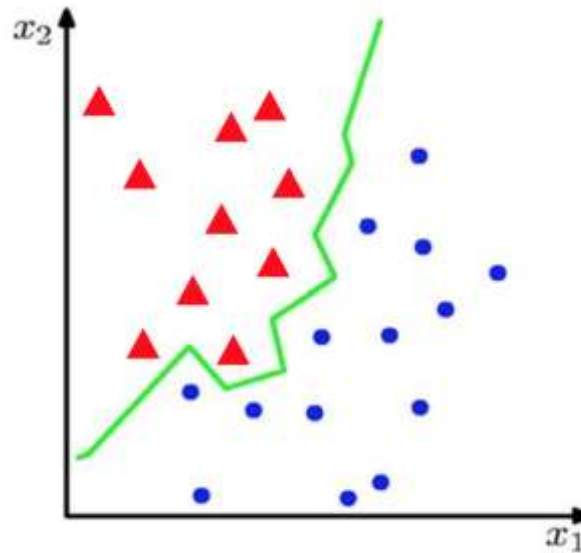


Metoda celor mai apropiați vecini. “Blestemul dimensionalității”.

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

Facultatea de Matematică și Informatică
Universitatea din București

Clasificare din exemple etichetate



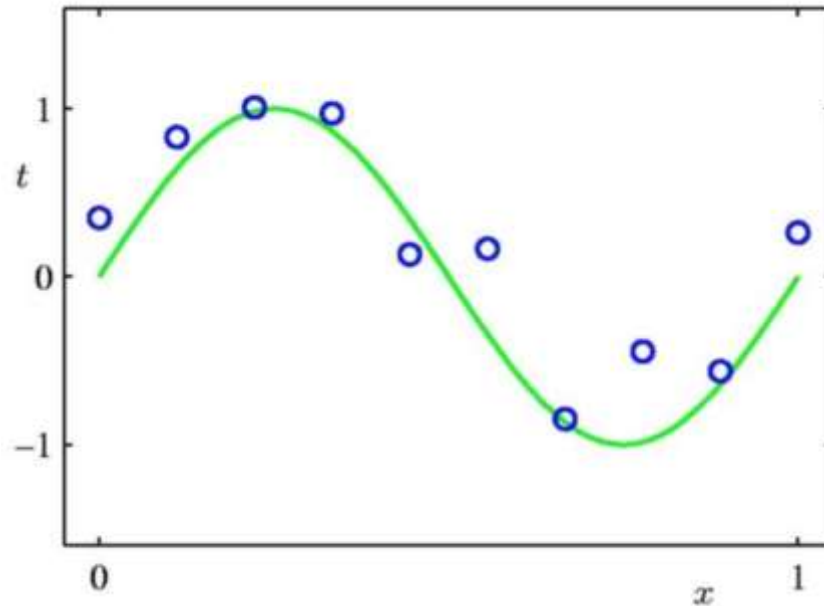
- Presupunem că avem un set de N exemple de antrenare:

$$(x_1, \dots, x_N) \text{ and } (y_1, \dots, y_N), x_i \in \mathbb{R}^d, y_i \in \{-1, 1\}$$

- 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:

$$(x_1, \dots, x_N) \text{ and } (y_1, \dots, y_N), x_i, y_i \in \mathbb{R}$$

- 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):

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

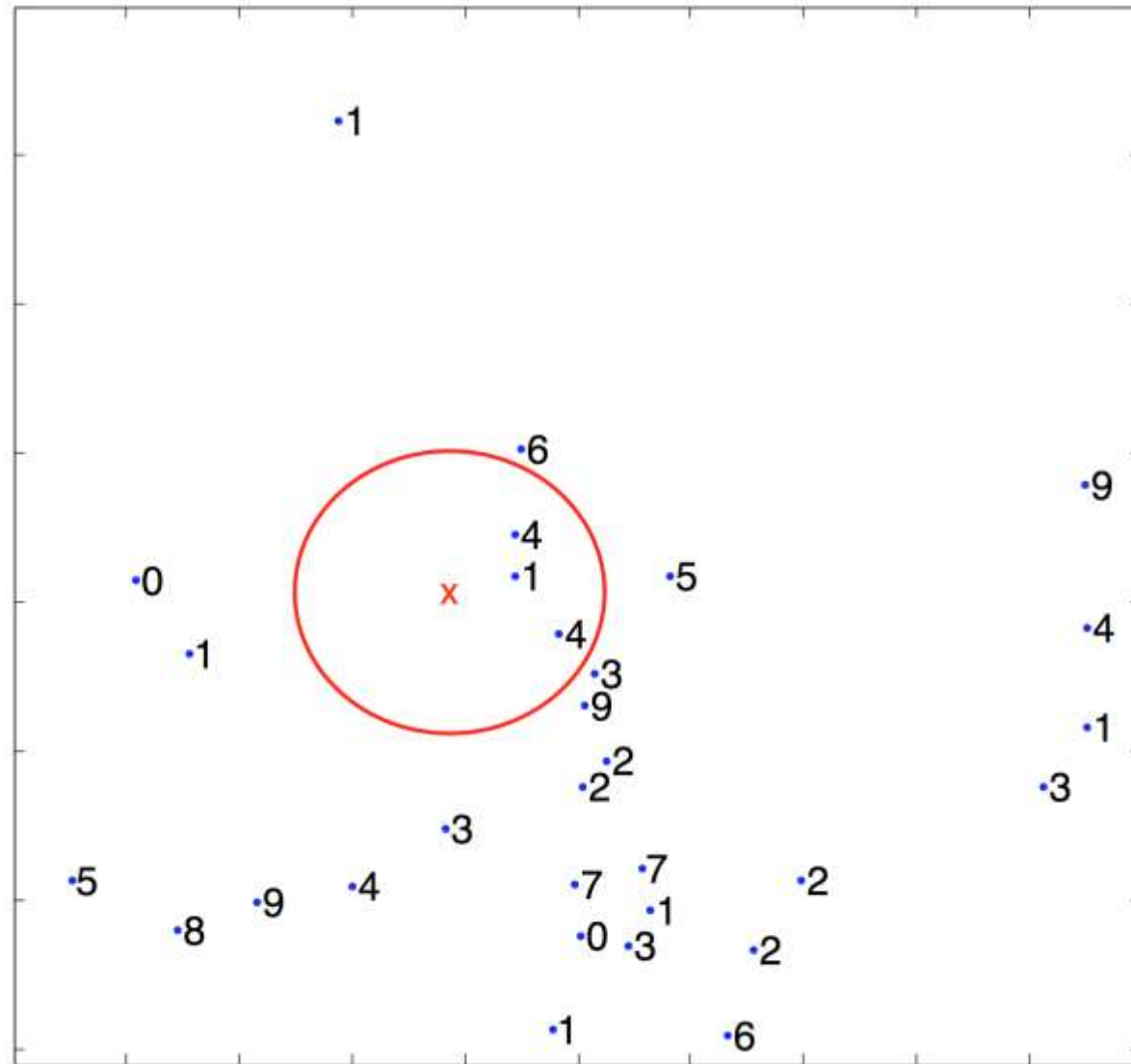
- Eroarea de generalizare:

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

- Eroarea empirică (estimată):

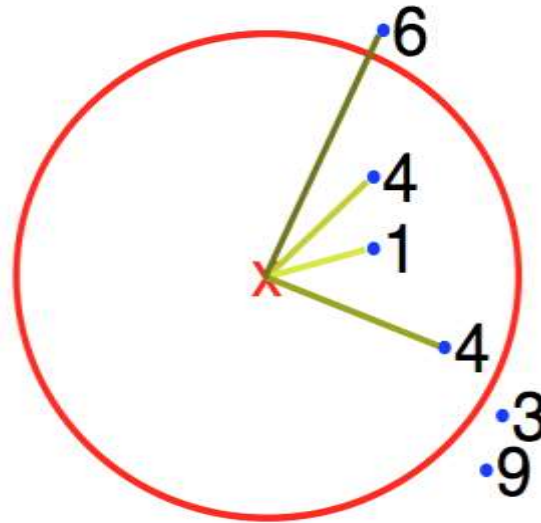
$$L_e(g) = \frac{1}{\ell} \sum_{i=1}^{\ell} \mathcal{L}(y_i, g(\mathbf{x}_i))$$

Care este eticheta exemplului de test x ?



Metoda celor mai apropiați vecini

Metoda celor mai apropiați vecini

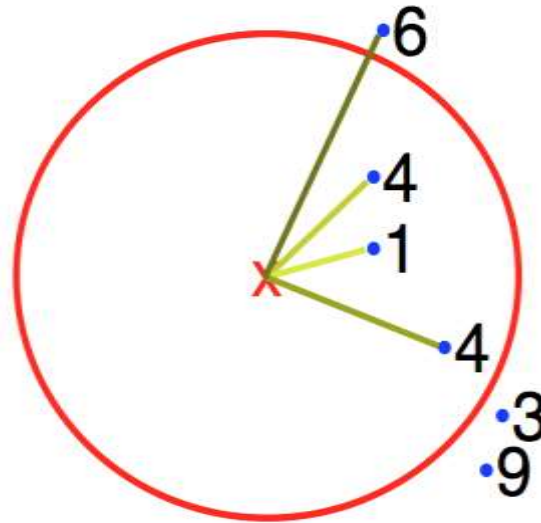


- Algoritmul k-NN:

- 1) Pentru fiecare exemplu de test x , găsim cei mai apropiați k vecini

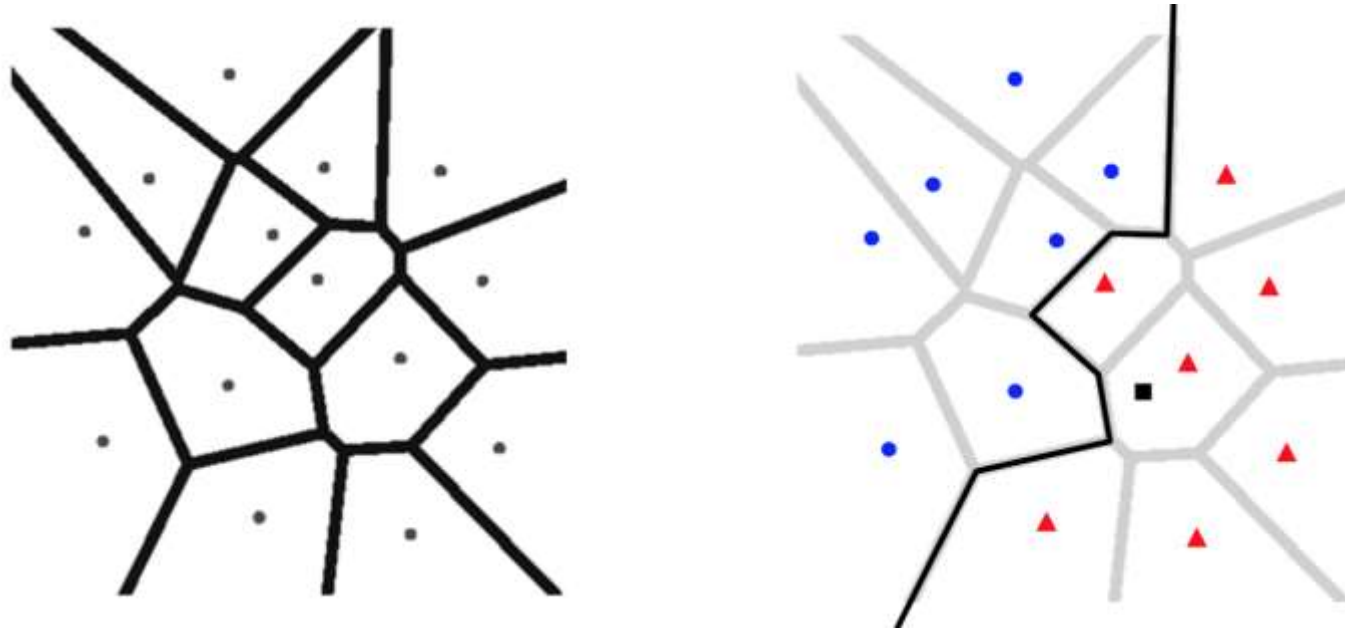
- 2) Atribuim eticheta majoritară conform celor k vecini

k-Nearest Neighbors (k-NN)



