

LAPI – Laboratorul de
Analiza și Prelucrarea
Imaginilor



Universitatea
POLITEHNICA din
București

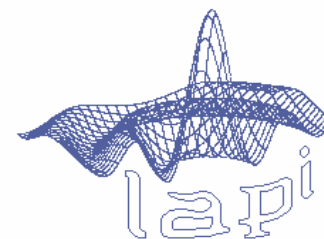


Facultatea de Electronică,
Telecomunicații și
Tehnologia Informației

Interfață Vizuală Om-Mașină

Analiza și recunoașterea gesturilor

Dr.ing. Ionuț Mironică



LAPCM

IV. Clasificarea gesturilor

- Introducere – algoritmi de învățare
- Învățare supervizată
- Concluzii

IV. Clasificarea gesturilor

Introducere - algoritmi de învățare

Ce este Machine learning?

- Este foarte greu de scris un algoritm care recunoaște un set de gesturi statice / dinamice sau care să rezolve problema de recunoaștere a feței:
 - Nu știm cum funcționează creierul uman pentru a clasifica gesturile;
 - Chiar dacă am ști nu am avea idee cum să programăm deoarece ar fi foarte complicat;
 - Ar trebui să scriem o funcție diferită pentru fiecare gest.
- În loc să scriem programe foarte multe, putem colecta exemple care specifică fiecare gest;
- Un algoritm de învățare va prelua aceste exemple și va “creea” un program care va face această clasificare în mod automat;

IV. Clasificarea gesturilor

Introducere - algoritmi de învățare

Ce este Machine learning?

- Există mii de algoritmi de învățare / sute dintre ei apar anual;
- Există mai multe tipuri de învățare:

Învățare supervizată

- datele de antrenare conțin și ieșirea dorită;

Învățare nesupervizată

- datele de antrenare nu conțin ieșirea dorită (clusterizare);
- ideea de bază este de a se găsi șabloane și pattern-uri în date care să fie evidențiate în mod automat.

Învățare semi-supervizată

- doar o parte din datele de antrenare conțin ieșirea dorită;

Reinforcement Learning

- se învață în funcție de feedback-ul primit după ce o decizie este luată.

IV. Clasificare

Introducere - algoritmi de învățare

Învățare nesupervizată

K-means

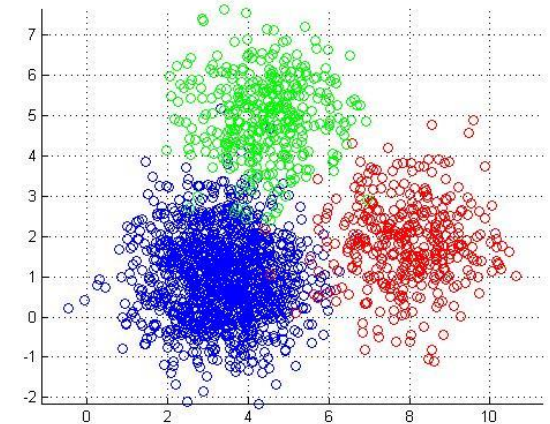
- Se re-asignează în mod iterativ punctele către cel mai apropiat vecin;

Clustering aglomerativ

- Fiecare punct reprezintă propriul său cluster și în mod iterativ se unesc cei mai apropiați centroizi;

Clustering MeanShift

- În funcție de funcția de densitate de probabilitate se estimează fiecare centroid.



IV. Clasificare

Introducere - algoritmi de învățare

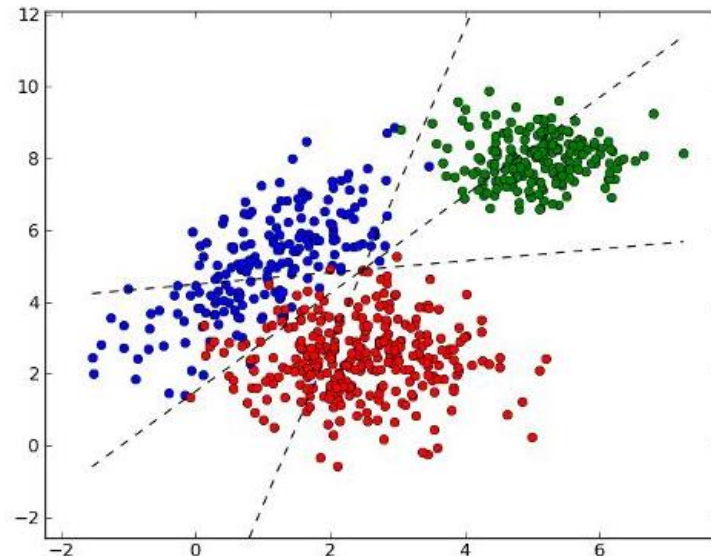
Învățare supervizată

- Se aplică o funcție de predicție la o trăsătură extrasă din imaginea sau documentul video iar acesta va avea ca ieșire clasa în care face parte gestul respectiv:

$f(\text{img}_1) = \text{"gest 1"}$

$f(\text{img}_2) = \text{"gest 2"}$

$f(\text{img}_3) = \text{"gest 3"}$



IV. Clasificare

Învățare supervizată

Model de bază

$$y = f(x)$$

The diagram shows the equation $y = f(x)$ in blue. Below it, three red arrows point upwards to the components of the equation: the first arrow points to y and is labeled 'Ieșire'; the second arrow points to f and is labeled 'Funcția de predicție'; the third arrow points to x and is labeled 'Trăsătură extrasă'.

Antrenare: fiind dată o mulțime de antrenare împreună cu răspunsul dorit $\{(x_1, y_1), \dots, (x_N, y_N)\}$, se estimează predicția funcției f prin minimizarea erorii de predicție pe mulțimea de antrenare;

Testare: se aplică funcția f pe un exemplu de test x (care nu a fost folosit în procesul de antrenare) și prezintă ieșirea funcției $y = f(x)$.

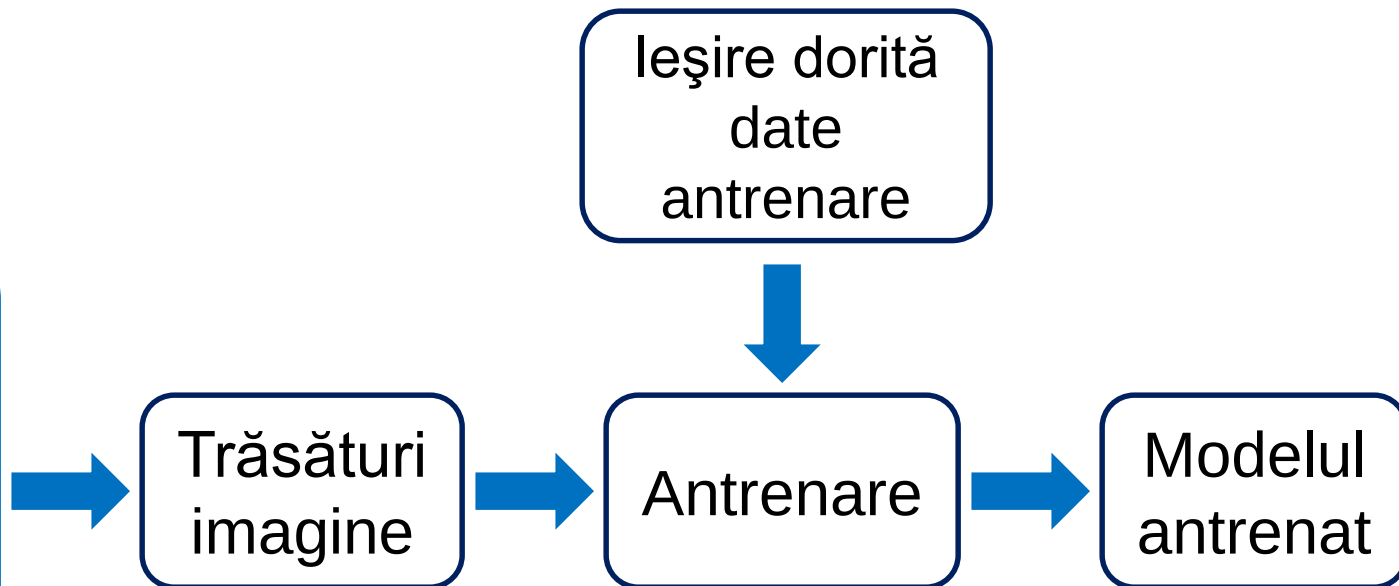
IV. Clasificare

Învățare supervizată

Model de bază

Antrenare

Imaginile de
antrenare

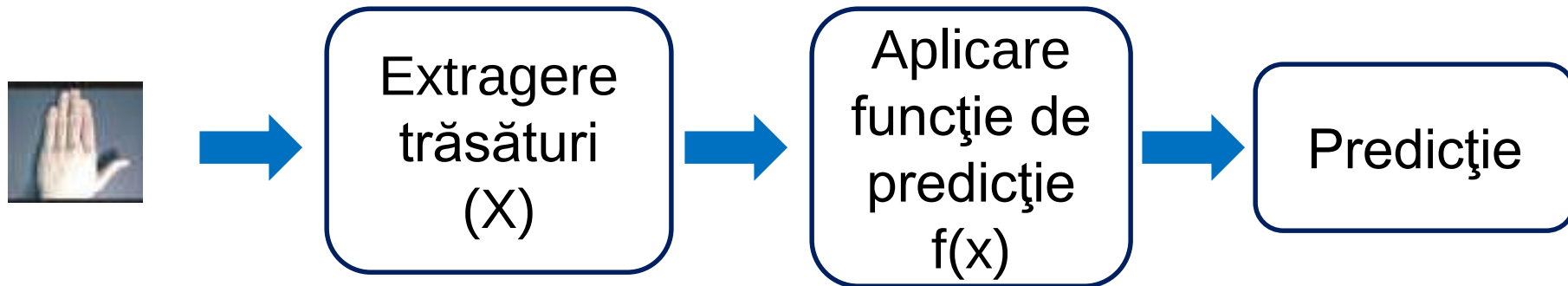


IV. Clasificare

Învățare supervizată

Model de bază

Testare



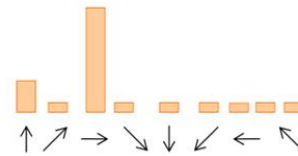
IV. Clasificare

Învățare supervizată

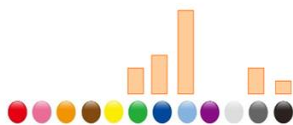
Trăsături extrase



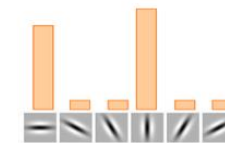
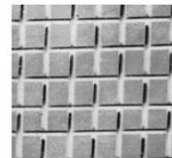
pixeli



