LAPI – Laboratorul de Analiza şi Prelucrarea Imaginilor

Universitatea POLITEHNICA din Bucureşti

Facultatea de Electronică, Telecomunicaţii şi Tehnologia Informaţiei

# Interfață Vizuală Om-Mașină
# Analiza și recunoașterea gesturilor

**Dr.ing. Ionuț Mironică**

## IV. Clasificarea gesturilor

- **Introducere – algoritmi de învăţare**
- **Învăţare supervizată**
- **Concluzii**



## IV. Clasificarea gesturilor
### Introducere - algoritmi de învăţare

**Ce este Machine learning?**

- Este foarte greu de scris un algoritm care recunoaşte un set de gesturi statice / dinamice sau care să rezolve problema de recunoaştere a feţei:
  - Nu ştim cum funcţionează creierul uman pentru a clasifica gesturile;
  - Chiar dacă am şti nu am avea idee cum sa programăm deoarece ar fi foarte complicat;
  - Ar trebui să scriem o funcţie diferită pentru fiecare gest.
- În loc să scriem programe foarte multe, putem colecta exemple care specifică fiecare gest;
- Un algoritm de învăţare va prelua aceste exemple şi va "creea" un program care va face această clasificare în mod automat;



## IV. Clasificarea gesturilor
### Introducere - algoritmi de învăţare

**Ce este Machine learning?**

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

**Învăţare supervizată**
- datele de antrenare conţin şi ieşirea dorită;

**Învăţare nesupervizată**
- datele de antrenare nu conţin ieşirea dorită (clusterizare);
- ideea de bază este de a se găsi şabloane şi pattern-uri în date care să fie evidenţiate în mod automat.

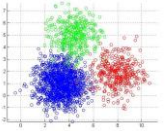
**Învăţare semi-supervizată**
- doar o parte din datele de antrenare conţin ieşirea dorită;

**Reinforcement Learning**
- se învaţă în funcţie de feedback-ul primit după ce o decizie este luată.



## IV. Clasificare
### Introducere - algoritmi de învăţare

**Învăţare nesupervizată**



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

Clustering aglomerativ
- Fiecare punct reprezintă propriul său cluster şi în mod iterativ se unesc cei mai apropiaţi centroizi;

Clustering MeanShift
- În funcţie de funcţia de densitate de probabilitate se estimează fiecare centroid.

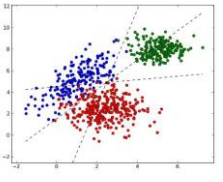


## IV. Clasificare
### Introducere - algoritmi de învăţare

**Învăţare supervizată**

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

f(  ) = "gest 1"
f(  ) = "gest 2"
f(  ) = "gest 3"



## IV. Clasificare
### Învăţare supervizată

**Model de bază**

$$y = f(\mathbf{x})$$

ieşire — Funcţia de predicţie — Trăsătură extrasă

**Antrenare:** fiind dată o mulţime de antrenare împreună cu răspunsul dorit $\{(\mathbf{x}_1,y_1), \ldots, (\mathbf{x}_N,y_N)\}$, se estimează predicţia funcţiei f prin minimizarea erorii de predicţie pe mulţimea de antrenare;

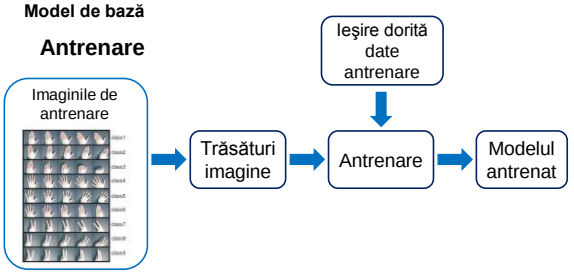
**Testare:** se aplică funcţia f pe un exemplu de test **x**(care nu a fost folosit în procesul de antrenare) şi prezintă ieşirea funcţiei $y = f(\mathbf{x})$.