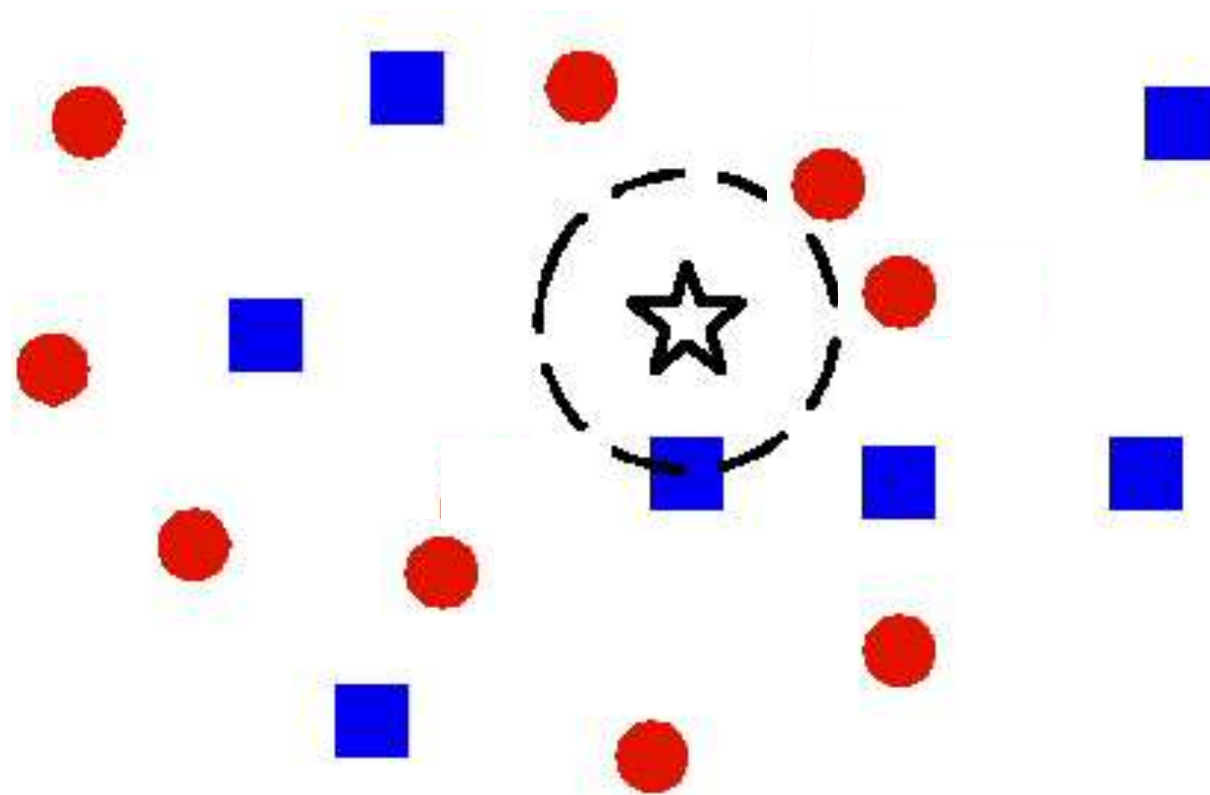
- Cum luăm o decizie în caz de egalitate?
 - 1) Alegem o etichetă din cele egale în mod aleator
 - 2) Aplicăm modelul 1-NN (nu există egalități)
 - 3) Utilizăm distanțele până la exemplul de test ca ponderi

Ce se întâmplă în cazul $k = 1$?

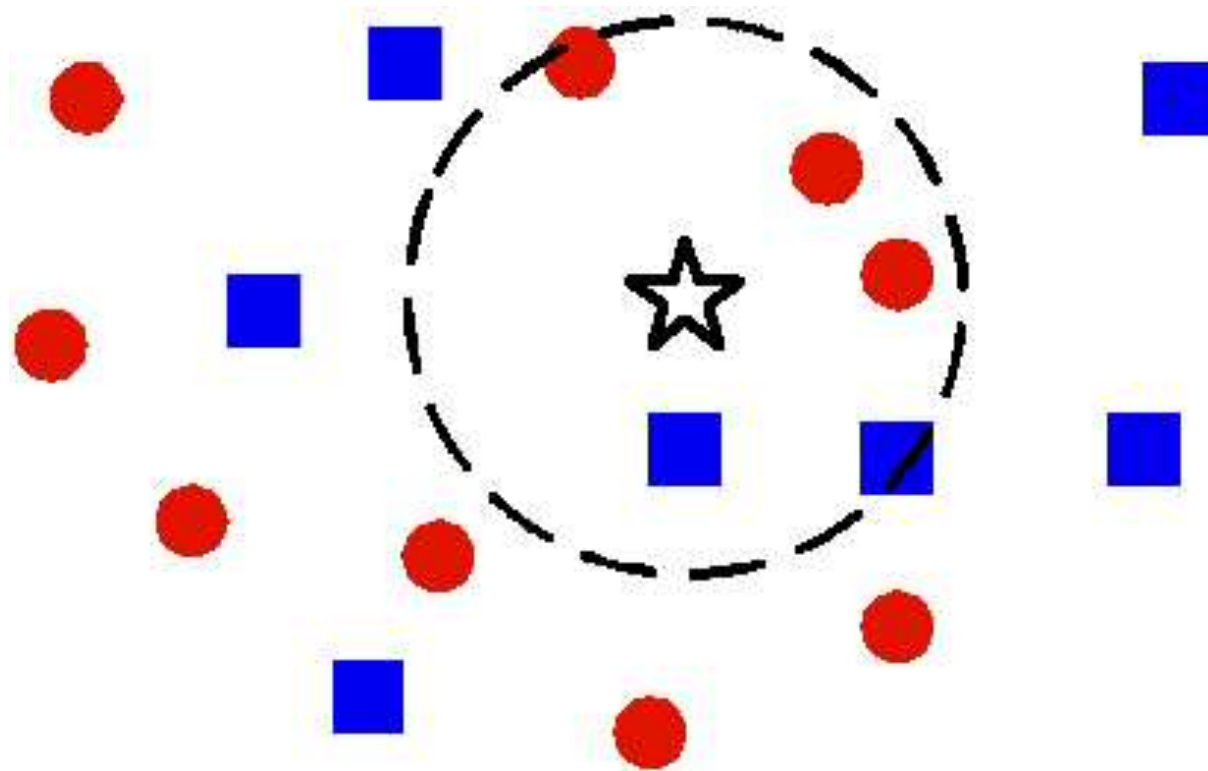


- Obținem o diagramă Voronoi:
 - spațiul este partiționat în regiuni
 - granițele de separare trec prin zonele în care distanțele între exemplele de training sunt egale
- Granița de separare este neliniară