muchii



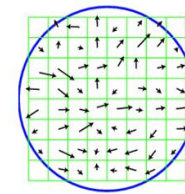
culoare



textură



formă

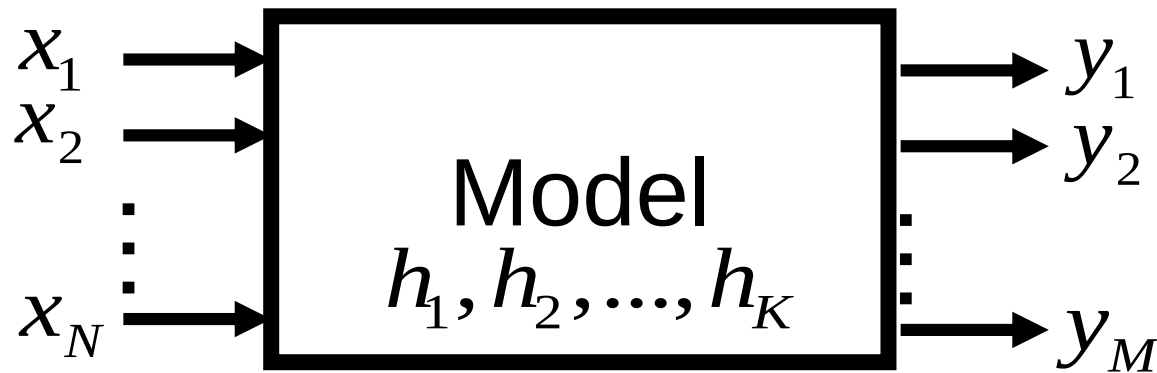


puncte de
interes

IV. Clasificare

Învățare supervizată

Învățare supervizată – schemă de bază



Variabile de intrare: $\mathbf{x} = (x_1, x_2, \dots, x_N)$

Variabile ascunse: $\mathbf{h} = (h_1, h_2, \dots, h_K)$

Variabile de ieșire: $\mathbf{y} = (y_1, y_2, \dots, y_K)$

IV. Clasificare

Învățare supervizată

Algoritmi existenți

- Support vector machines (SVM),
 - Rețele neurale,
 - Naïve Bayes,
 - Rețele bayesiene,
 - Arbori aleatorii (Random trees),
 - K-nearest neighbor (K-NN),
- Etc.

Care este cel mai bun algoritm?

IV. Clasificare

Învățare supervizată

Teorema “No free lunch”



IV. Clasificare

Învățare supervizată

Puterea de generalizare

Bias: (bias = ipoteza de lucru apriori) cât de mult diferă modelul mediu față de setul de antrenare?

- În funcție de gradul de adevăr al presupunerilor / simplificărilor pot apare diferite erori de modelare.

Varianță: cât de mult modelele estimate pe setul de antrenare diferă de cele pe care se va face testarea.

IV. Clasificare

Învățare supervizată

Puterea de generalizare

Underfitting: modelul este prea “simplu” pentru a reprezenta toate caracteristicile relevante ale claselor:

- Bias ridicat și variație scăzută;
- Eroare de antrenare ridicată și eroare de testare scăzută.

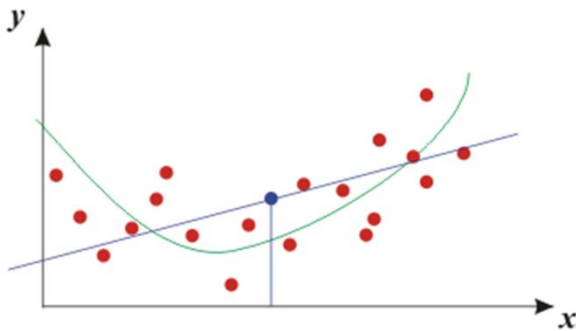
Overfitting: modelul este prea “complex” și modelează caracteristici irelevante (zgomot):

- Bias scăzut și varianță mare;
- Eroare de antrenare scăzută și eroare de testare ridicată.

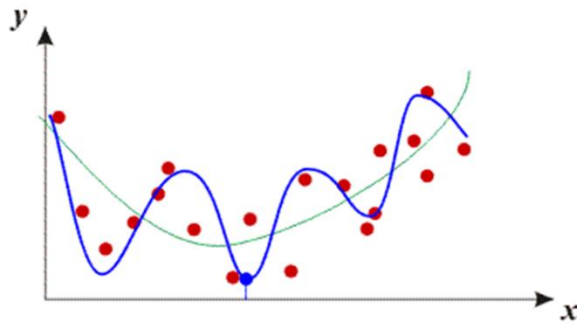
IV. Clasificare

Învățare supervizată

Compromisul dintre bias și varianță



Underfitting - modelele cu prea puțini parametri sunt inexacte deoarece modelul este prea simplu (prea multă flexibilitate).



Overfitting - modelele cu prea mulți parametri sunt inexacte (prea multă sensibilitate la datele de intrare pentru antrenare).

IV. Clasificare

Învățare supervizată

Compromisul dintre bias și variație

$$\text{Eroare} = \text{zgomot} + \text{bias} + \text{varianță}$$

Erori care nu
pot fi eliminate



Erori datorate
presupunerilor
false



Erori datorate variației
elementelor de intrare



IV. Clasificare

Învățare supervizată

Bias

- Bias-ul reprezintă presupunerile efectuate de model pentru a face funcția de optimizat mai ușor de învățat;
- În general, algoritmi simpli au un bias ridicat, aceștia fiind ușor de înțeles dar mai puțin flexibili;
- Au o performanță scăzută pentru probleme complexe.

Exemple de algoritmi cu bias scăzut:

- arbori de decizie, random forests, k-Nearest Neighbors, Support Vector Machines.

Exemple de algoritmi cu bias ridicat:

- regresie liniară și logistică.

IV. Clasificare

Învățare supervizată

Varianță

Varianța ridicată reprezintă faptul că modelul poate fi schimbat dacă se utilizează diferite date de antrenare.

Funcția țintă este estimată pe baza datelor de antrenare, așa că este de așteptat ca fiecare algoritm să prezinte varianță, dar ideal este ca aceasta să nu se schimbe prea mult.

Exemple de algoritmi cu varianță scăzută:

- regresie liniară și logistică.

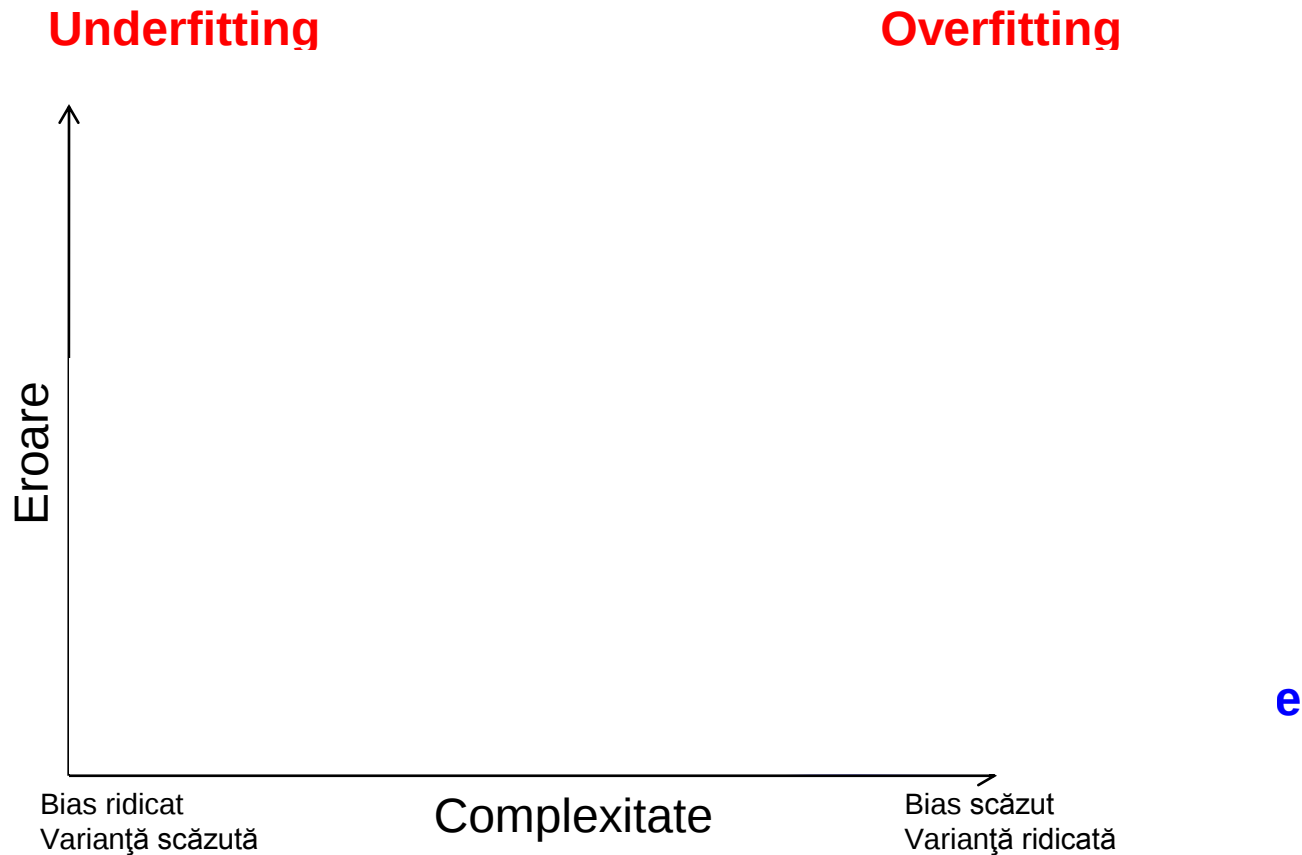
Exemple de algoritmi cu varianță ridicată:

- Arbori de decizie (mai ales cei nefasonați), SVM, k-Nearest Neighbor.

