Introducere în învățare supervizată

ML@AIMAS 2024

Șl. dr. ing. Mihai Nan
Prof. dr. ing. Adina Magda Florea

Departamentul de Calculatoare din UPB

Cuprins

Definiții

- Învățarea este procesul prin care un sistem își îmbunătățește performanțele (***Herbert Simon***)
- Învățarea este achiziția cunoștințelor explicite
- Învățarea este achiziția deprinderilor de rezolvare a problemelor
- Învățarea este formarea teoriilor, formarea ipotezelor și inferența inductivă
- Machine Learning (**ML**) – observă cantități mari de date și caută șabloane pentru predicție

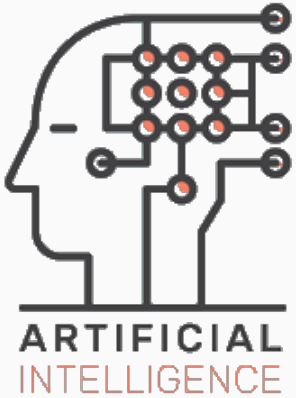
Obiectivele algoritmilor bazați pe ML

1. să rezolve probleme complicate pentru care un algoritm clasic nu ar putea obține rezultate;
2. să realizeze predicții cât mai exacte (în special, pentru date care nu au mai fost analizate);
3. să gestioneze o varietate de probleme de învățare.

Definiție

Inteligența Artificială este orice tehnologie proiectată să imite, într-un fel sau altul, modul în care funcționează un om.

Inteligența artificială



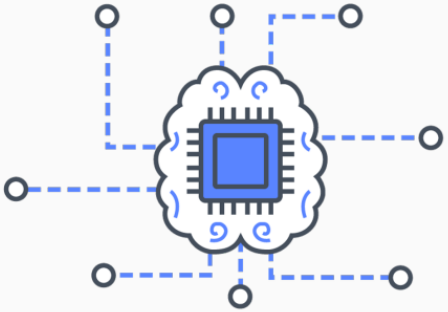
> *Este vorba despre inteligență artificială atunci când o mașină se comportă într-un mod care ar putea fi considerat inteligent, dacă ar fi vorba de un om.*
>
> *John McCarthy*

Principalele calități pe care ar trebui să le îndeplinească un sistem pentru a putea fi considerat inteligent:

- capacitatea de a raționa;
- abilitatea de a descoperi sensul într-o situație dată;
- abilitatea de a generaliza, plecând de la un caz particular;
- capacitatea de a învăța din experiențe.

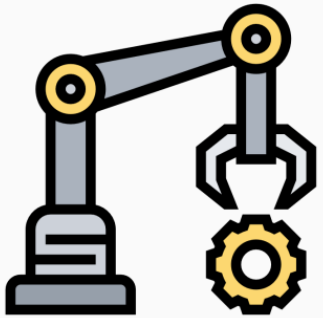
Testul Turing elaborat în anul 1950 poate fi folosit pentru a testa capacitatea unei mașini de a arăta un comportament inteligent care să nu poată fi deosebit de cel uman.
Algoritmii de Inteligență Artificială combină noțiuni din multe domenii, cum ar fi: matematică, algoritmică, biologie.