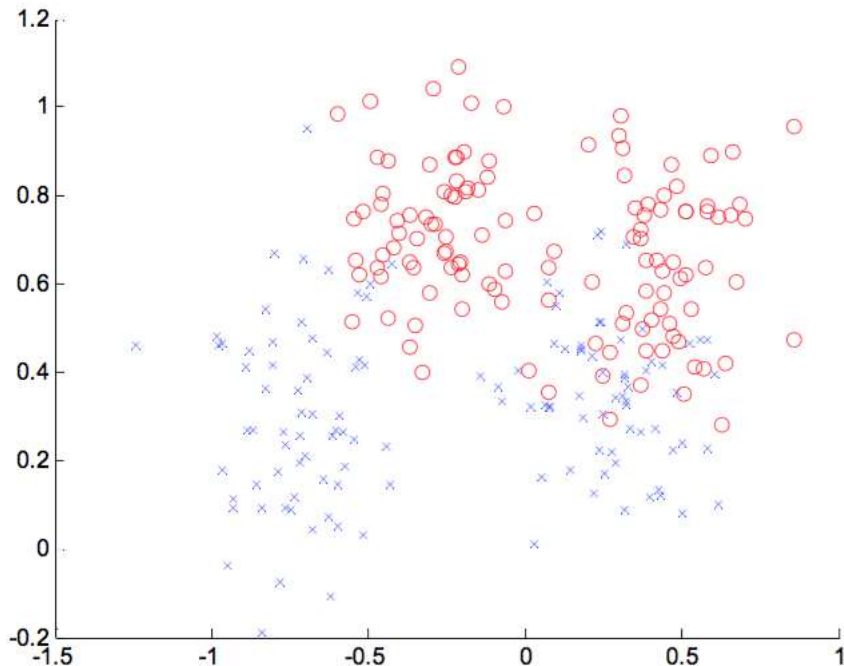
1-NN versus k-NN



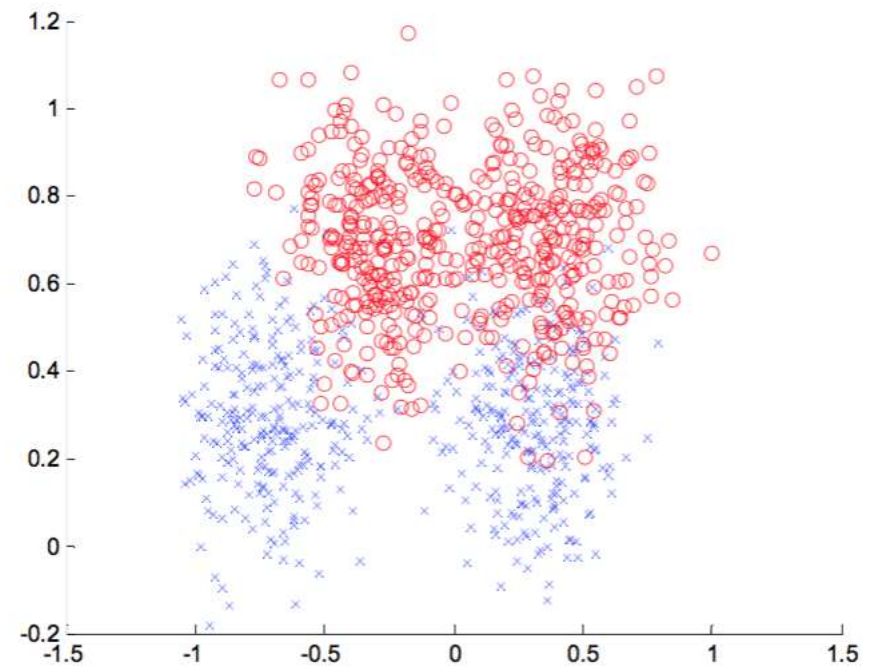
1-NN versus k-NN



Presupunerea pe care se bazează modelul k-NN



Training data

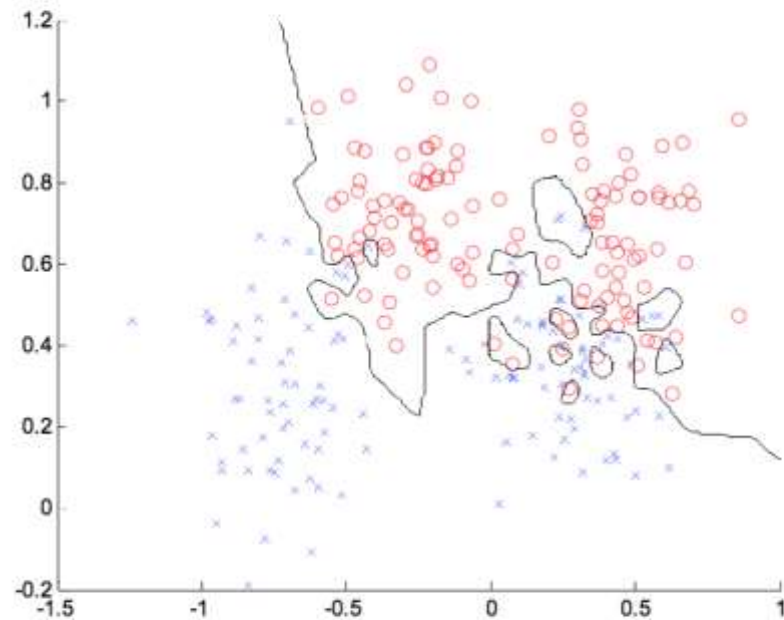


Testing data

- 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

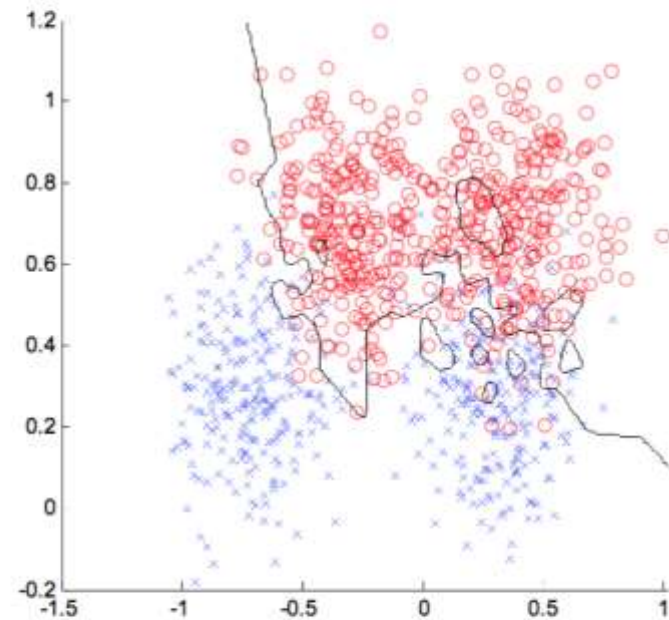
Ce se întâmplă atunci când variem parametrul k ?

Training data



error = 0.0

Testing data

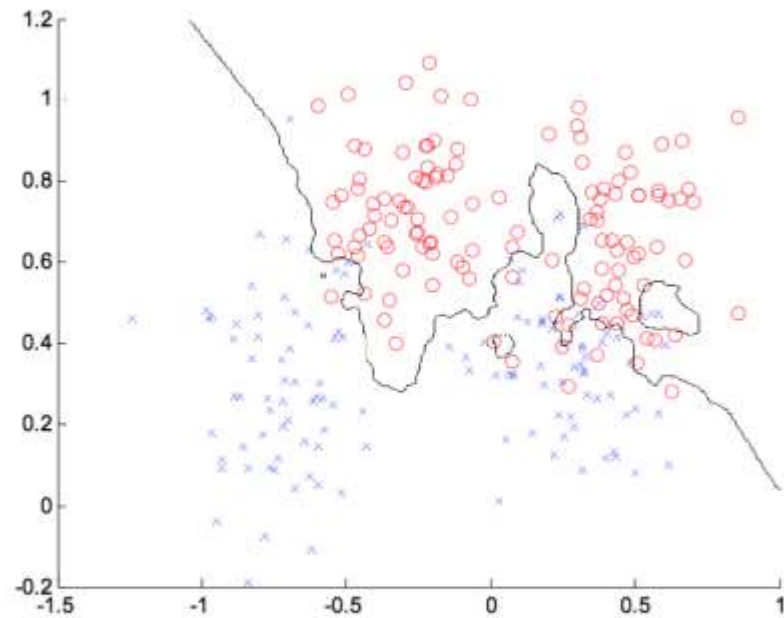


error = 0.15

- $k = 1$

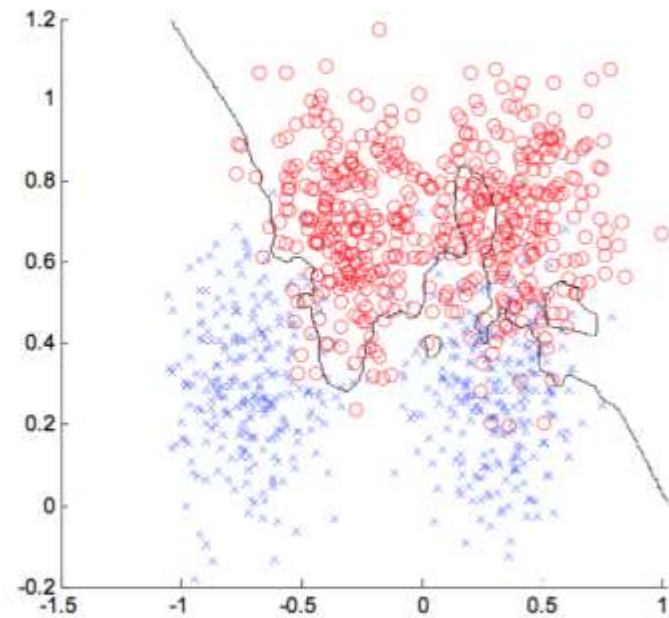
Ce se întâmplă atunci când variem parametrul k ?

