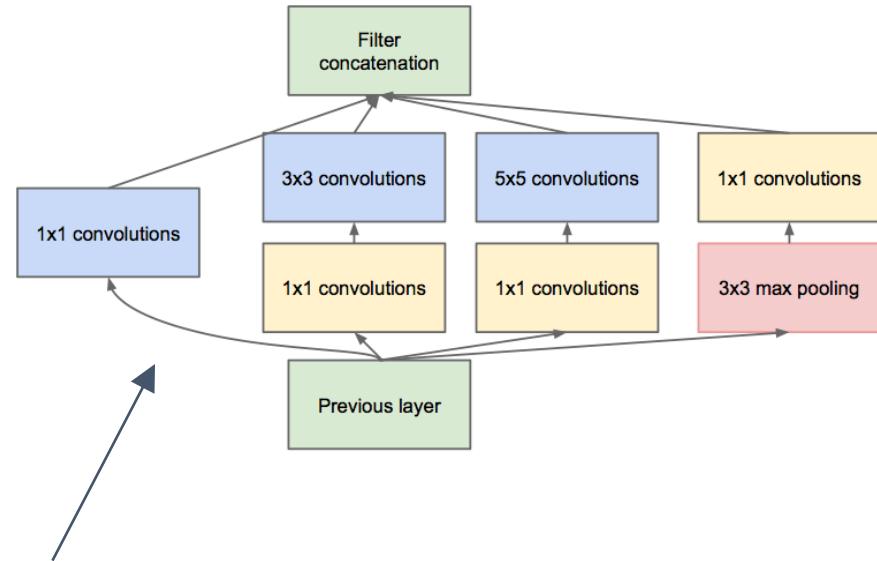
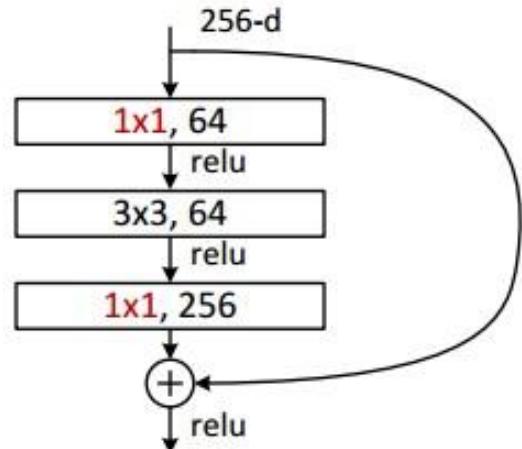


bottleneck  
(for ResNet-50/101/152)



# Studiu de caz: ResNet

[He et al., 2015]

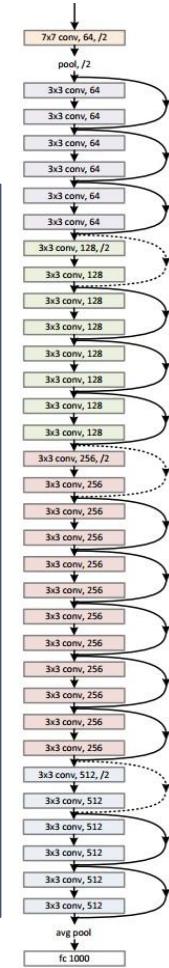


(truc utilizat și în GoogLeNet)

## Studiu de caz: ResNet

[He et al., 2015]

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112			7×7, 64, stride 2		
conv2_x	56×56			3×3 max pool, stride 2		
		$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1			average pool, 1000-d fc, softmax		
FLOPs		$1.8 \times 10^9$	$3.6 \times 10^9$	$3.8 \times 10^9$	$7.6 \times 10^9$	$11.3 \times 10^9$



# Concluzii

- ConvNets: folosim straturi CONV, POOL, FC
- Tendință către filtre mai mici și arhitecturi mai adânci
- Tendință către eliminarea straturilor POOL/FC (doar CONV)
- Arhitectura tipică arată astfel:  
 **$[(CONV-RELU)*N-POOL?] * M - (FC-RELU)*K, SOFTMAX$**   
unde N este deobicei în jur de ~5, M este mare,  $0 \leq K \leq 2$
- Arhitecturile recente (ResNet / GoogLeNet) pun sub semnul întrebării această paradigmă