Domeniile Inteligenței Artificiale



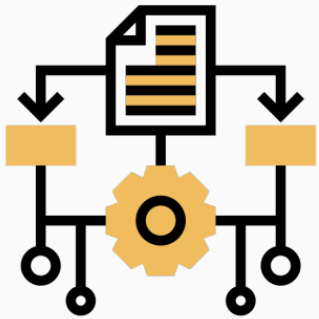
Machine Learning

Neural Networks



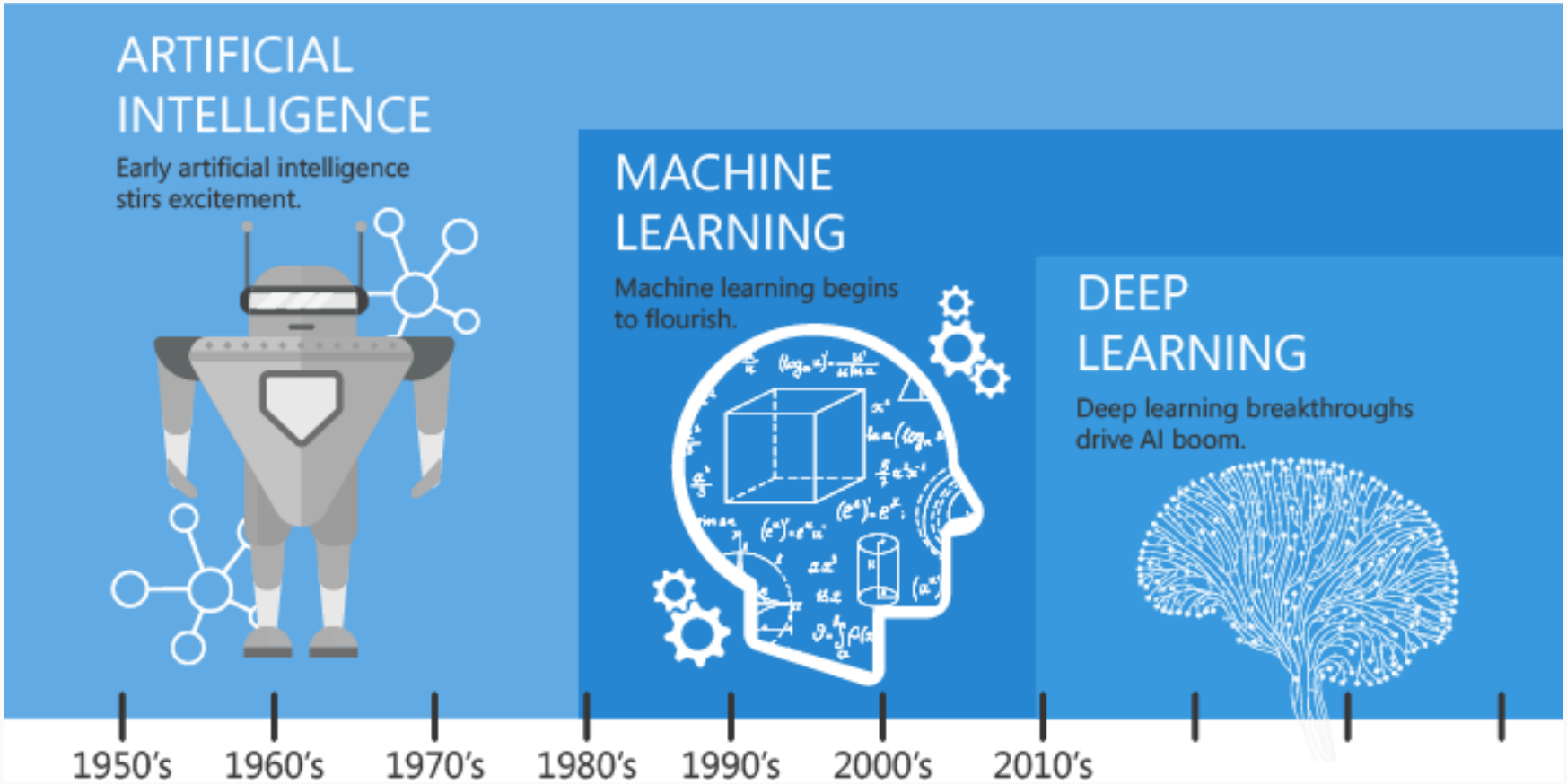
Robotics

Expert Systems



Fuzzy Logic

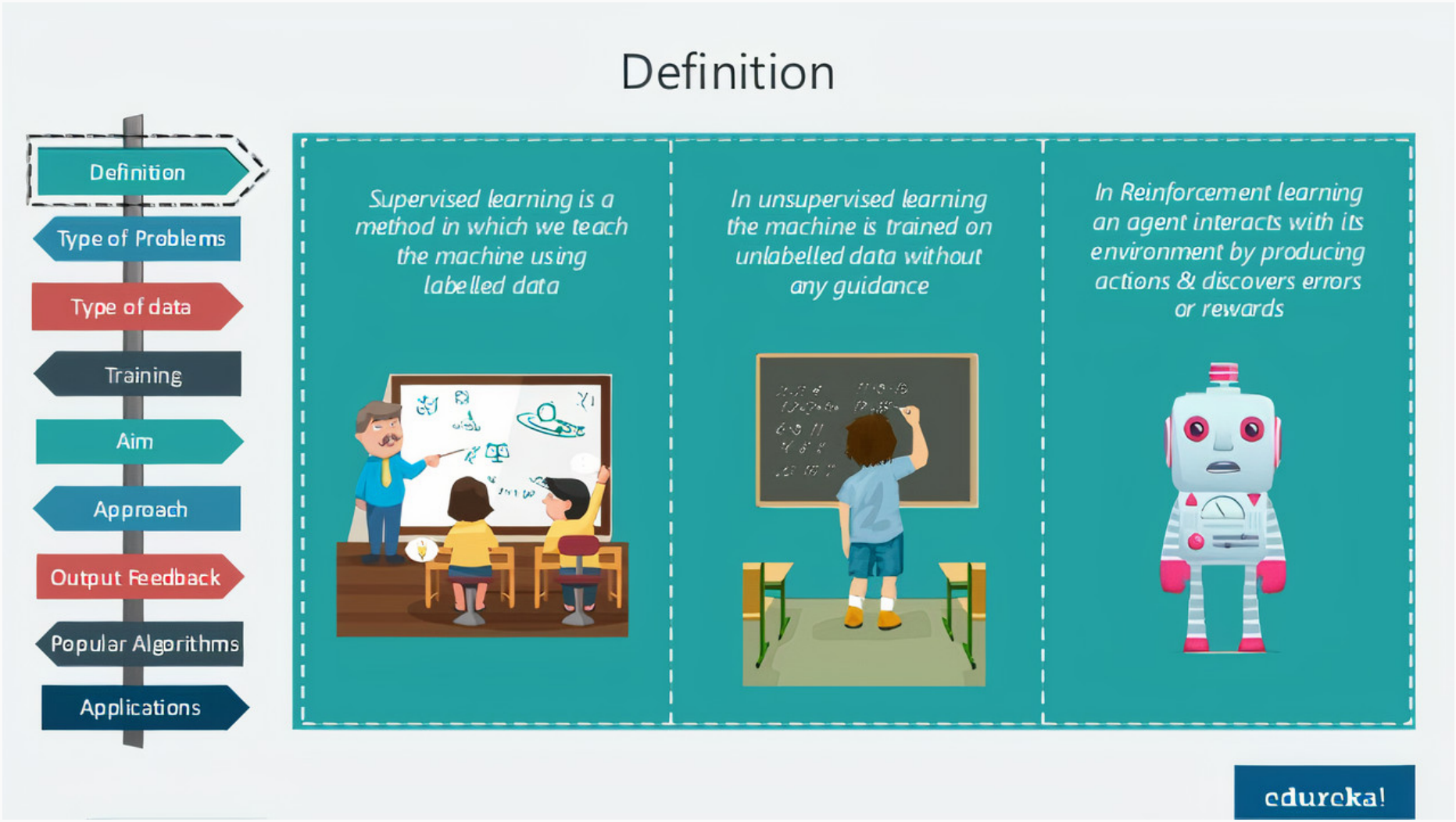
Natural Language Processing



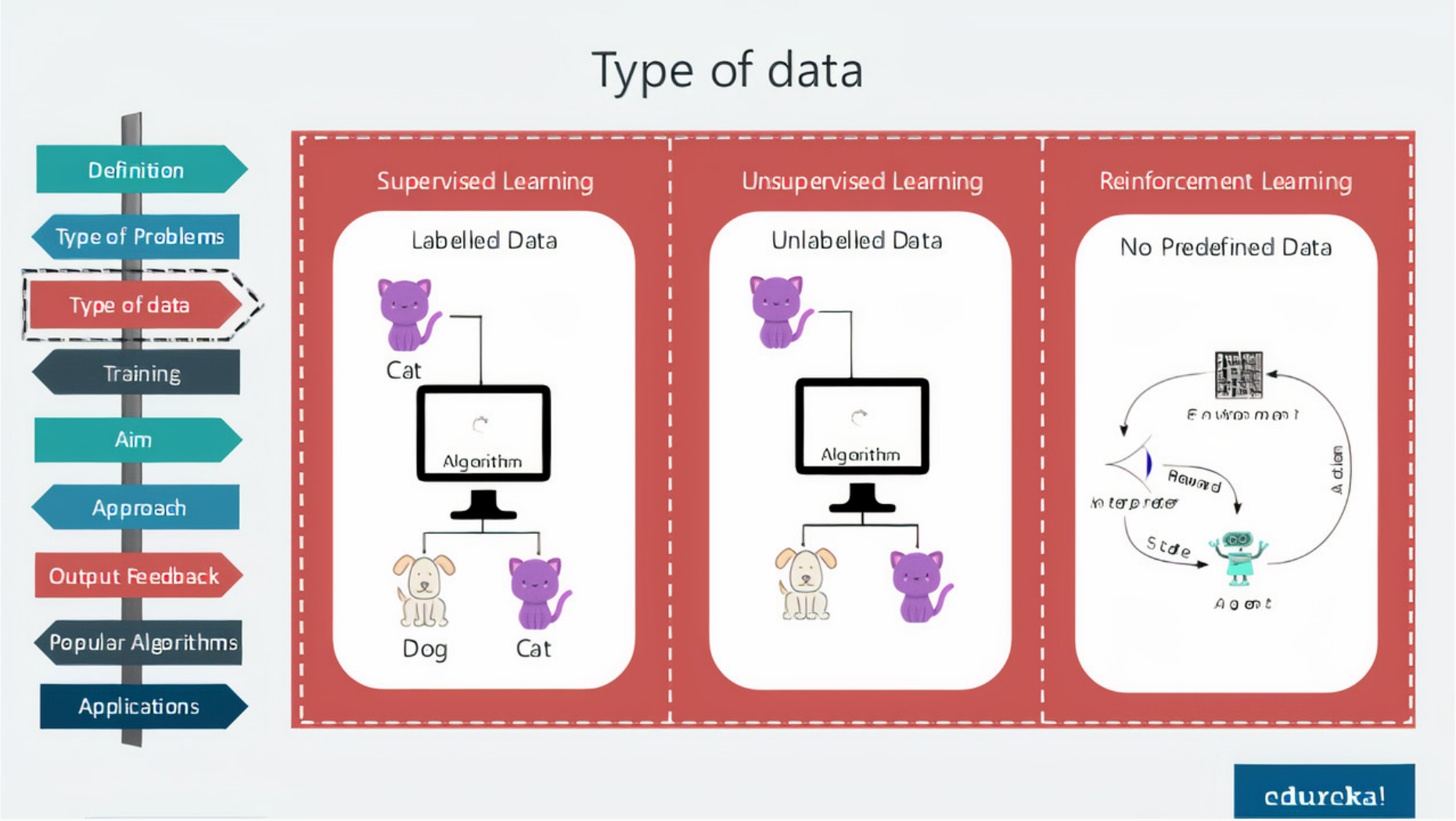
- Task-uri definite prin exemple;
- Relații / corelații în cantități mari de date;
- Mediu în schimbare și date diferite (cu zgomot);
- Cantitate de cunoștințe prea mare pentru a fi reprezentate explicit.