## IV. Clasificare
### Învăţare supervizată

**Model de bază**

**Antrenare**

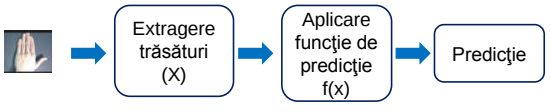





## IV. Clasificare
### Învăţare supervizată

**Model de bază**

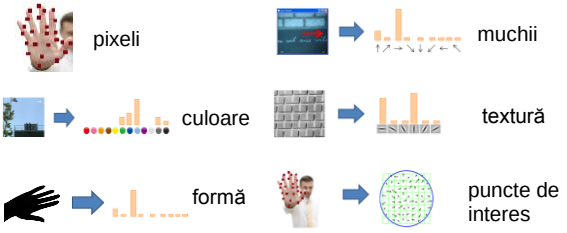
**Testare**






## IV. Clasificare
### Învăţare supervizată

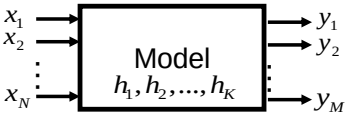
**Trăsături extrase**






## IV. Clasificare
### Învăţare supervizată

**Învăţare supervizată – schemă de bază**




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

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

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



## IV. Clasificare
### Învăţare supervizată

**Algoritmi existenţi**

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

Etc.

**Care este cel mai bun algoritm?**

## IV. Clasificare
### Învăţare supervizată

**Teorema "No free lunch"**






## IV. Clasificare
### Învăţare supervizată

**Puterea de generalizare**

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

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

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



## IV. Clasificare
### Învăţare supervizată

**Puterea de generalizare**

**Underfitting:** modelul este prea "simplu" pentru a reprezenta toate caracteristicile relevante ale claselor:

- Bias ridicat şi variaţie scăzută;
- Eroare de antrenare ridicată şi eroare de testare scăzută.

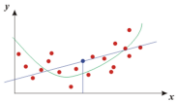
**Overfitting:** modelul este prea "complex" şi modelează caracteristici irelevante (zgomot):

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

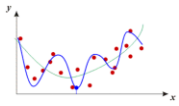


## IV. Clasificare
### Învăţare supervizată

**Compromisul dintre bias şi varianţă**




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




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



## IV. Clasificare
### Învăţare supervizată

**Compromisul dintre bias şi variaţie**

Eroare = zgomot + bias + varianţă

Erori care nu pot fi eliminate

Erori datorate presupunerilor false

Erori datorate variaţiei elementelor de intrare



## IV. Clasificare
### Învăţare supervizată

**Bias**

- Bias-ul reprezintă presupunerile efectuate de model pentru a face funcția de optimizat mai ușor de învățat;
- În general, algoritmii simpli au un bias ridicat, aceștia fiind ușor de înțeles dar mai puțini 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
## Învăţare 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.

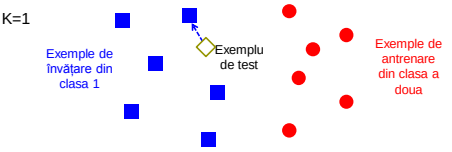
Exemple de algoritmi cu varianță scăzută:
- regresie liniară și logistică.

Exemple de algoritmi cu varianță ridicată:
- Arbori de decizie (mai ales cei nefasonați), SVM, k-Nearest Neighbor.



# IV. Clasificare
## Învăţare supervizată

**K-Nearest Neighbor (Cei mai apropiaţi vecini)**




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
## Învăţare supervizată

**Compromisul dintre bias şi varianţă**

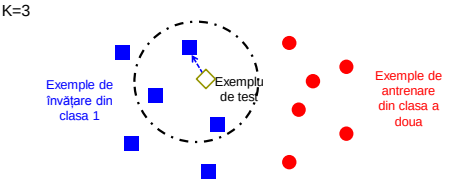



Slide: D. Hoiem



# IV. Clasificare
## Învăţare supervizată

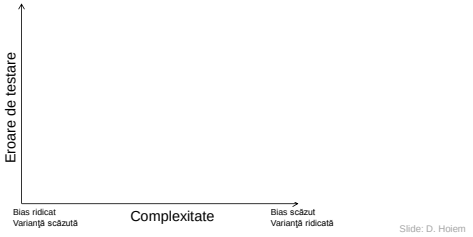
**K-Nearest Neighbor (Cei mai apropiaţi vecini)**