IV. Clasificare

Învățare supervizată

Compromisul dintre bias și varianță

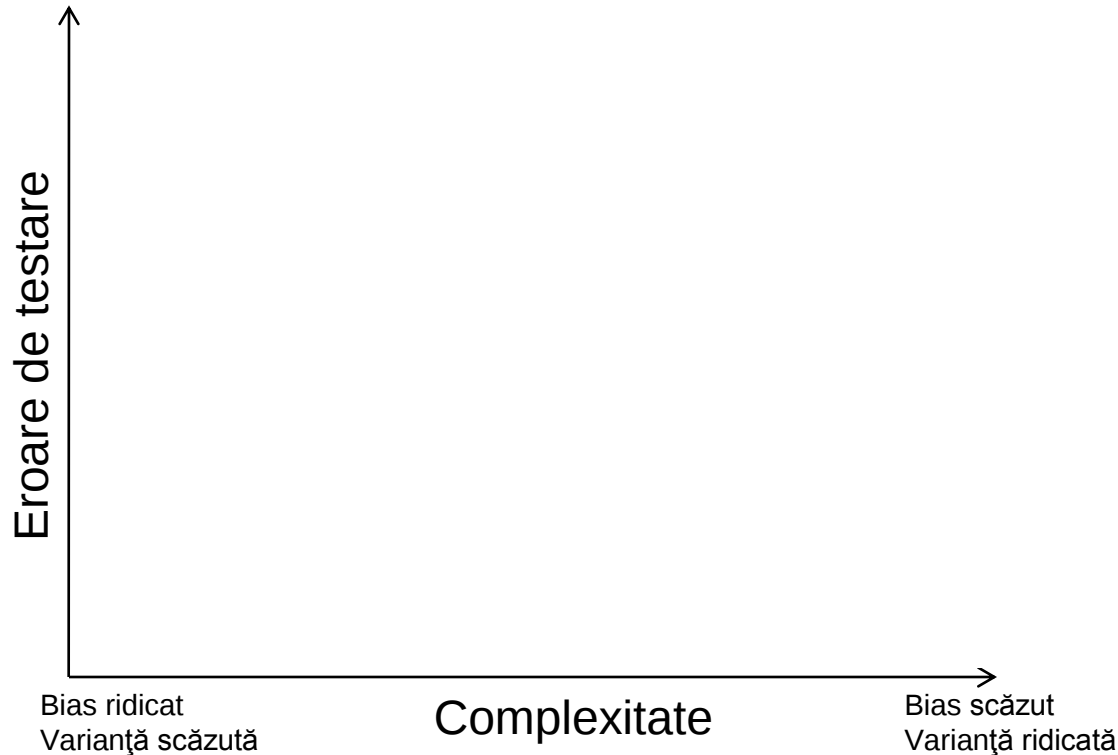


Slide: D. Hoiem

IV. Clasificare

Învățare supervizată

Compromisul dintre bias și varianță



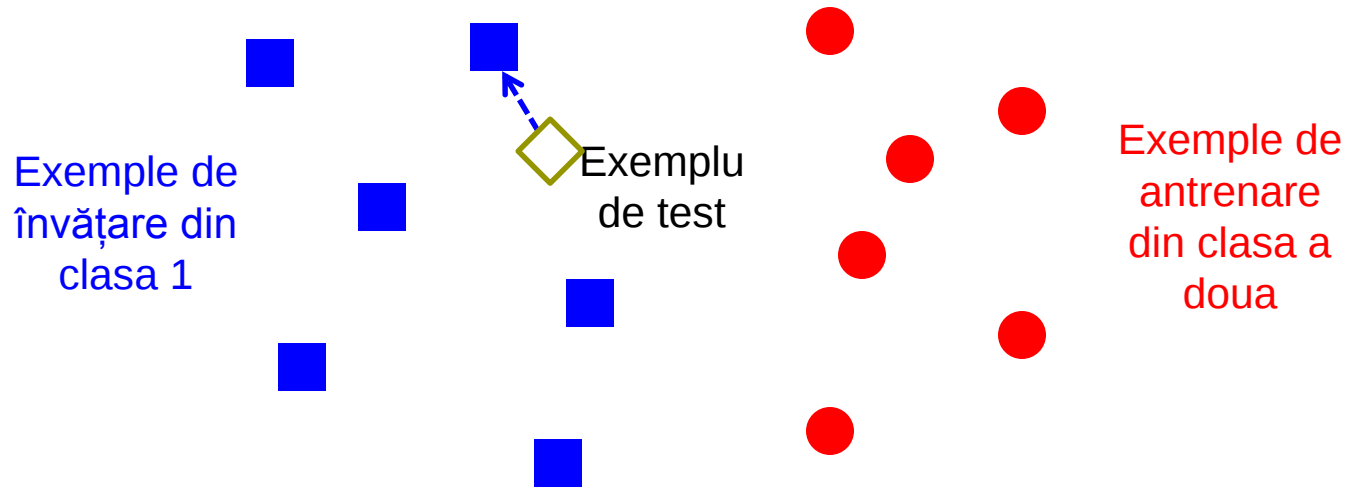
Slide: D. Hoiem

IV. Clasificare

Învățare supervizată

K-Nearest Neighbor (Cei mai apropiați vecini)

K=1



$f(x)$ = va lua valoarea celui mai apropiat vecin a lui x

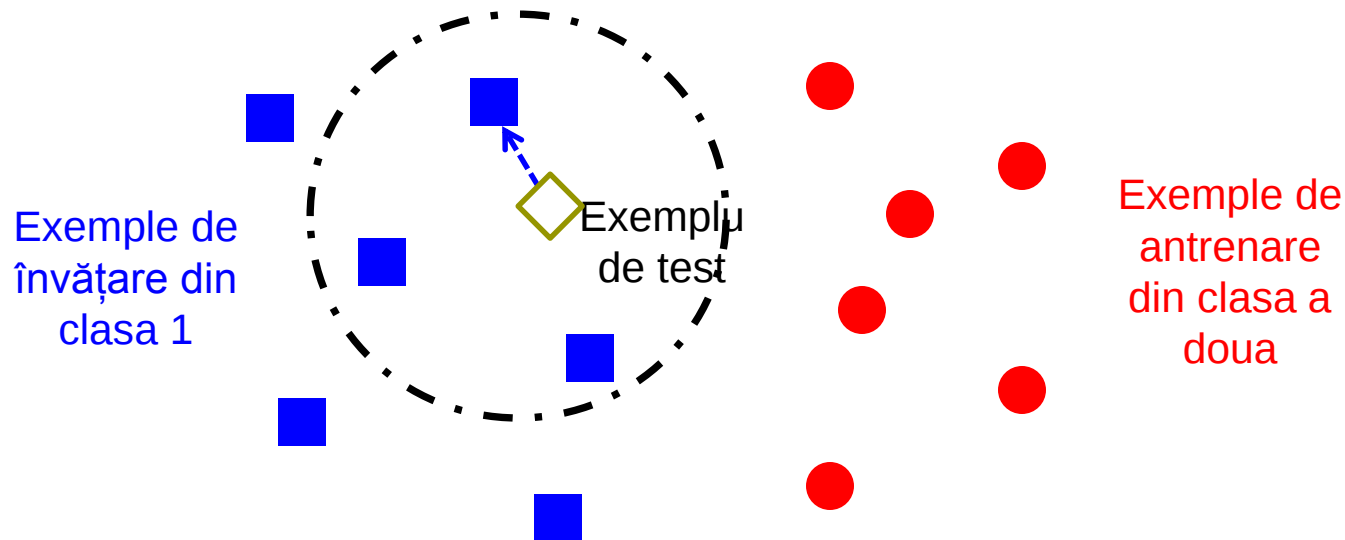
- Tot ceea ce este nevoie este distanța dintre x și toate trăsăturile din baza de antrenare;
- Nu este nevoie de un proces de antrenare.