- **Exemplu** – un obiect sau o instanță din datele utilizate;
- **Caracteristici (Features)** – mulțimea de atribute, adesea reprezentat ca un vector, asociat unui exemplu;
- **Etichete (Labels)** – o valoare asociată unui exemplu (poate fi o singură valoare, o mulțime de valori, o categorie);
- **Mulțime de învățare (Training set)** – date utilizate în etapa de antrenare a algoritmului;
- **Mulțime de testare (Testing set)** – date utilizate exclusiv pentru testarea algoritmului.
- **Scenariile standard de învățare**
  1. **supervised learning** – date de antrenament etichetate
  2. **unsupervised learning** – fără date etichetate
  3. **semi-supervised learning** – date de antrenament etichetate + date neetichetate
  4. **transductive learning** – date de antrenament etichetate + date de testare fără etichetă;
  5. **reinforcement learning**

Învățare Automată – Definiții și Terminologie

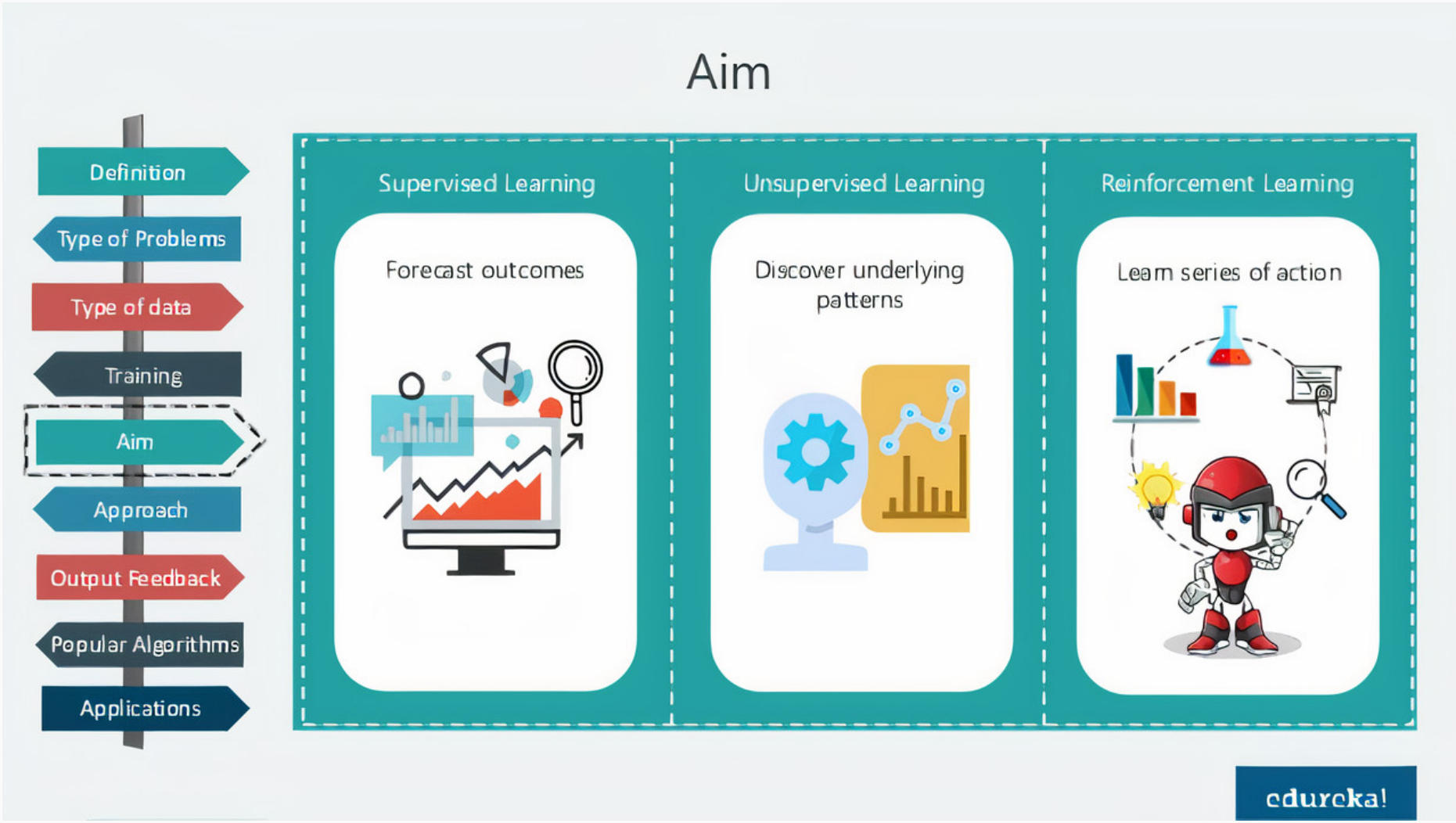


Învățare Automată – Definiții și Terminologie

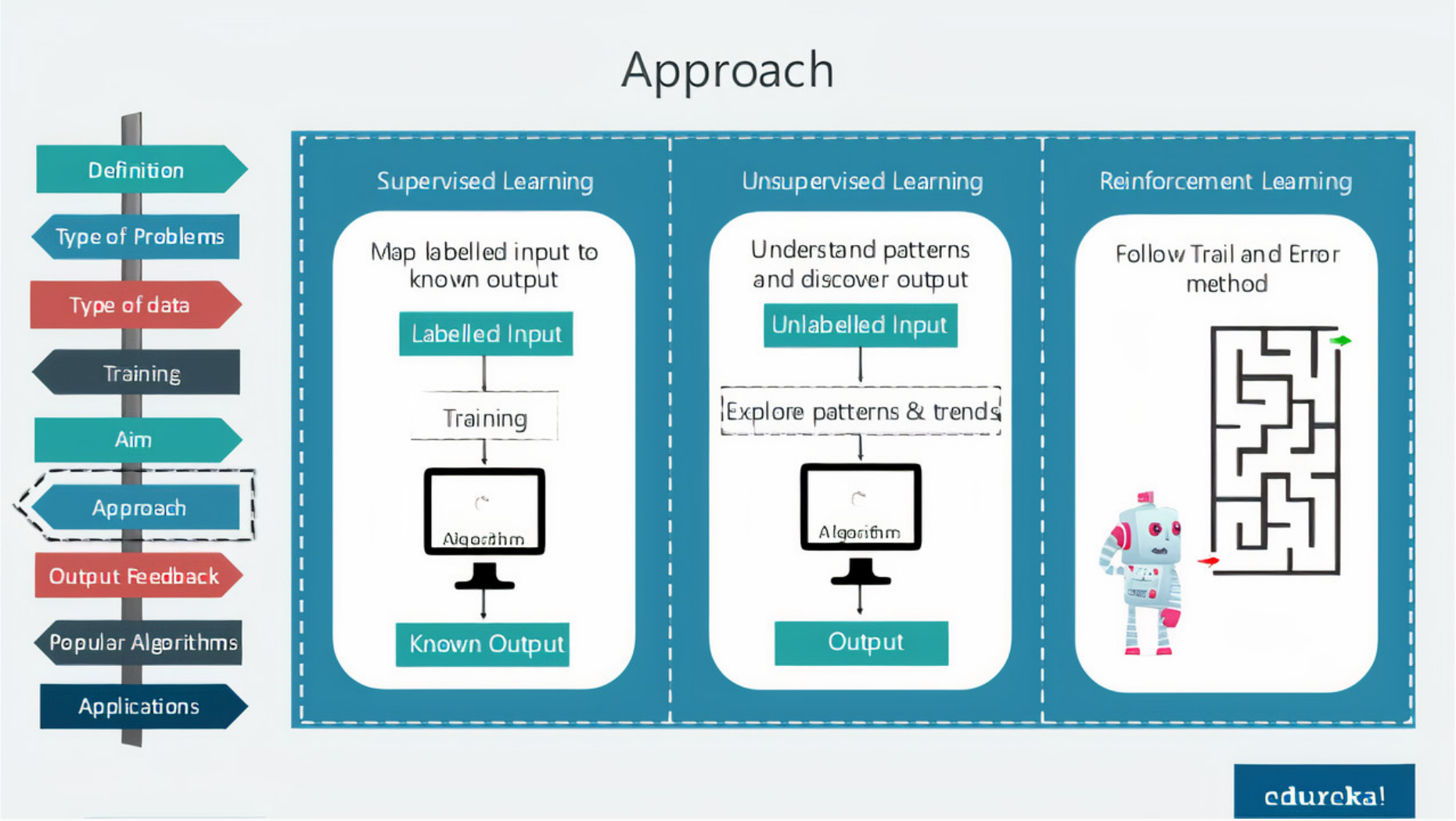


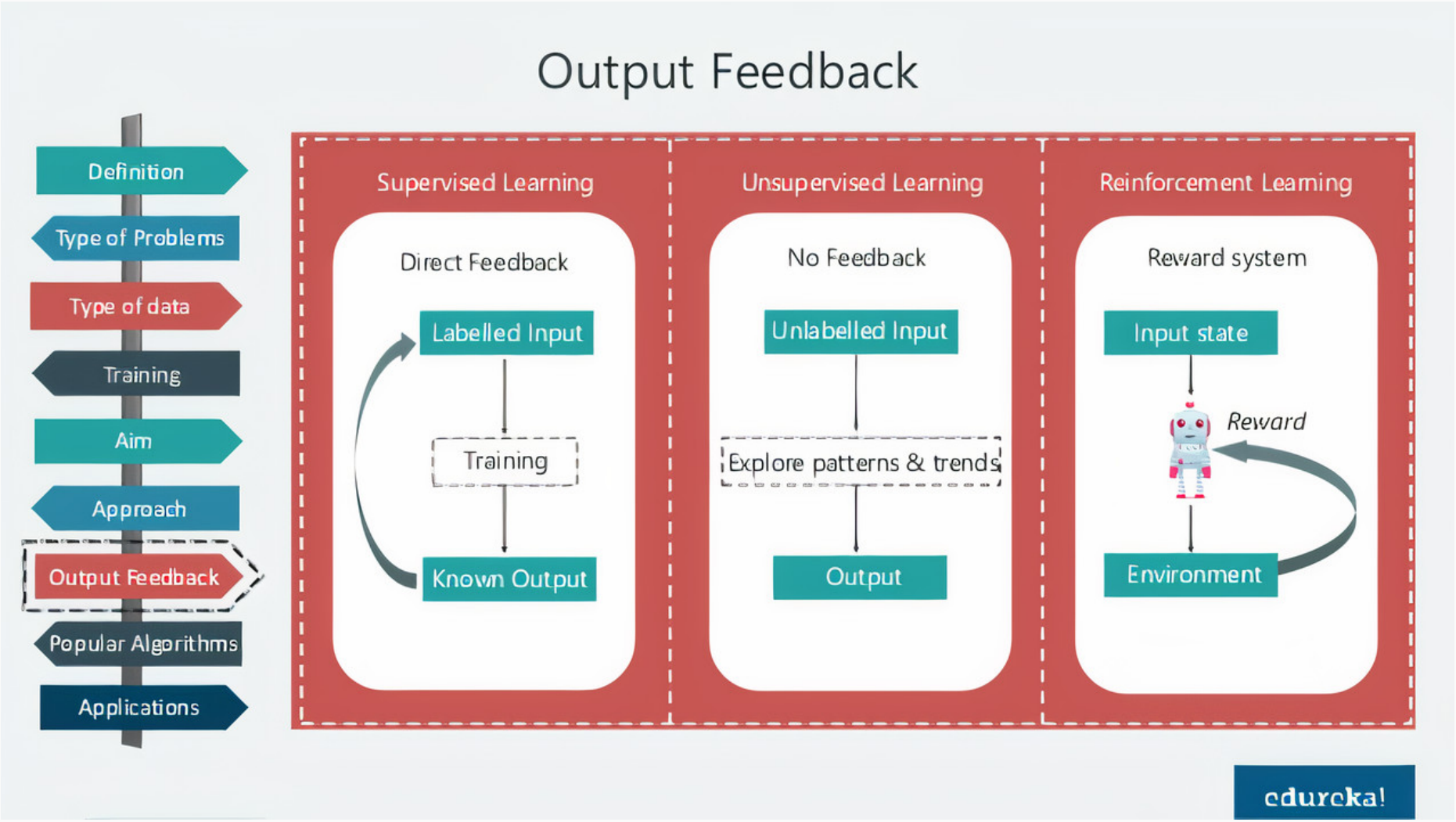
Învățare Automată – Definiții și Terminologie