Training data



error = 0.0760

Testing data

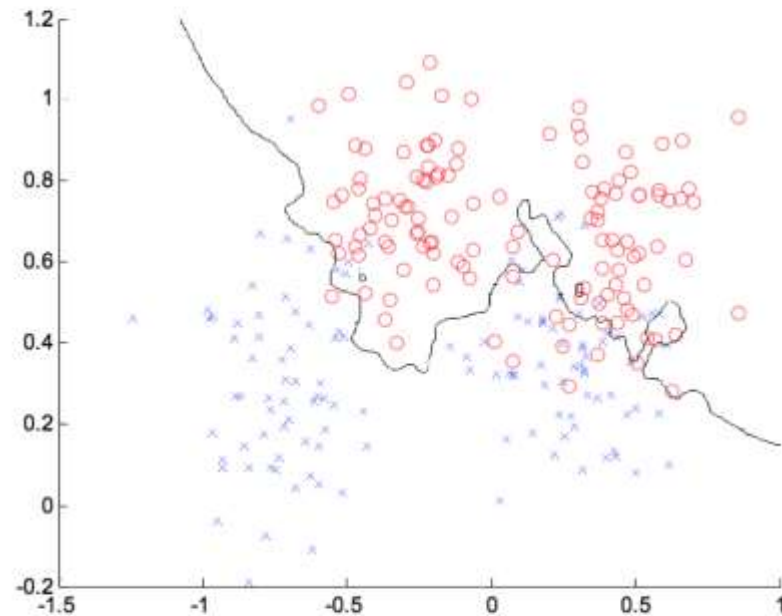


error = 0.1340

- $k = 3$

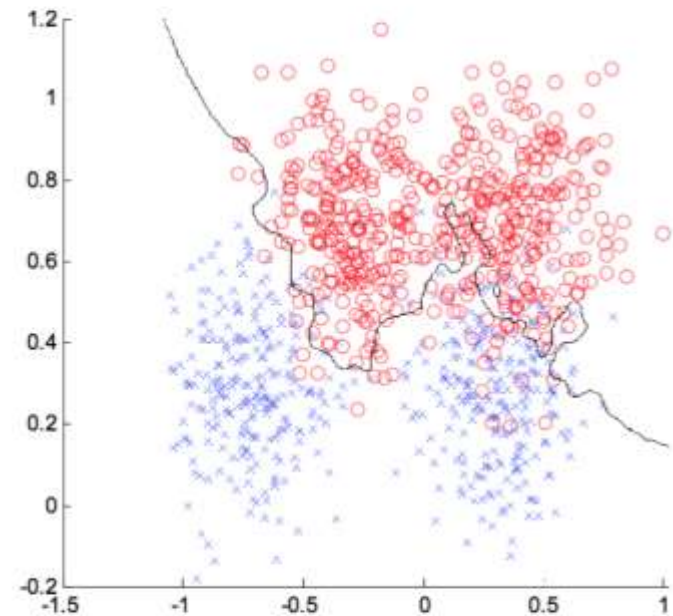
Ce se întâmplă atunci când variem parametrul k ?

Training data



error = 0.1320

Testing data

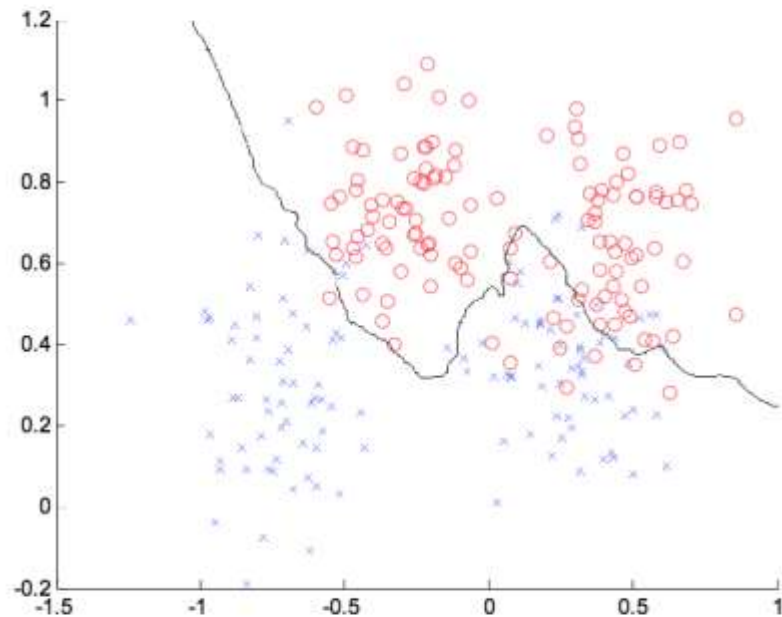


error = 0.1110

- $k = 7$

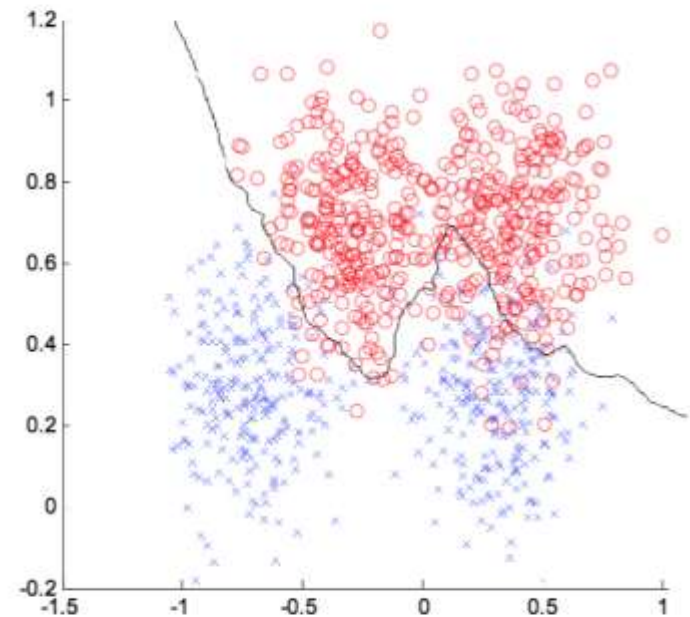
Ce se întâmplă atunci când variem parametrul k ?