IV. Clasificare

Învățare supervizată

K-Nearest Neighbor (Cei mai apropiați vecini)

$K=3$

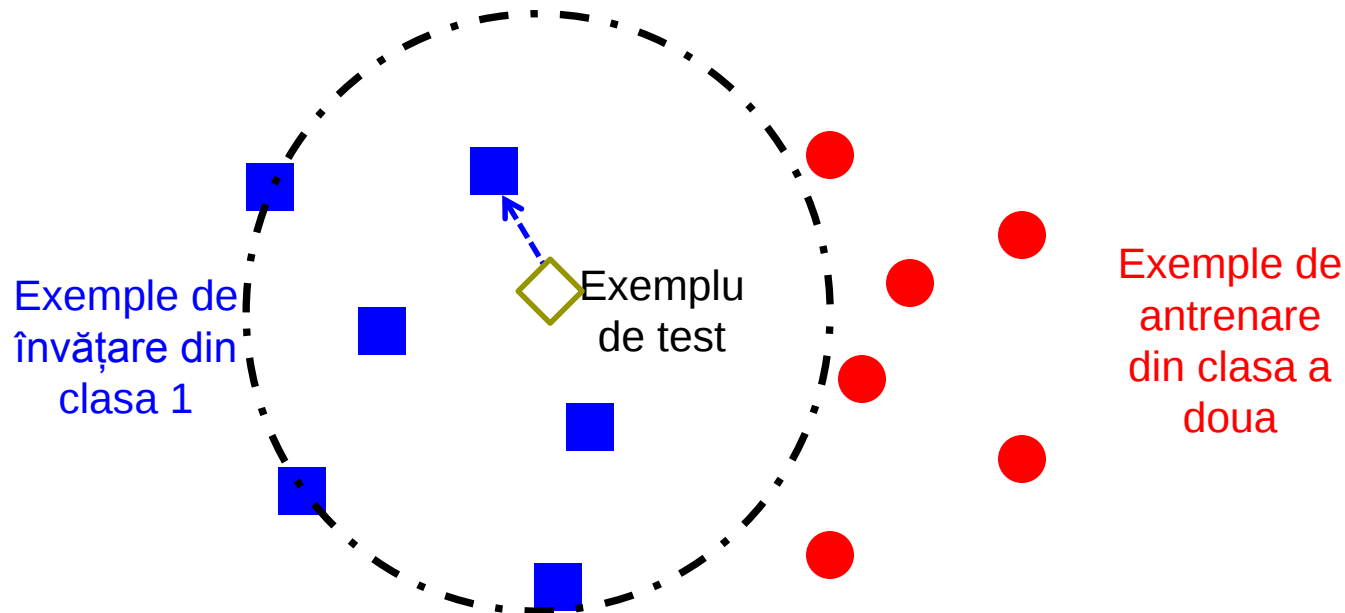


IV. Clasificare

Învățare supervizată

K-Nearest Neighbor (Cei mai apropiați vecini)

K=5



IV. Clasificare

Învățare supervizată

K-Nearest Neighbor (Cei mai apropiați vecini)

Avantaje

- Simplu (ușor de implementat pentru o primă încercare);
- Erori mari de clasificare (nu are rezultate bune pentru probleme complexe);

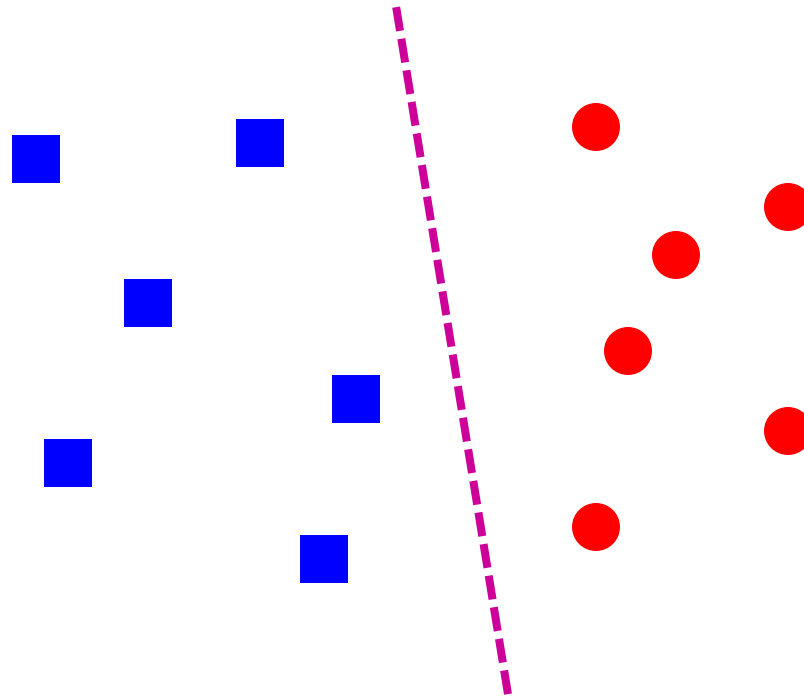
Dezavantaje

- Ocupă foarte multă memorie (se reține în memorie toată baza de antrenare);
- Lent – este nevoie să se compare elementul de clasificat cu toate trăsăturile din baza de date;
- Dependent de tipul de metrică utilizat.

IV. Clasificare

Învățare supervizată

Clasificator liniar



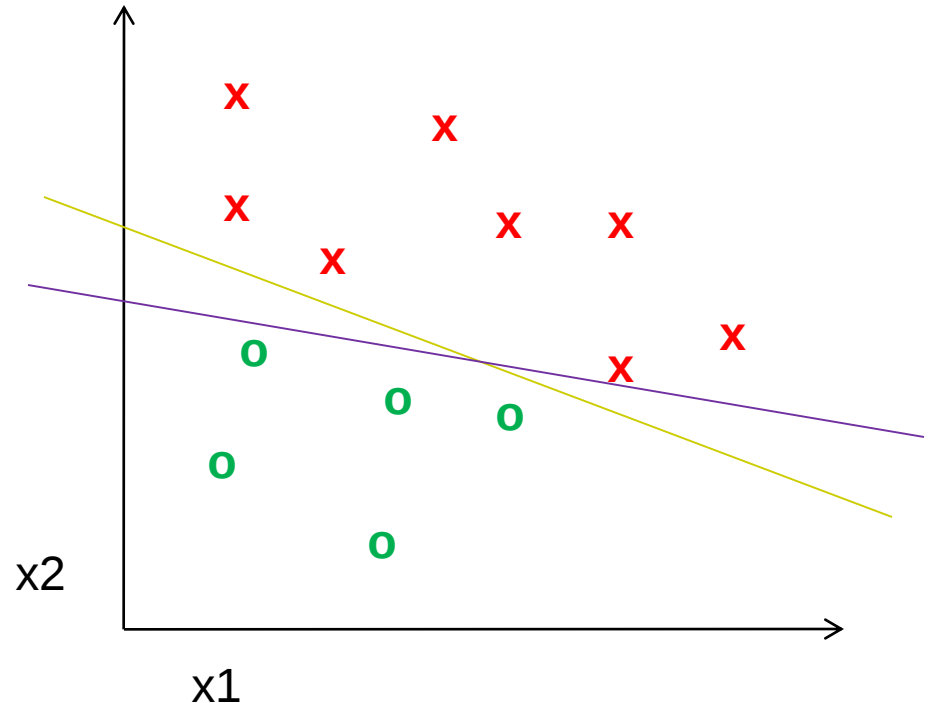
Caută o funcție liniară care separă clasele

$$f(\mathbf{x}) = \text{sgn}(\mathbf{w} \cdot \mathbf{x} + b)$$

IV. Clasificare

Învățare supervizată

SVM liniar



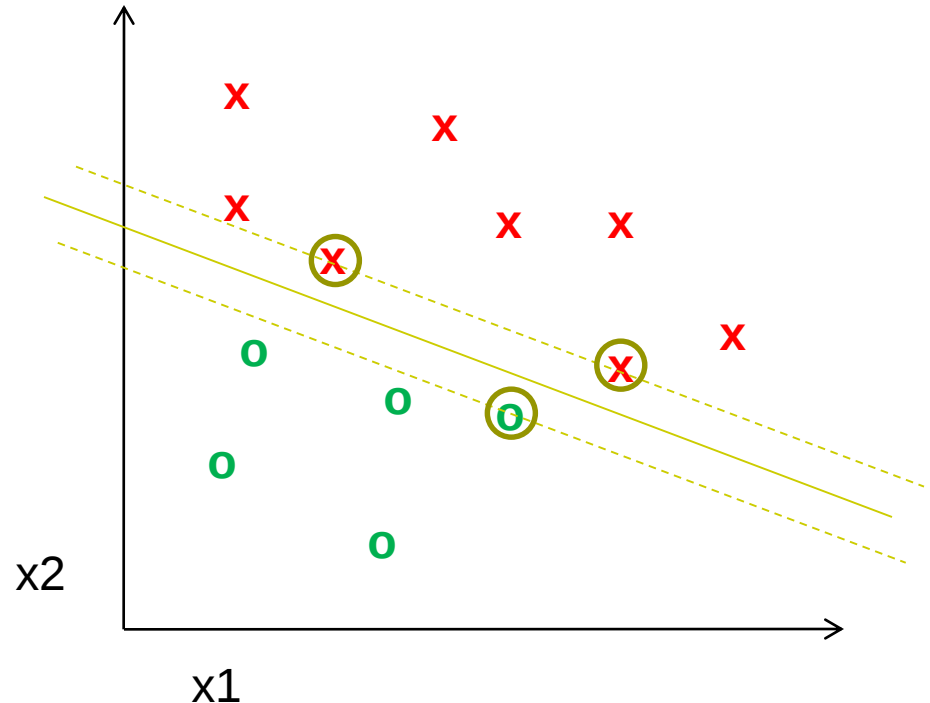
- Caută o funcție ce separă două clase în mod optim:

$$f(\mathbf{x}) = \text{sgn}(\mathbf{w} \cdot \mathbf{x} + b)$$

IV. Clasificare

Învățare supervizată

SVM liniar



- Caută o funcție ce separă două clase în mod optim:

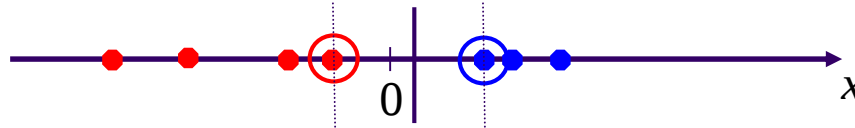
$$f(\mathbf{x}) = \text{sgn}(\mathbf{w} \cdot \mathbf{x} + b)$$

IV. Clasificare

Învățare supervizată

SVM liniar

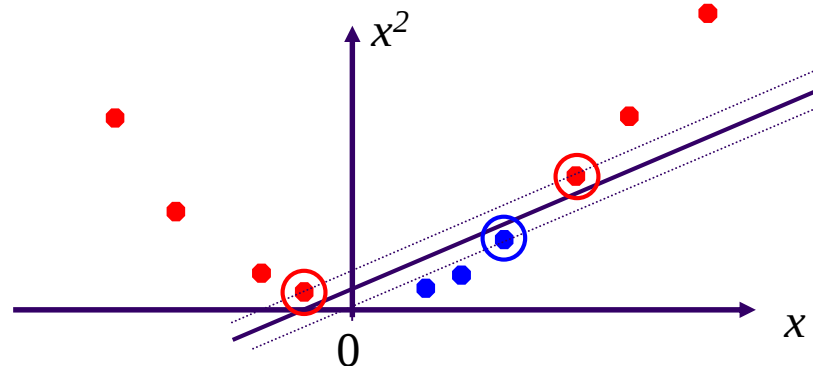
- Dacă clasele sunt liniar separabile, SVM liniar funcționează:



- Dar ce se întâmplă dacă baza de date este mai complicată?



- Soluție – se poate mapa pe un spațiu cu o dimensiune mai mare:

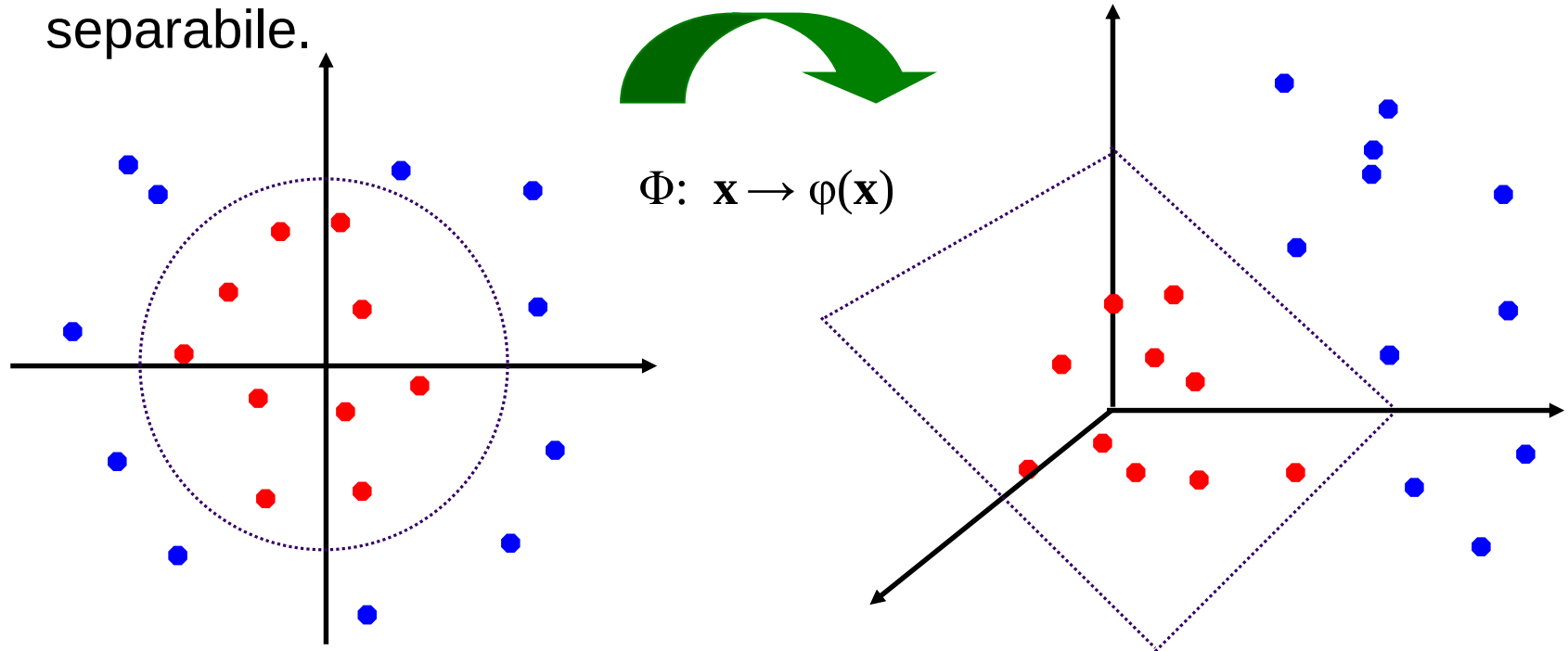


IV. Clasificare

Învățare supervizată

SVM neliniar

- Idee de bază: spațiul inițial poate fi mapat către un spațiu multidimensional în care trăsăturile de antrenare devin liniar separabile.

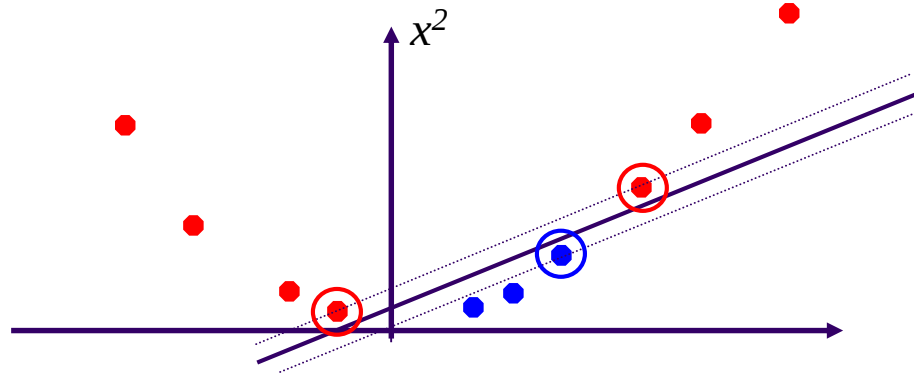


IV. Clasificare

Învățare supervizată

SVM neliniar - exemplu

Dacă se mapează funcția $\varphi(x) = (x, x^2)$



$$\varphi(x) \cdot \varphi(y) = (x, x^2) \cdot (y, y^2) = xy + x^2 y^2$$

$$K(x, y) = xy + x^2 y^2$$

IV. Clasificare

Învățare supervizată

SVM neliniar – exemple de nuclee

- Nucleul de intersecție de histogramă:

$$I(h_1, h_2) = \sum_{i=1}^N \min(h_1(i), h_2(i))$$

- Nucleu gaussian:

$$K(h_1, h_2) = \exp\left(-\frac{1}{A} D(h_1, h_2)^2\right)$$

IV. Clasificare

Învățare supervizată

SVM multclasă

- Din păcate, nu există un algoritm SVM adaptat pentru clasificarea de multclasă;
- În practică, se poate obține un algoritm SVM prin combinarea mai multor modele SVM:
 1. Unu vs. alții,
 2. Unu vs. unu.

IV. Clasificare

Învățare supervizată

SVM multclasă

- **Unu vs. alții**
 - Antrenare: se antrenează câte un model SVM pentru fiecare clasă vs celalalte;
 - Testare: se aplică fiecare model SVM și se alocă clasa care returnează cea mai mare valoare de încredere.
- **Unu vs. unu**
 - Antrenare: se antrenează un model SVM pentru fiecare pereche de clase;
 - Testare: fiecare SVM antrenat votează pentru o clasă, iar clasa desemnată va fi cea cu scorul cel mai mare.

IV. Clasificare

Învățare supervizată

SVM

Avantaje

- Multe implementări:
OpenCV, LibSVM, <http://www.kernel-machines.org/software> etc,
- SVM nonlinear are o putere de clasificare și generalizare foarte mare, fiind foarte flexibil;
- SVM lucrează bine chiar și cu o bază de antrenare foarte mică.