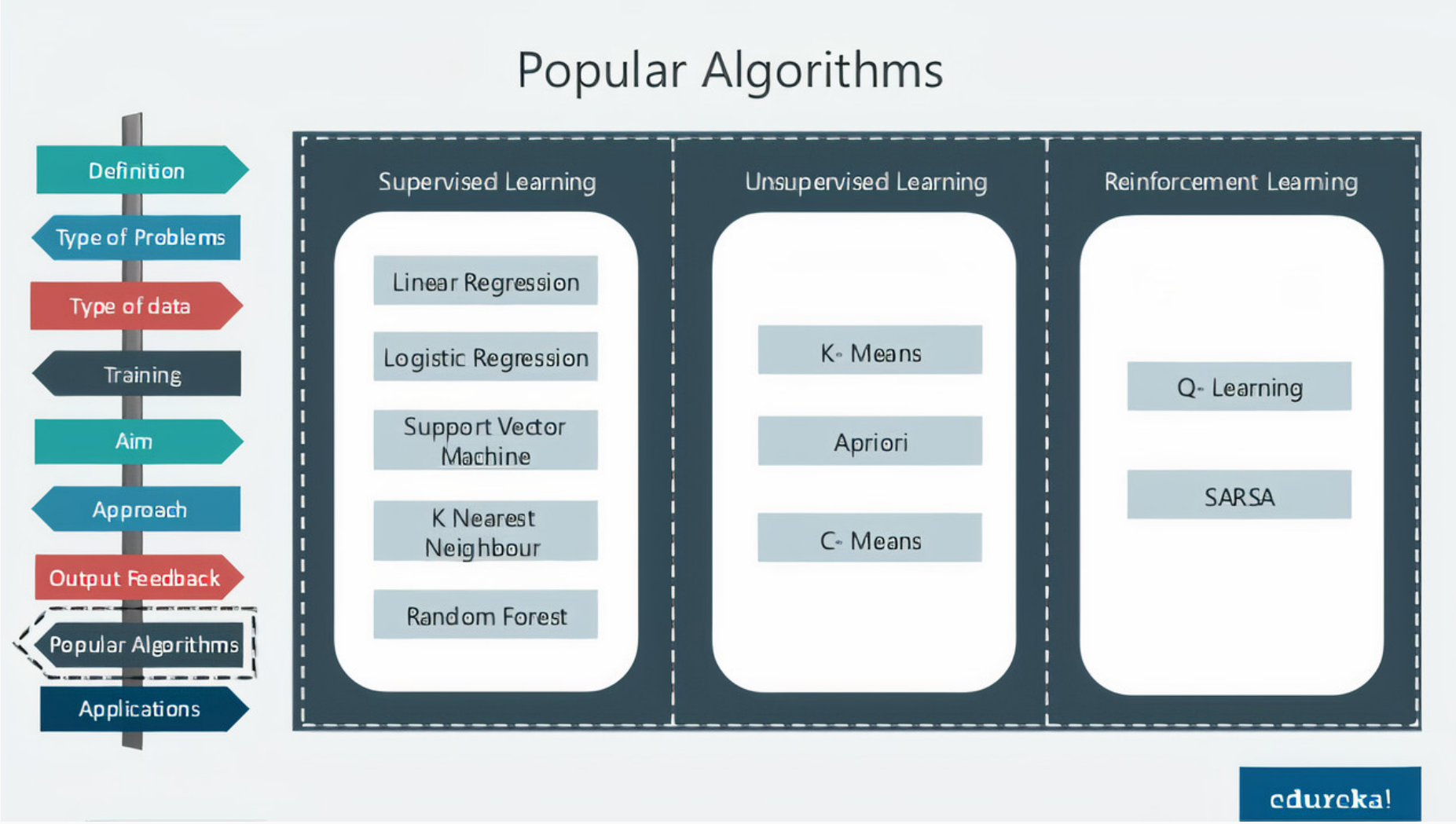
Învățare Automată – Definiții și Terminologie



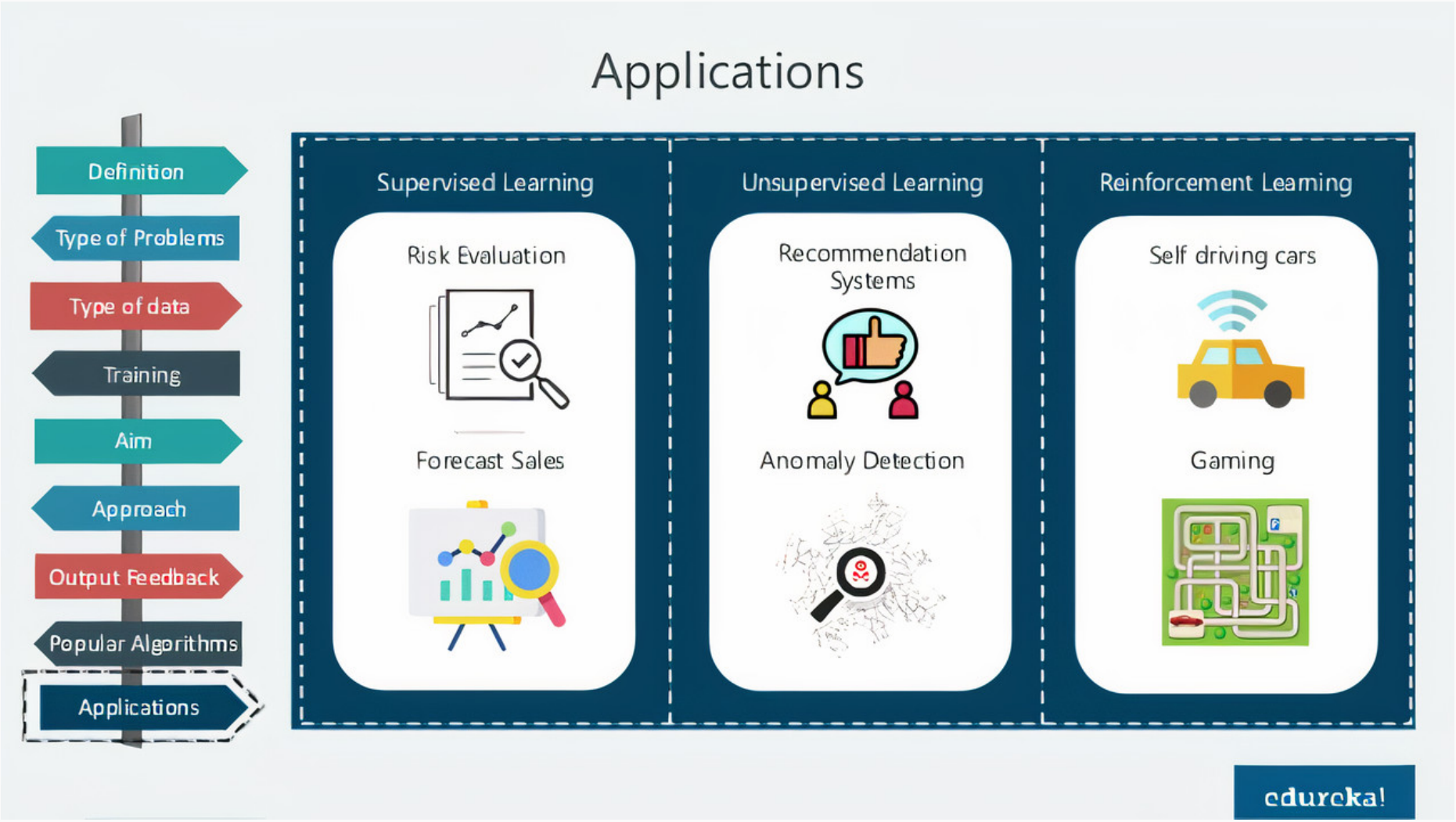
Învățare Automată – Definiții și Terminologie