Training data



error = 0.1120

Testing data



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

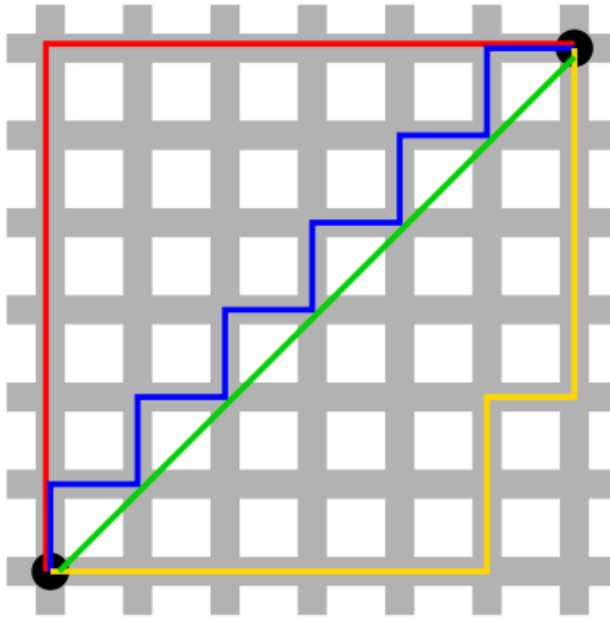
- O funcție de distanță
 - **De exemplu distanța Euclidiană**
- Câți vecini să luăm în considerare?
 - **1**
- Cum să antrenăm modelul pe exemplele din vecinătate?
 - **Prezicem eticheta celui mai apropiat vecin**

Distanța Euclidiană (L_2)

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

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

Distanța Manhattan (L_1)



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

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

Distanța Minkowski (L_p)

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

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

- 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:

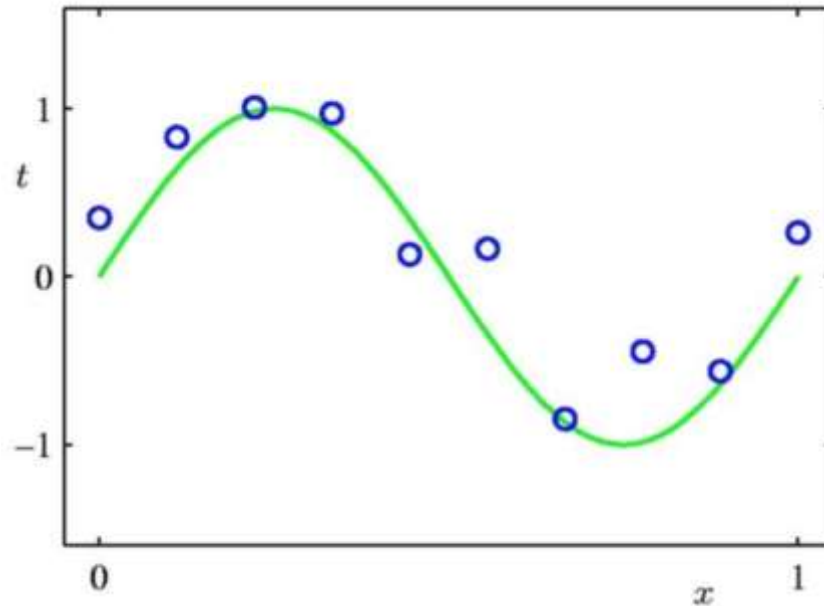
$$d_{Hamming}(x, y) = 1 + 0 + 0 + 1 = 2$$

- Câte trăsături (componente) diferă între cei doi vectori

Distanța Edit (Levenshtein)

- De exemplu, utilă pentru probleme de clasificare cu șiruri 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)

Regresie din exemple etichetate



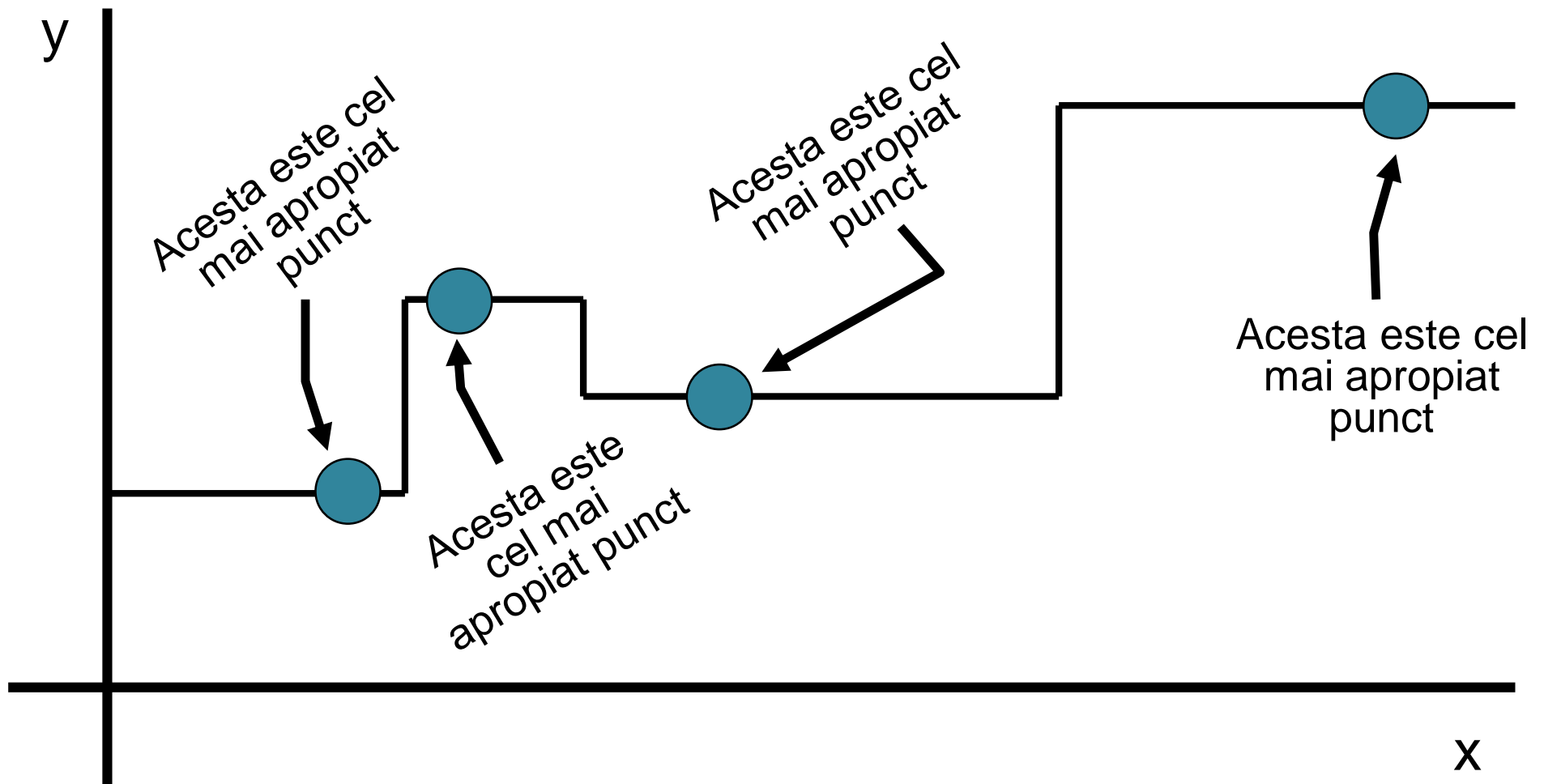
- Presupunem că avem un set de N exemple de antrenare:

$$(x_1, \dots, x_N) \text{ and } (y_1, \dots, y_N), x_i, y_i \in \mathbb{R}$$

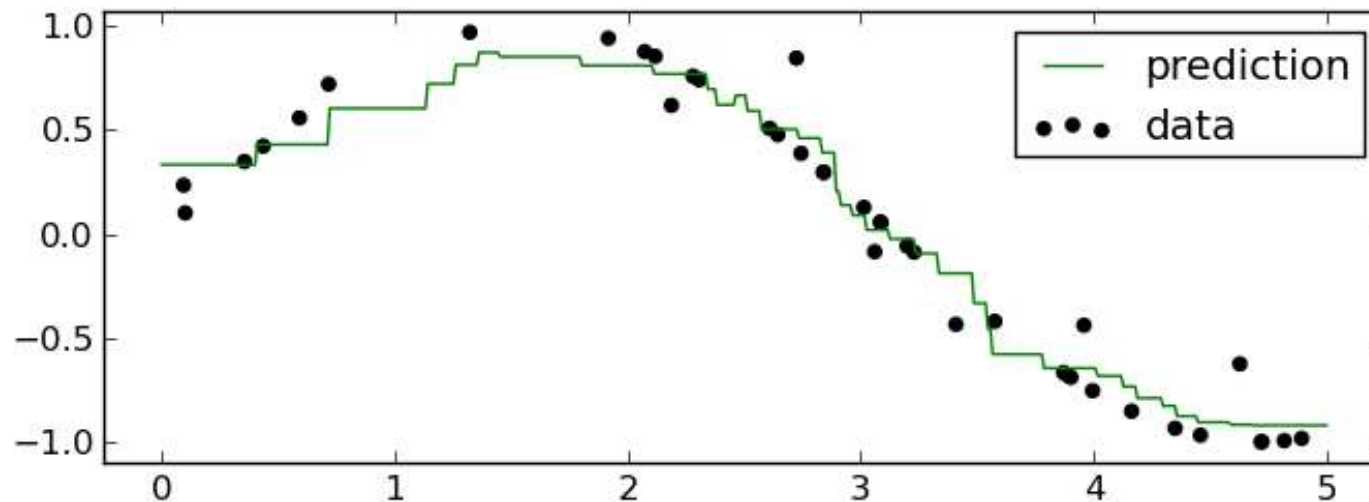
- Problema regresiei constă în estimarea funcției $g(x)$ a.î.:

$$g(x_i) = y_i$$

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 apropiați k vecini și etichetele lor
- 2) Output-ul este media etichetelor celor k vecini