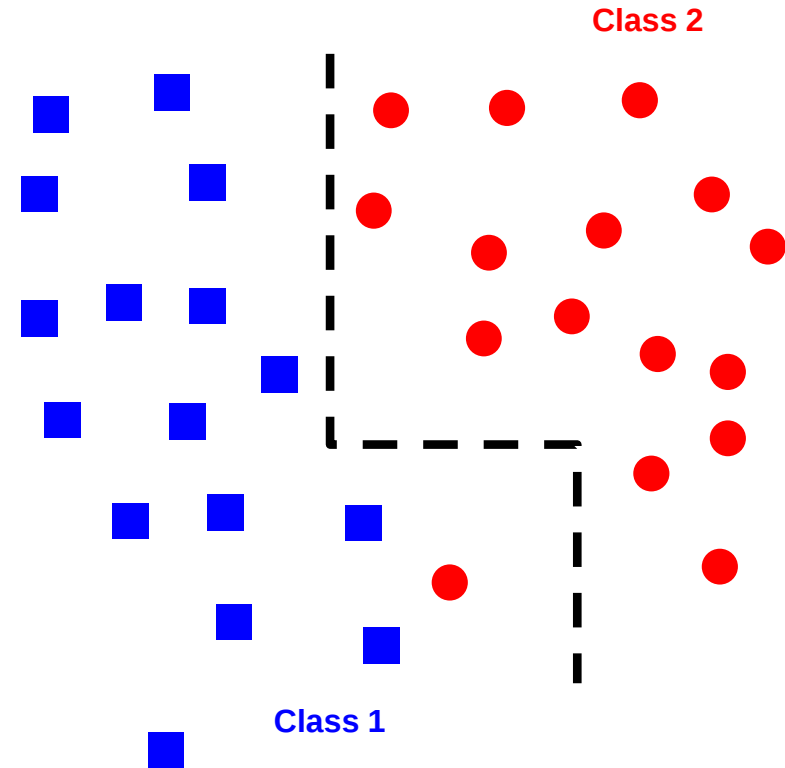
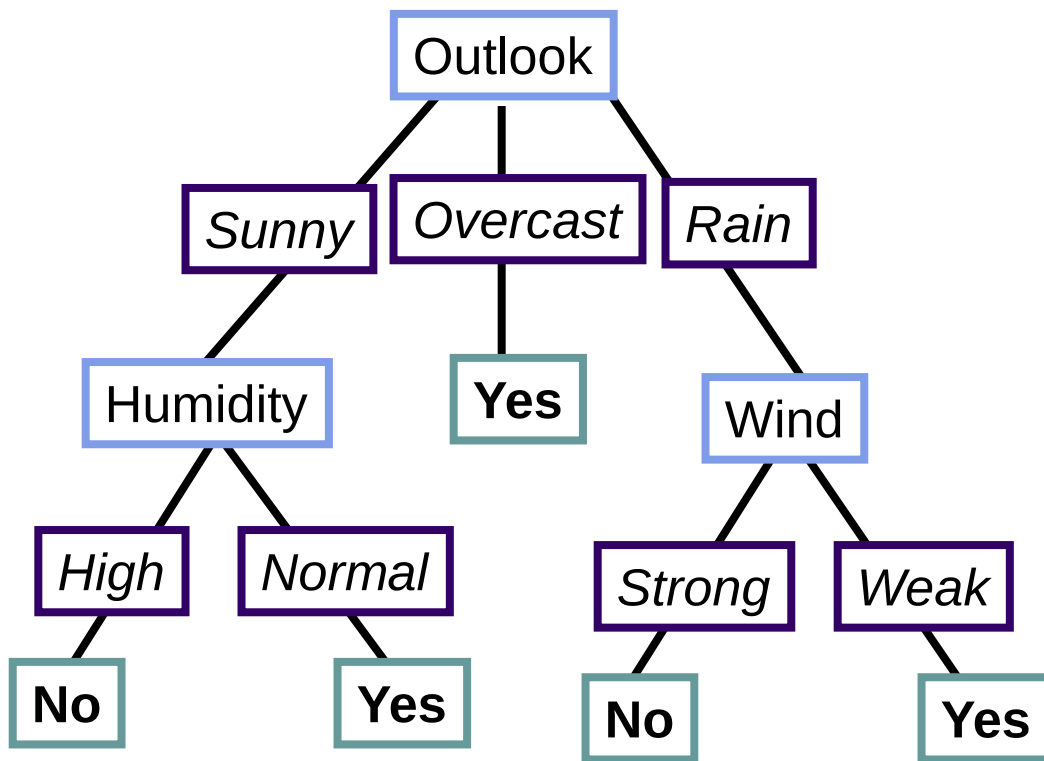
Dezavantaje

- Nu există SVM multiclasă (trebuie combinate mai multe modele SVM);
- Variantele nelineare sunt costisitoare din punct de vedere computațional (nu pot fi folosite pentru aplicații large-scale).

IV. Clasificare

Învățare supervizată

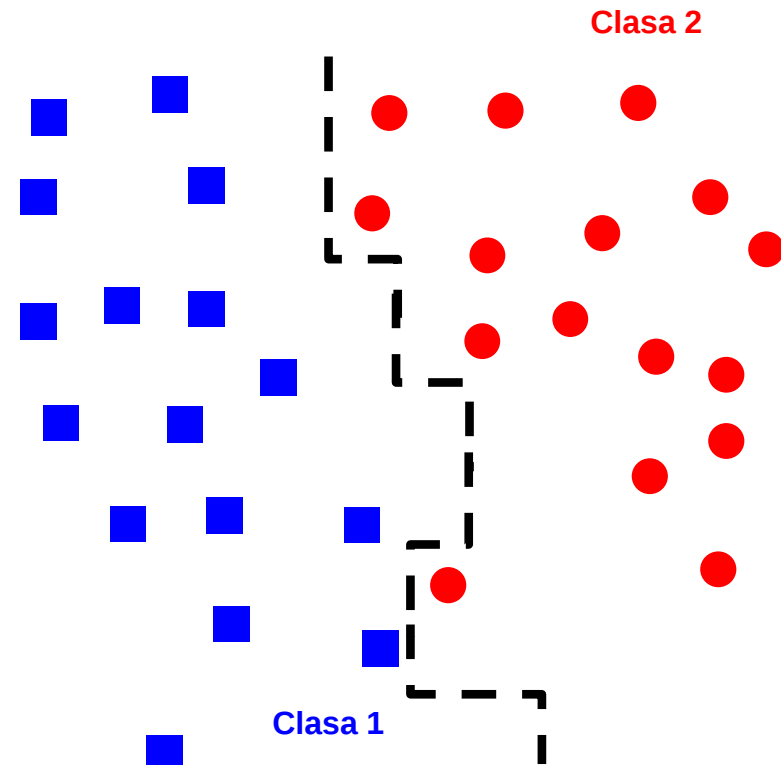
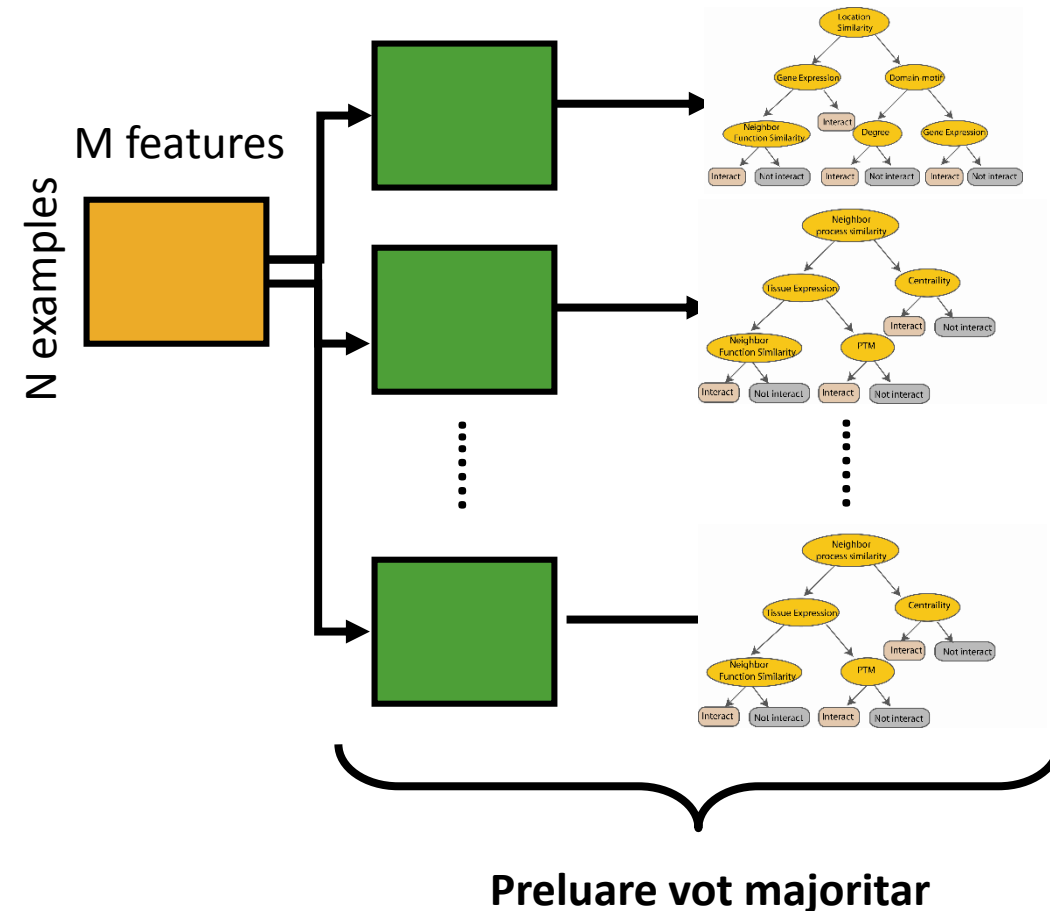
Arbori de decizie