Output Feedback
Definition
Type of Problems
Type of data
Training
Aim
Approach
Output Feedback
Popular Algorithms
Applications
Supervised Learning
Direct Feedback
Labelled Input
Training
Known Output
Unsupervised Learning
No Feedback
Unlabelled Input
Explore patterns & trends
Output
Reinforcement Learning
Reward system
Input state
Reward
Environment
edureka!


Popular Algorithms
Definition
Type of Problems
Type of data
Training
Aim
Approach
Output Feedback
Popular Algorithms
Applications
Supervised Learning
Linear Regression
Logistic Regression
Support Vector Machine
K Nearest Neighbour
Random Forest
Unsupervised Learning
K- Means
Apriori
C- Means
Reinforcement Learning
Q- Learning
SARSA
edureka!

Învățare Automată – Definiții și Terminologie



- Fie $\mathbf{T} = \{\mathbf{X_1}, \mathbf{X_2}, \ldots, \mathbf{X_m}\}$ o mulțime de învățare. $\mathbf{X_i}$ este un exemplu și este de forma:

$$\mathbf{X_i} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

  unde x_i reprezintă atribute ce pot fi: valori reale, valori numerice întregi, valori simbolice, valori booleene
- Din moment ce discutăm despre **învățare supervizată**, cunoaștem $f(\mathbf{X_1}), f(\mathbf{X_2}), \ldots, f(\mathbf{X_m})$

Scopul

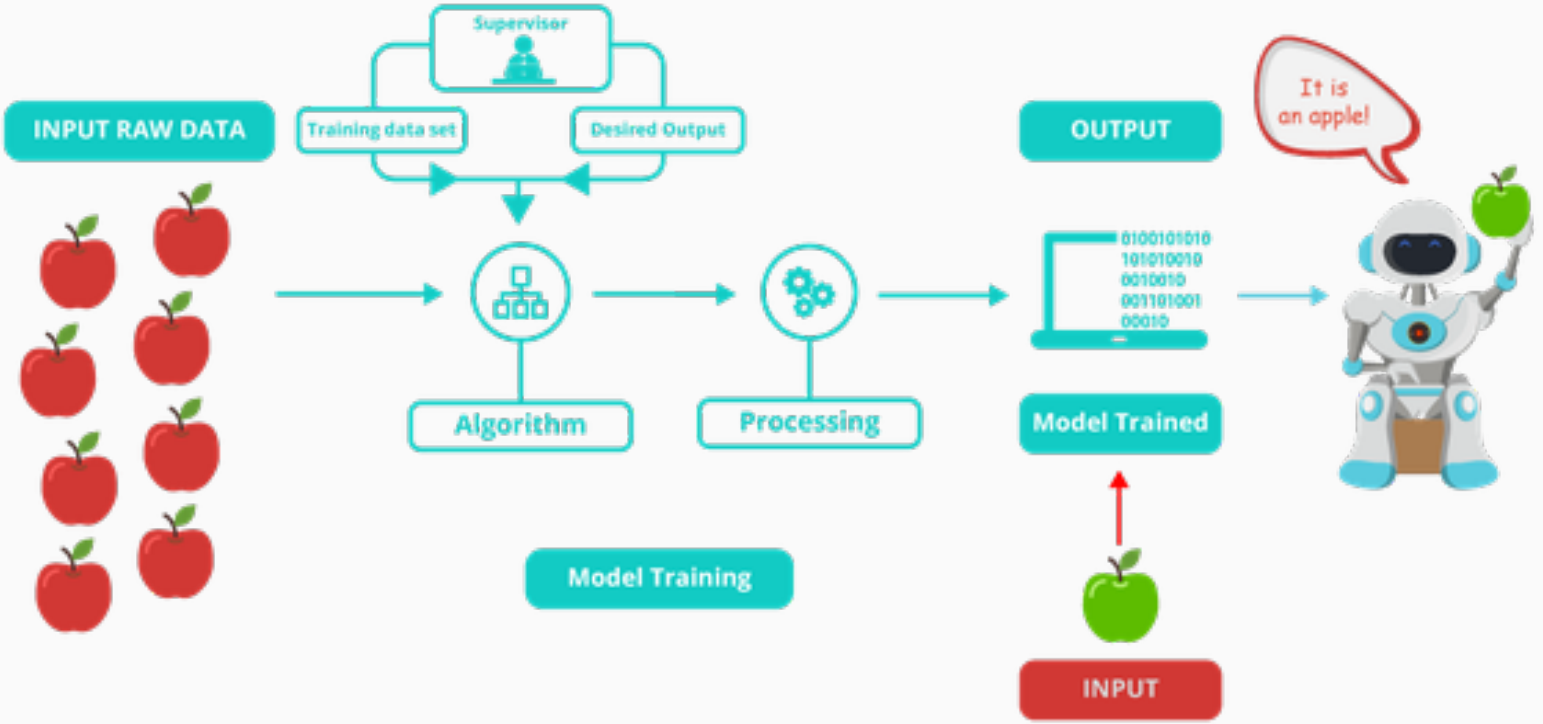
- Găsim h astfel încât:

$$h(\mathbf{X_i}) = f(\mathbf{X_i}), i = 1, m \Rightarrow h(\mathbf{X_i}) = f(\mathbf{X_i}), \forall i$$

Tipuri de probleme

1. Clasificare – f are ca rezultate valori discrete (grupează exemple)
2. Regresie – f are ca rezultate valori continue (estimează sau prezice valori)