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

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

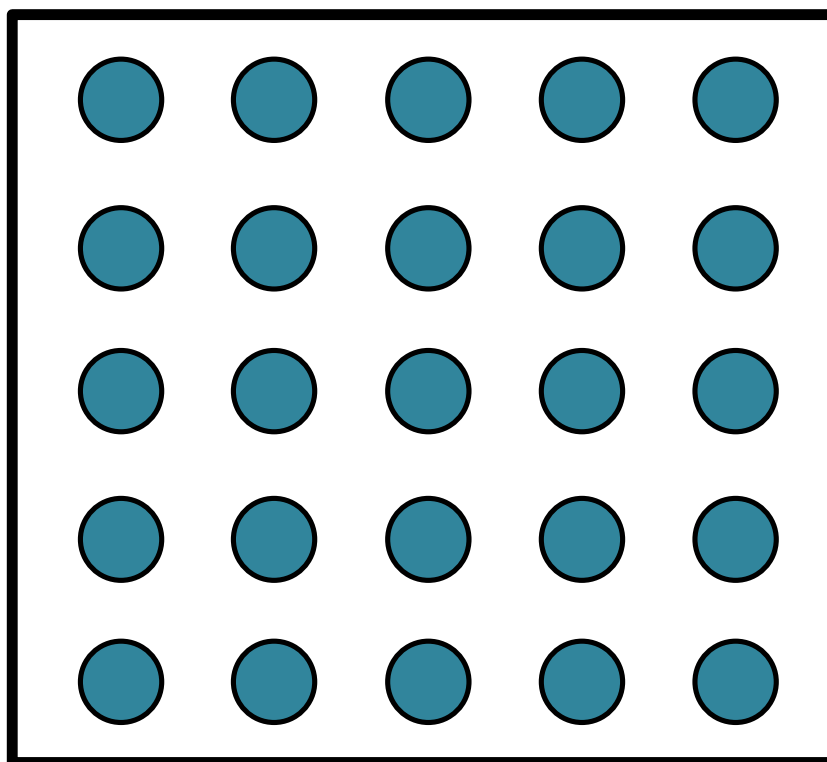
Blestemul dimensionalității

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



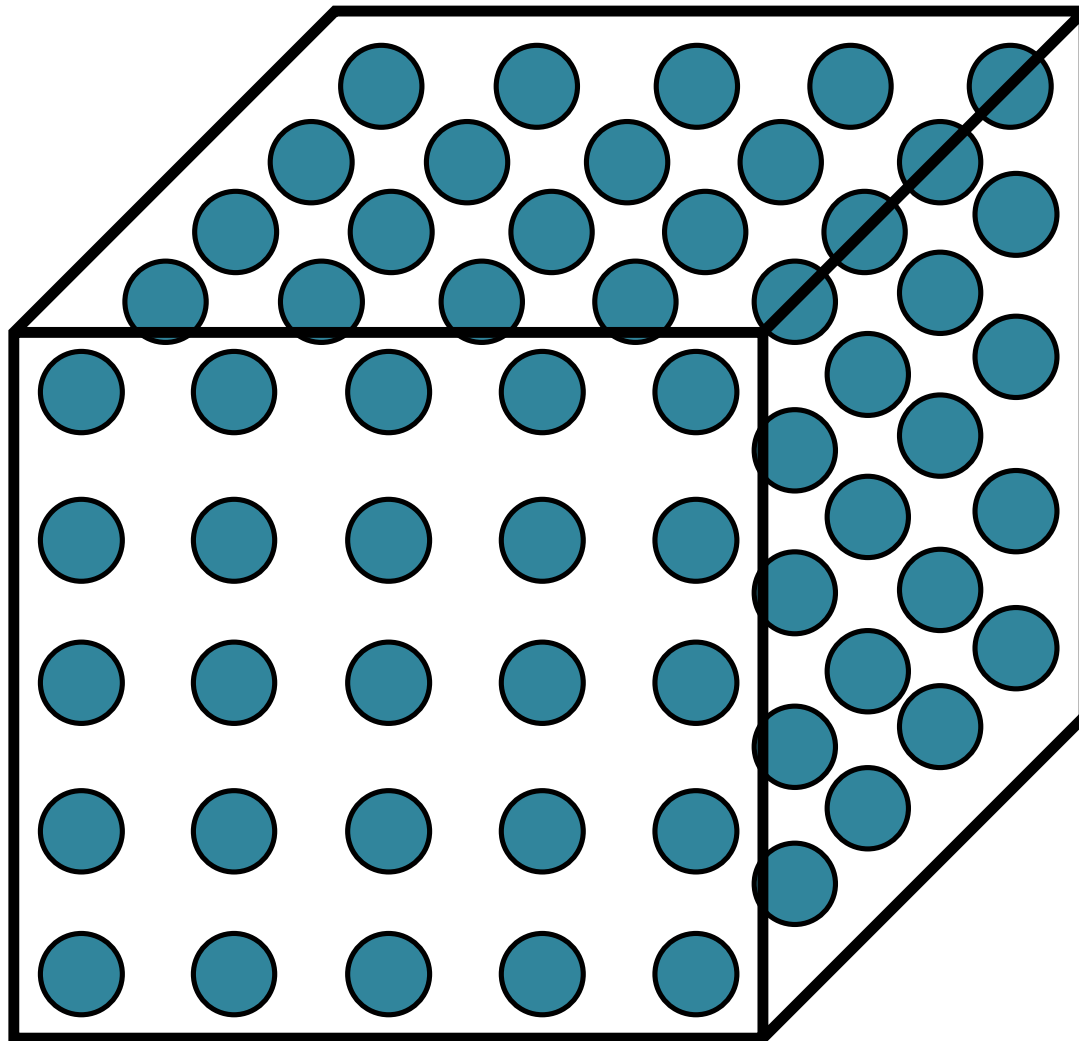
Blestemul dimensionalității

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



Blestemul dimensionalității

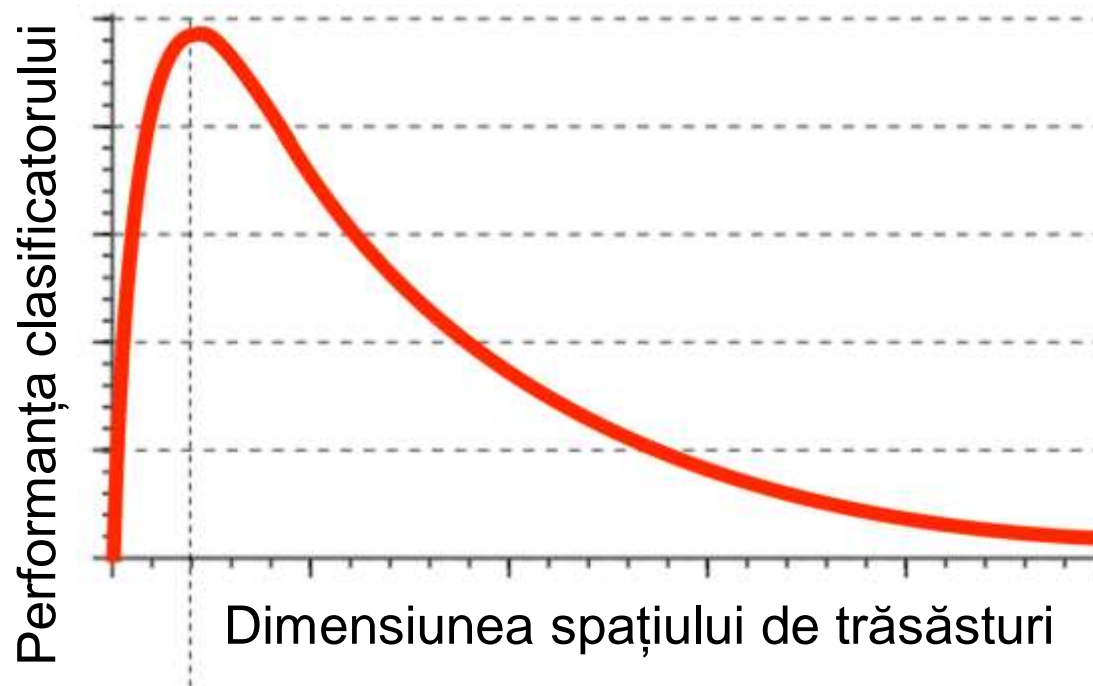
- Pentru a “umple” un spațiu 3D (de exemplu \mathbb{R}^3) avem nevoie de 125 puncte:



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 a învăța un model care să generalizeze
- De cele mai multe ori aceste date nu sunt disponibile în practică

Fenomenul Hughes



Numărul optim de trăsături

- 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

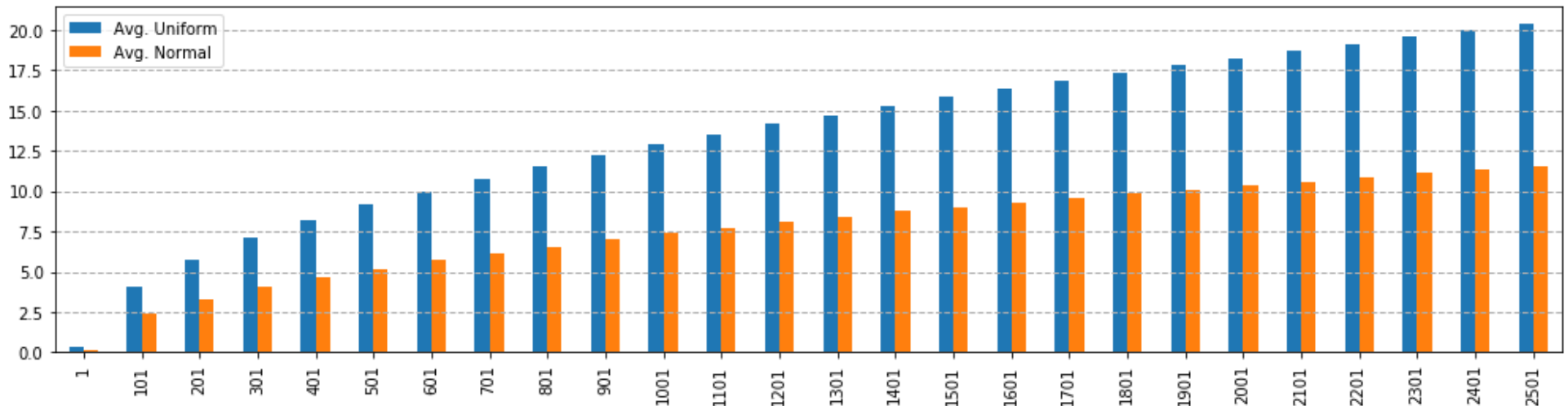
Blestemul dimensionalității

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

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

- 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



Distanța medie pentru 1000 de puncte din cubul unitate

- 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