IV. Clasificare

Învățare supervizată

Random forests



IV. Clasificare

Învățare supervizată

Naive Bayes

Clasificatorul Naive Bayes reprezintă un algoritm probabilistic bazat pe teorema lui Bayes, care pleacă de la premisa de independență între trăsături:

$$P(Y|X_1, \dots, X_n) = \frac{P(X_1, \dots, X_n|Y)P(Y)}{P(X_1, \dots, X_n)}$$

Likelihood Prior

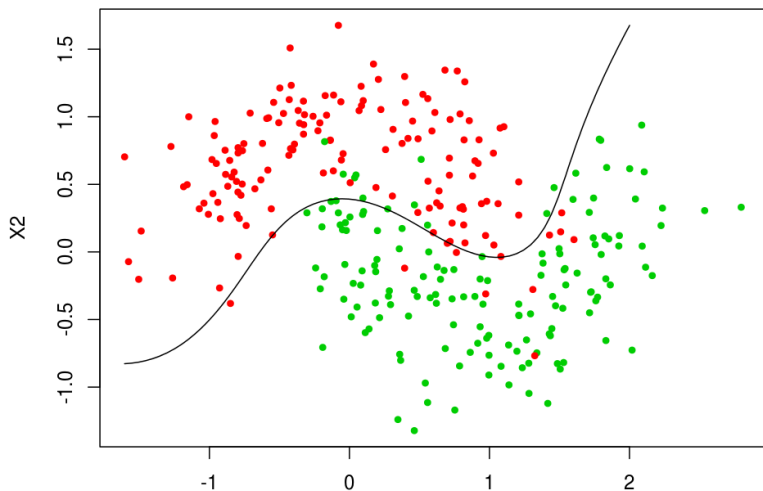
Normalization Constant

IV. Clasificare

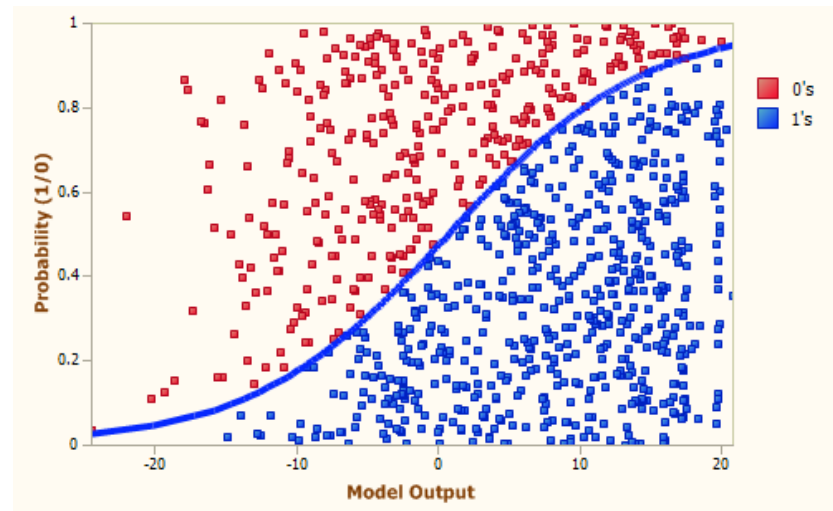
Învățare supervizată

Regresie logistică

Utilizează funcții matematice simple între cele două clase pentru a crea granițe între acestea.



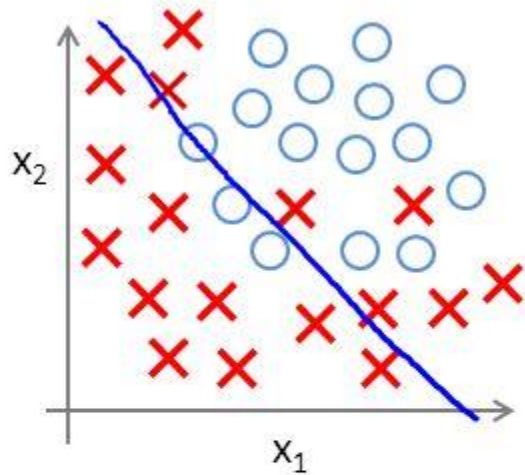
Funcții polinomiale



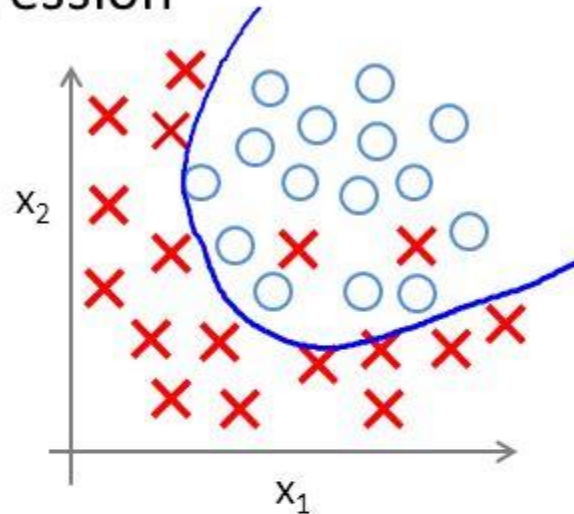
Funcția logistică

Logistic regression

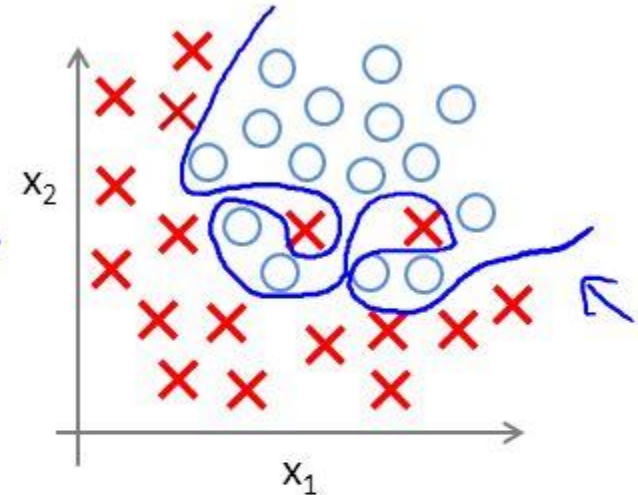
Example: Logistic regression



$\rightarrow h_{\theta}(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$
 (g = sigmoid function)
 "Underfit"

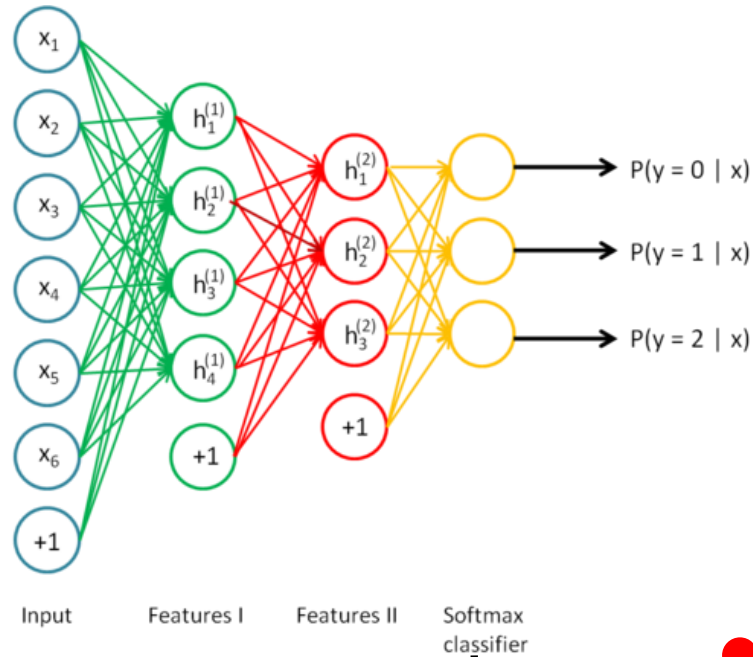
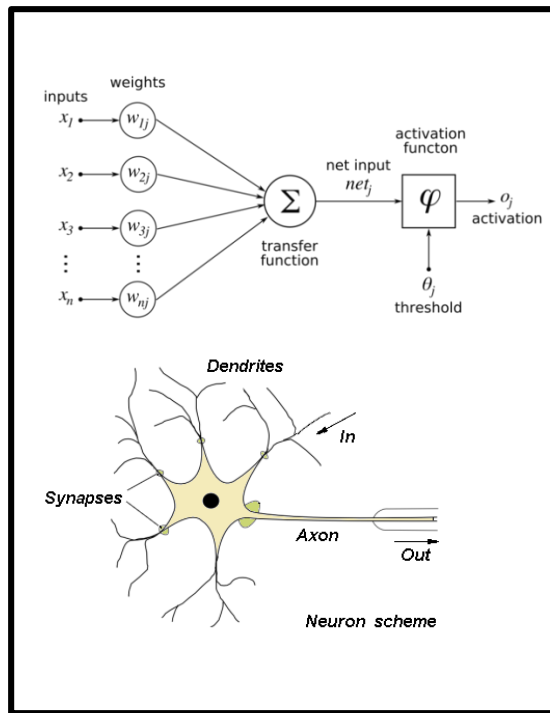


$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_5 x_1 x_2)$



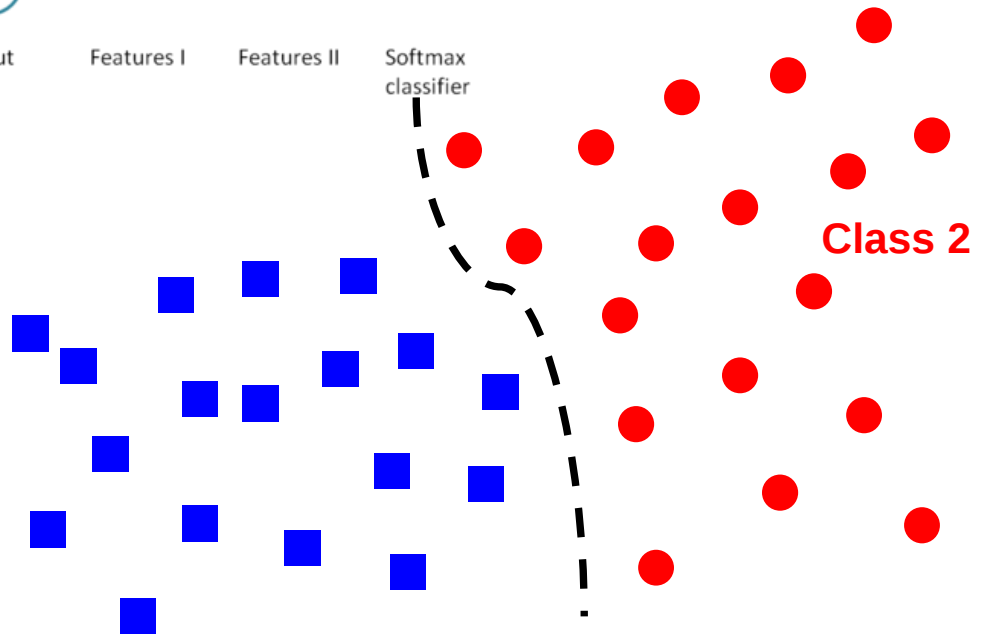
$g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2 + \theta_5 x_1^2 x_2^3 + \theta_6 x_1^3 x_2 + \dots)$
 "Overfit"