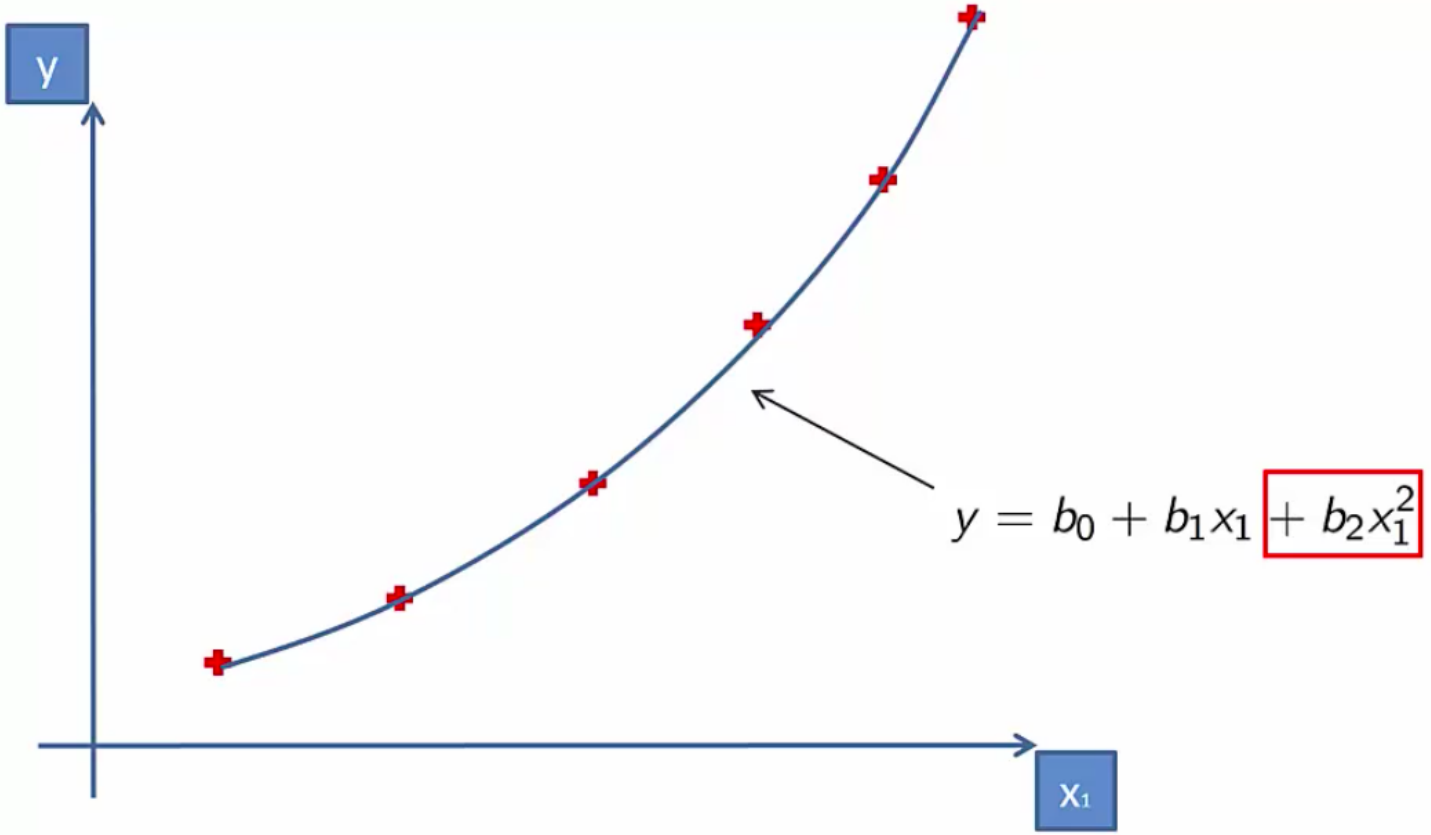
Învățare supervizată – Clasificare



Important

În timpul antrenării se folosește o funcție de eroare care testează corectitudinea clasei prezise.

Învățare supervizată – Regresie



- Fie $\mathbf{T} = \{\mathbf{X_1}, \mathbf{X_2}, \ldots, \mathbf{X_m}\}$ o mulțime de învățare. $\mathbf{X_i}$ este un exemplu și este de forma:

$$\mathbf{X_i} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

  unde x_i reprezintă atribute ce pot fi: valori reale, valori numerice întregi, valori simbolice, valori booleene
- Din moment ce discutăm despre **învățare nesupervizată**, **NU** cunoaștem $f(\mathbf{X_1}), f(\mathbf{X_2}), \ldots, f(\mathbf{X_m})$

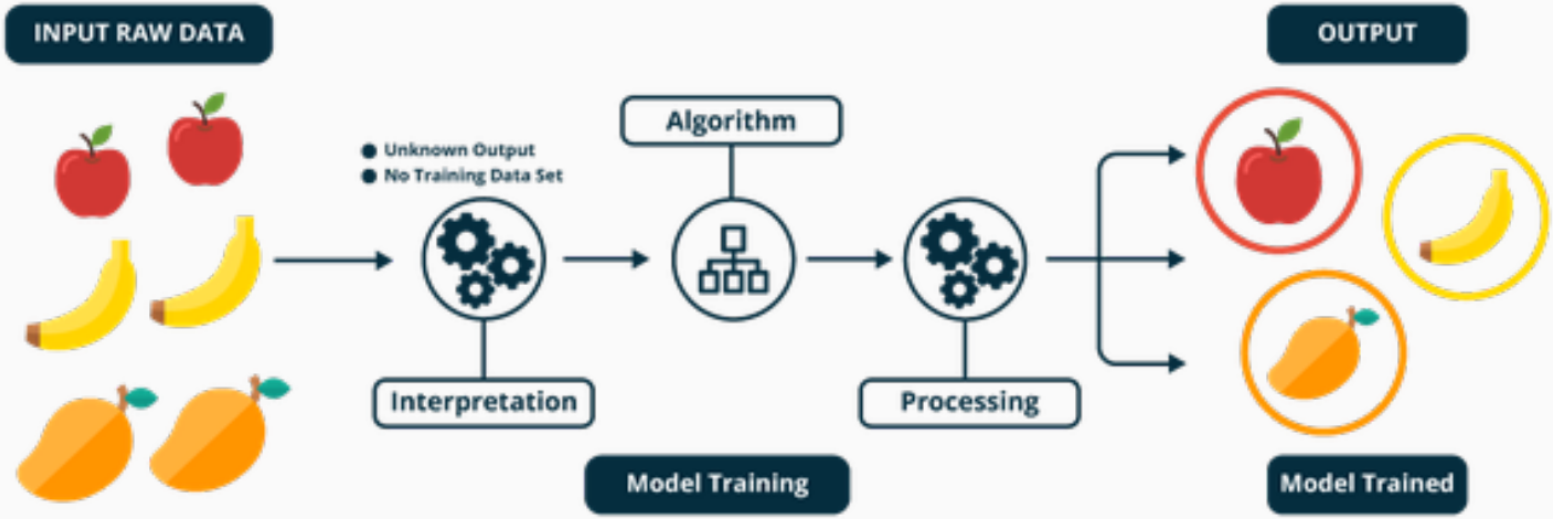
Scopul

- Gruparea datelor în categorii în funcție de similaritatea dintre ele

Tipuri de probleme

1. Identificarea unor grupuri (Clusterizare)
2. Reducerea dimensiunii
3. Modelarea densității datelor

Învățare nesupervizată



Important

Nu se bazează pe conceptul de funcție de eroare ci pe cel de calitate a modelului extras din date, care trebuie maximizată prin procesul de antrenare.