# IV. Clasificare
## Învăţare supervizată

**Compromisul dintre bias şi varianţă**

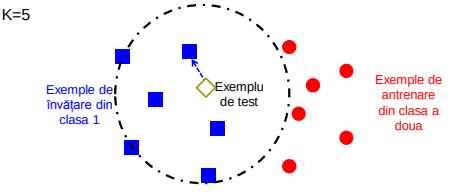



Slide: D. Hoiem



# IV. Clasificare
## Învăţare supervizată

**K-Nearest Neighbor (Cei mai apropiaţi vecini)**



## IV. Clasificare
### Învăţare supervizată

**K-Nearest Neighbor (Cei mai apropiaţi vecini)**

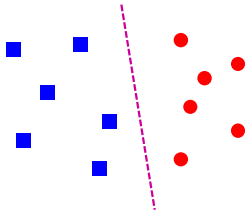
**Avantaje**
- Simplu (uşor de implementat pentru o primă încercare);
- Erori mari de clasificare (nu are rezultate bune pentru probleme complexe);

**Dezavantaje**
- Ocupă foarte multă memorie (se reţine 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
### Învăţare supervizată

**Clasificator liniar**



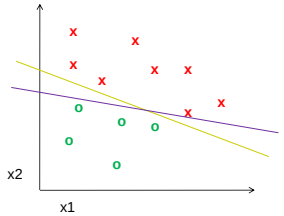
Caută o funcție liniară care separă clasele

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



## IV. Clasificare
### Învăţare supervizată

**SVM liniar**



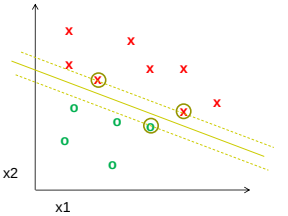

- Caută o funcţie ce separă două clase în mod optim:

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



## IV. Clasificare
### Învăţare supervizată

**SVM liniar**




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

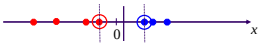
$f(\mathbf{x}) = \text{sgn}(\mathbf{w} \cdot \mathbf{x} + b)$



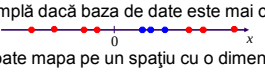
## IV. Clasificare
### Învăţare supervizată

**SVM liniar**

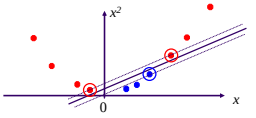
- Dacă clasele sunt liniar separabile, SVM liniar funcţionea:




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




- Soluţie – se poate mapa pe un spaţiu cu o dimensiune mai mare:

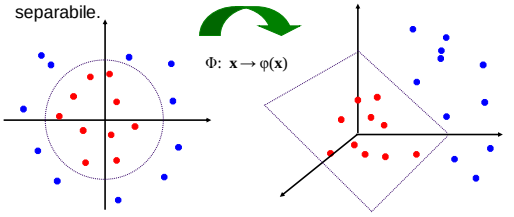





## IV. Clasificare
### Învăţare supervizată

**SVM neliniar**

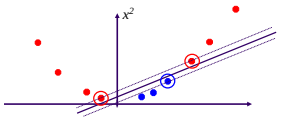
- Idee de bază: spaţiul iniţial poate fi mapat către un spaţiu multidimensional în care trăsăturile de antrenare devin liniar separabile.



## IV. Clasificare
### Învăţare supervizată

**SVM neliniar - exemplu**

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




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



## IV. Clasificare
### Învăţare supervizată

**SVM neliniar – exemple de nuclee**

- Nucleul de intersecţie de histogrmă:

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

- Nucleu gausian:

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



## IV. Clasificare
### Învăţare supervizată

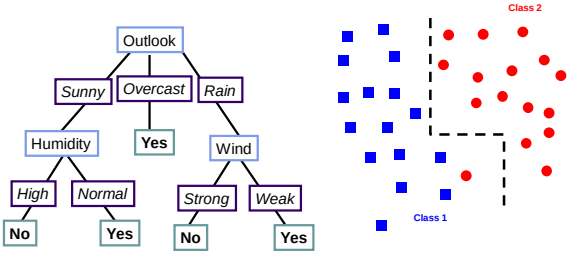
**SVM multiclasă**

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



## IV. Clasificare
### Învăţare supervizată

**Arbori de decizie**






## IV. Clasificare
### Învăţare supervizată

**SVM multiclasă**

- **Unu vs. alţii**
  - Antrenare: se antrenează câte un model SVM pentru fiecare clasă vs celalalte;
  - Testare: se aplică fiecare model SVM şi 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.