Neural networks

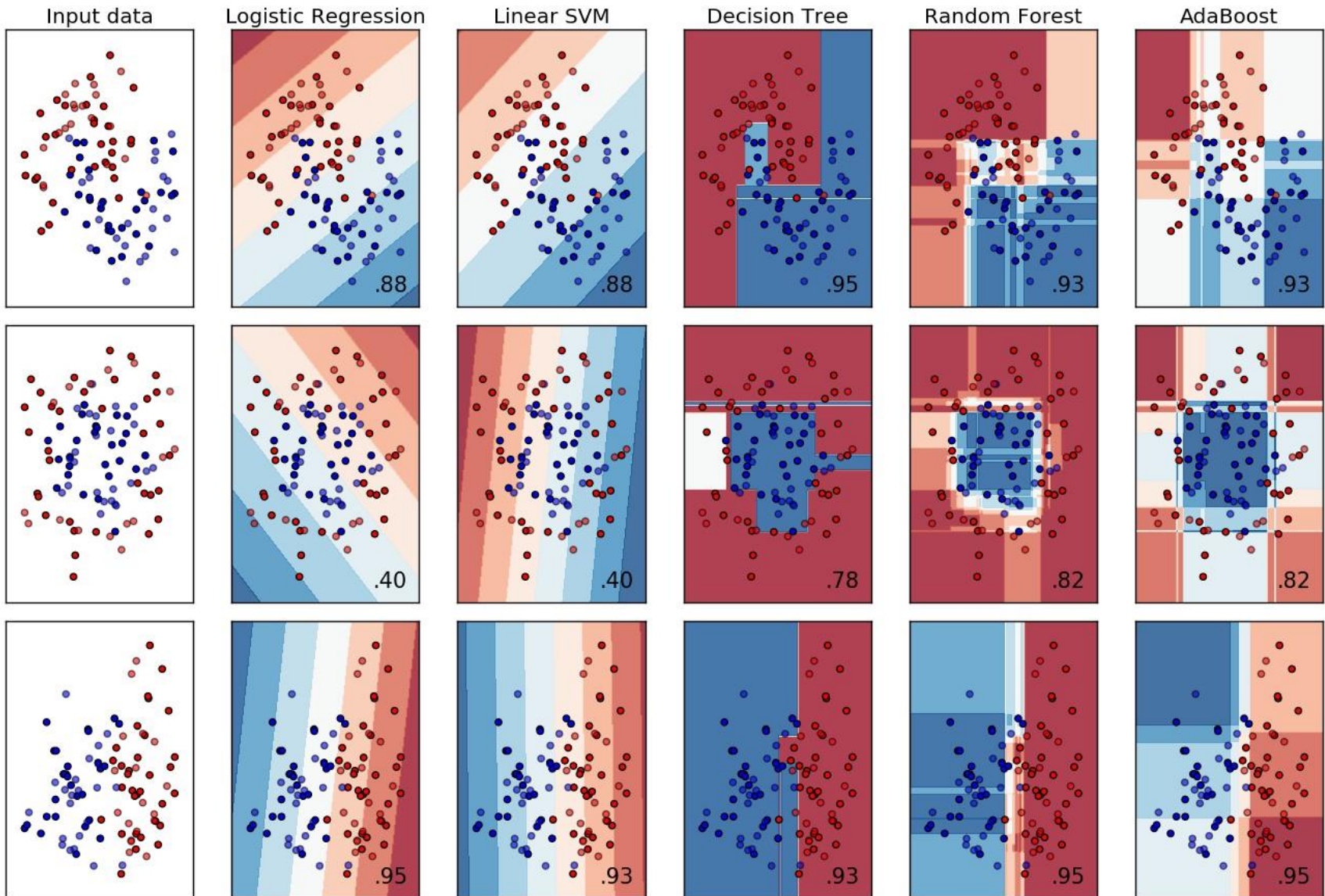


Class 1

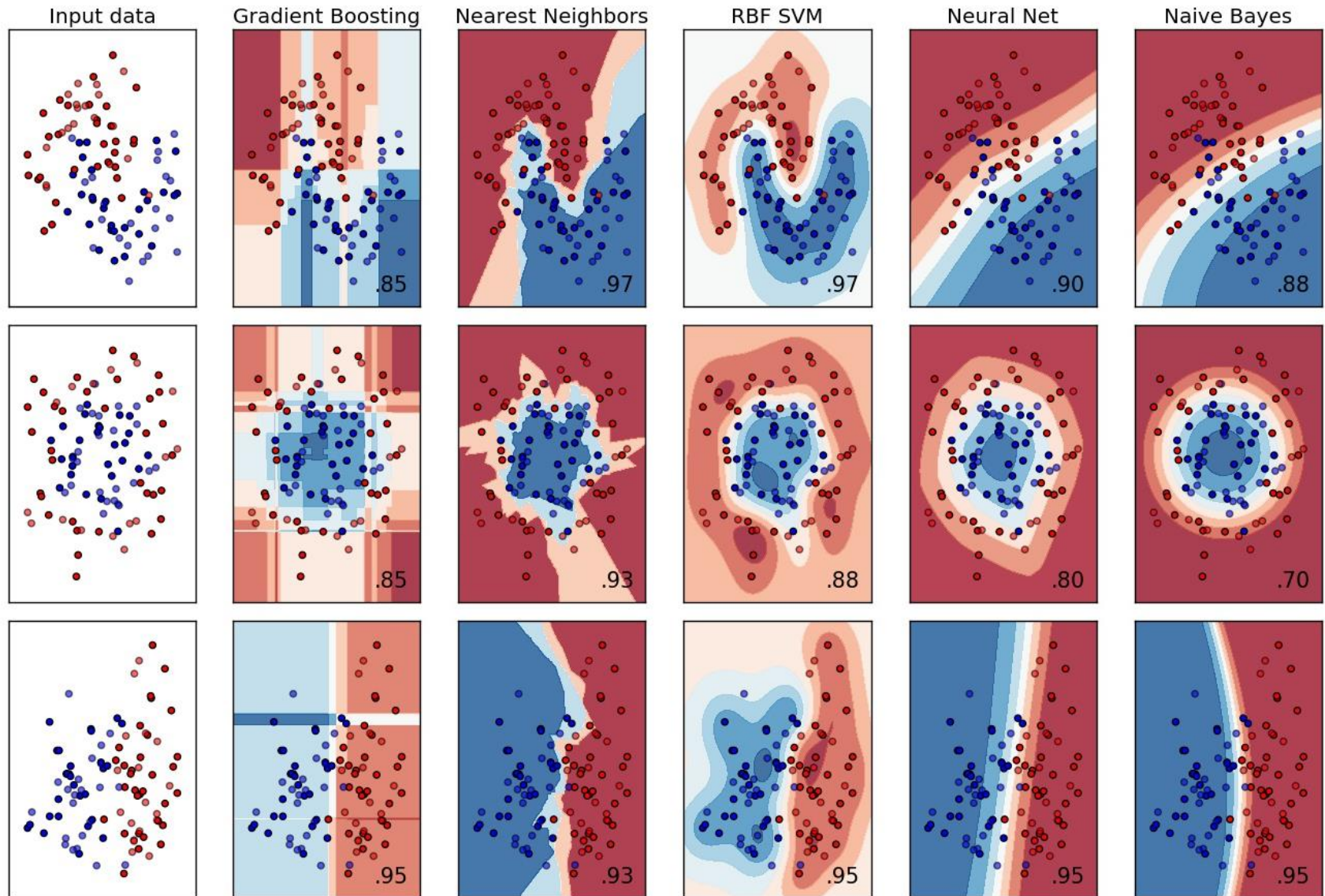
Class 2



IV. Clasificare



IV. Clasificare



IV. Clasificare

Concluzii

- No free lunch: algoritmi de machine learning sunt unelte cu avantaje și dezavantaje;
- Mai întâi trebuie încercați clasificatorii mai simpli dacă se mapează suficient de bine pe aplicație și abia apoi cei mai complicați;
- Este mai bine să avem trăsături mai inteligente și clasificatori mai simpli decât clasificatori foarte complicați și trăsături simple;
- Să se folosească clasificatori mai complicați atunci când avem un set foarte divers și mare de antrenare (pentru un bun compromis bias-varianță).

IV. Clasificare

Concluzii

- Nici un clasificator nu este cel mai bun decât ceilalți: este nevoie de a face presupuneri pentru a generaliza problema.
- Sunt trei tipuri de erori, generate de:
 - Zgomot: inerente și nu care nu pot fi total înlăturate;
 - Bias: datorită presupunerilor și simplificărilor făcute;
 - Varianță: datorită inabilității de a estima perfect parametrii datorită volumului limitat al datelor de antrenare.





Întrebări?