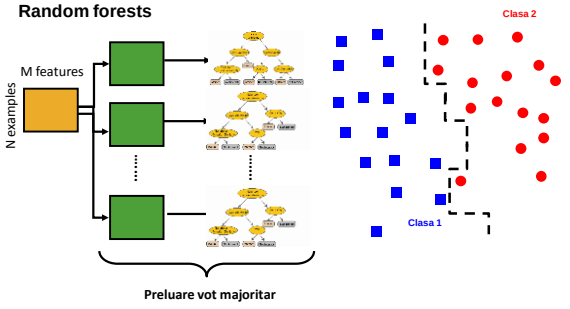
## IV. Clasificare
### Învăţare supervizată

**SVM**

**Avantaje**
- Multe implementări:
  OpenCV, LibSVM, http://www.kernel-machines.org/software etc,
- SVM nonlinear are o putere de clasificare şi generalizare foarte mare, fiind foarte flexibil;
- SVM lucrează bine chiar şi 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 computaţional (nu pot fi folosite pentru aplicaţii large-scale).

## IV. Clasificare
### Învăţare supervizată
**Random forests**






## IV. Clasificare
### Învăţare supervizată
**Naive Bayes**

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

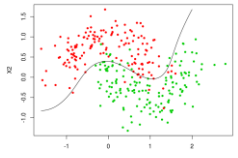
$$P(Y|X_1,\ldots,X_n) = \frac{P(X_1,\ldots,X_n|Y)P(Y)}{P(X_1,\ldots,X_n)}$$

Likelihood, Prior, Normalization Constant

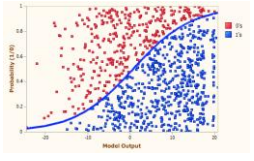


## IV. Clasificare
### Învăţare supervizată
**Regresie logistică**

Utilizează funcţii matematice simple între cele două clase pentru a creea granițe între acestea.



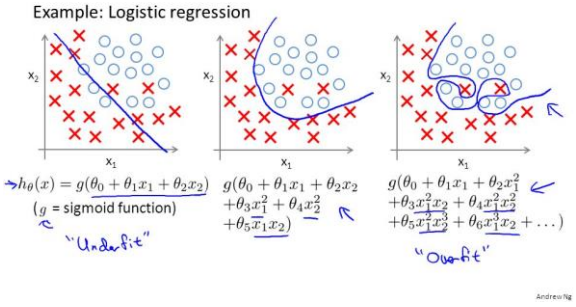
Funcţii polinomiale



Funcția logistică

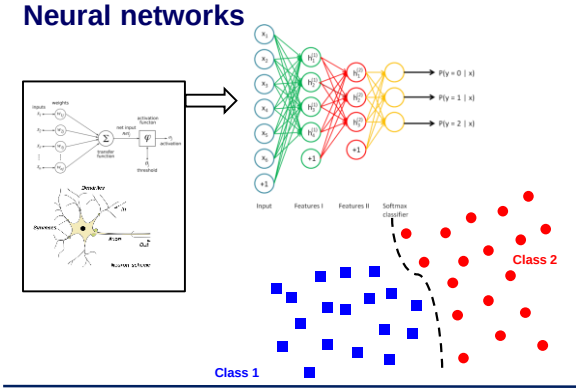


## Logistic regression



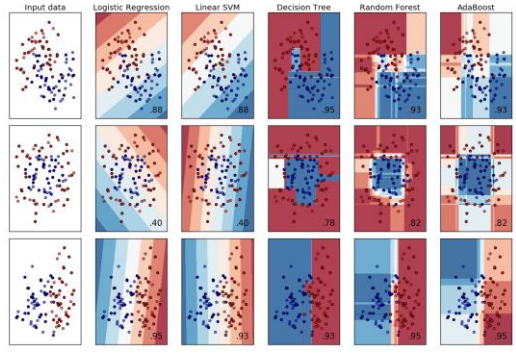



## Neural networks

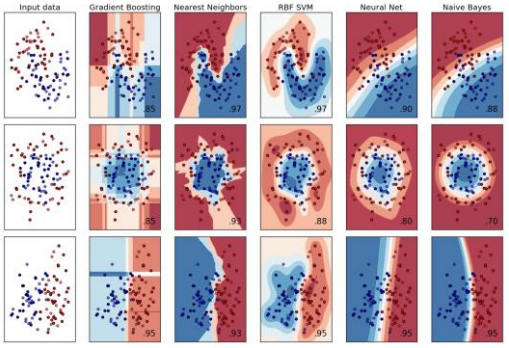





## IV. Clasificare



## IV. Clasificare






## IV. Clasificare
### Concluzii

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



## 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?