Metode de învățare nesupervizată

1. Clustering
   - K-means clustering (grupare), K-means ierarhic (grupare ierarhică)
2. Rețele neurale
   - Rețele neurale cu auto-organizare
   - Autoencoders
   - Deep belief networks
3. Expectation maximization (EM)
   - HMM – algoritmul Baum-Welch
   - Retele Bayesiene – estimarea parametrilor in cazul datelor incomplete
   - Soft clustering – clasele se pot suprapune
4. Principal Component Analysis (PCA)

Învățare prin recompensă

- Fie $\mathbf{T} = \{\mathbf{X_1}, \mathbf{X_2}, \ldots, \mathbf{X_m}\}$ o mulțime de învățare. $\mathbf{X_i}$ este un exemplu și este de forma:

$$\mathbf{X_i} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

  unde x_i reprezintă atribute ce pot fi: valori reale, valori numerice întregi, valori simbolice, valori booleene
- Se cunosc recompensele pentru valorile $h(\mathbf{X_1}), \ldots, h(\mathbf{X_m})$, dar **NU** cunoaștem $f(\mathbf{X_1}), f(\mathbf{X_2}), \ldots, f(\mathbf{X_m})$

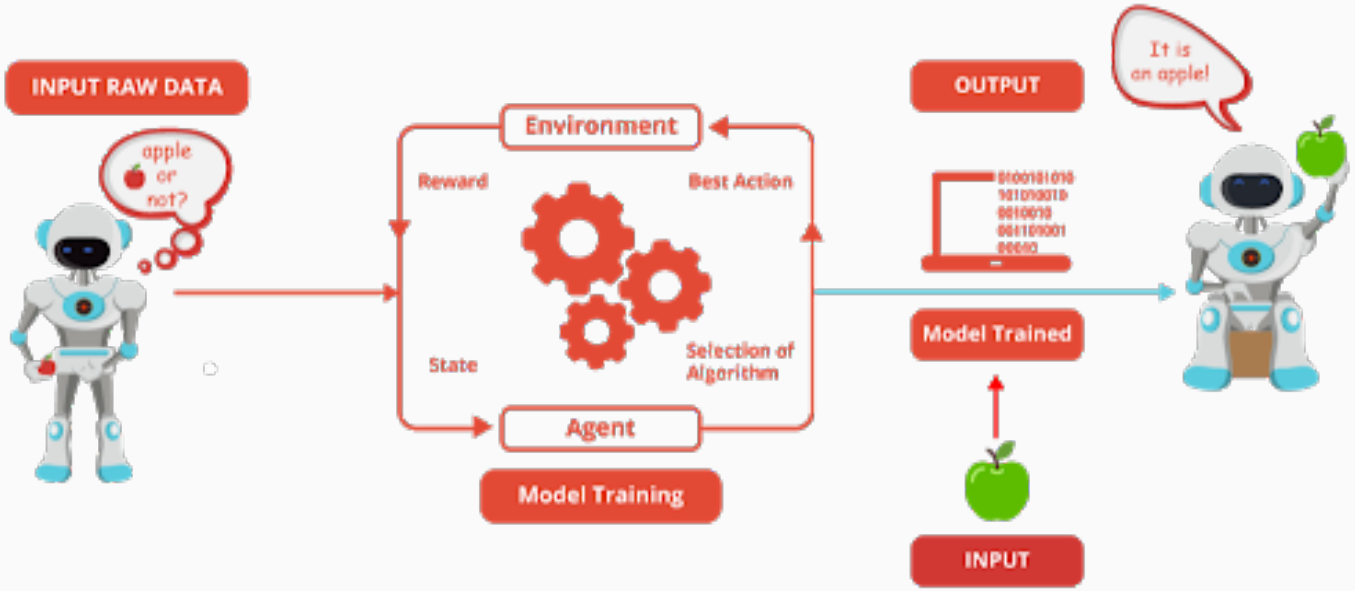
Scopul

- Învățarea, de-a lungul unei perioade, a unor acțiuni ce pot fi aplicate asupra mediul, în funcție de starea mediului, cu scopul maximizării unei recompense pe termen lung

Tipuri de probleme

1. Dresarea unui câine
2. Jocuri
3. Robotică

Învățare prin recompensă

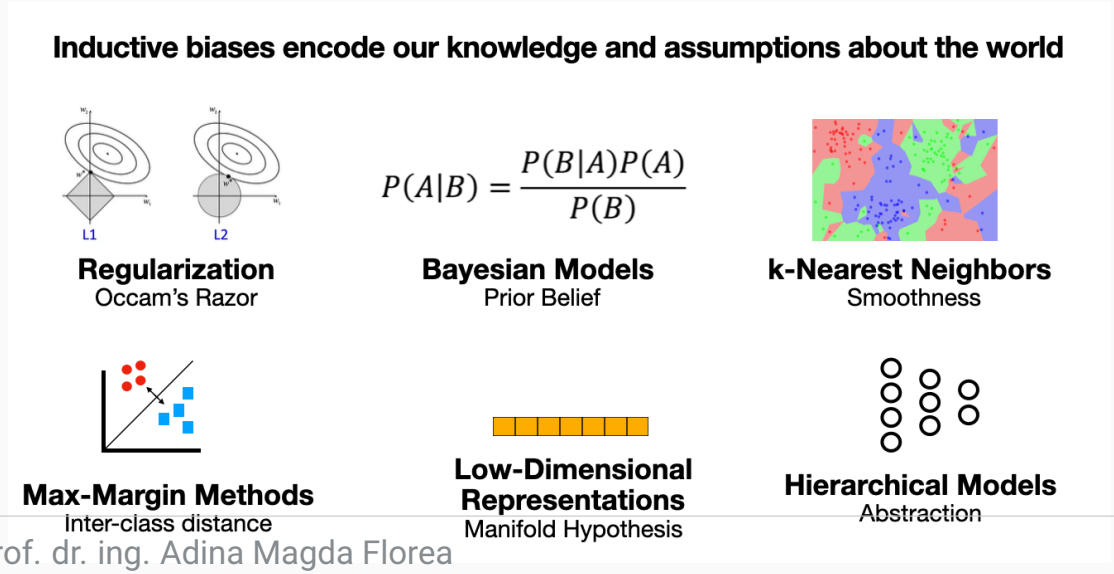


Caracteristici importante

1. Regimuri de învățare
   - Batch
   - Incremental
2. Zgomot
   - zgomot care afectează intrările (de ex. valorile atributelor)
   - zgomot care afectează ieșirile (alterare ieșiri)

Ipoteza favorită a inducției (Inductive bias)

O mulțime de presupuneri (explicite sau implicite) pe care algoritmul de învățare se bazează pentru a obține un model (a generaliza) din setul de învățare.



Inductive bias

- Spațiu de ipoteze restricționat (Restricted Bias) – Funcții liniare sau polinomiale de ordin mic
- Ipoteza preferata (Preference Bias)

  - Occam's razor
  - Independenta condițională maximă
  - Margine maximă
  - Arbore de complexitate redusă
  - Nearest neighbours

  - Recente: CNN: spatial bias, group equivariance / Deep RL agents – percepție structurată și raționament relațional

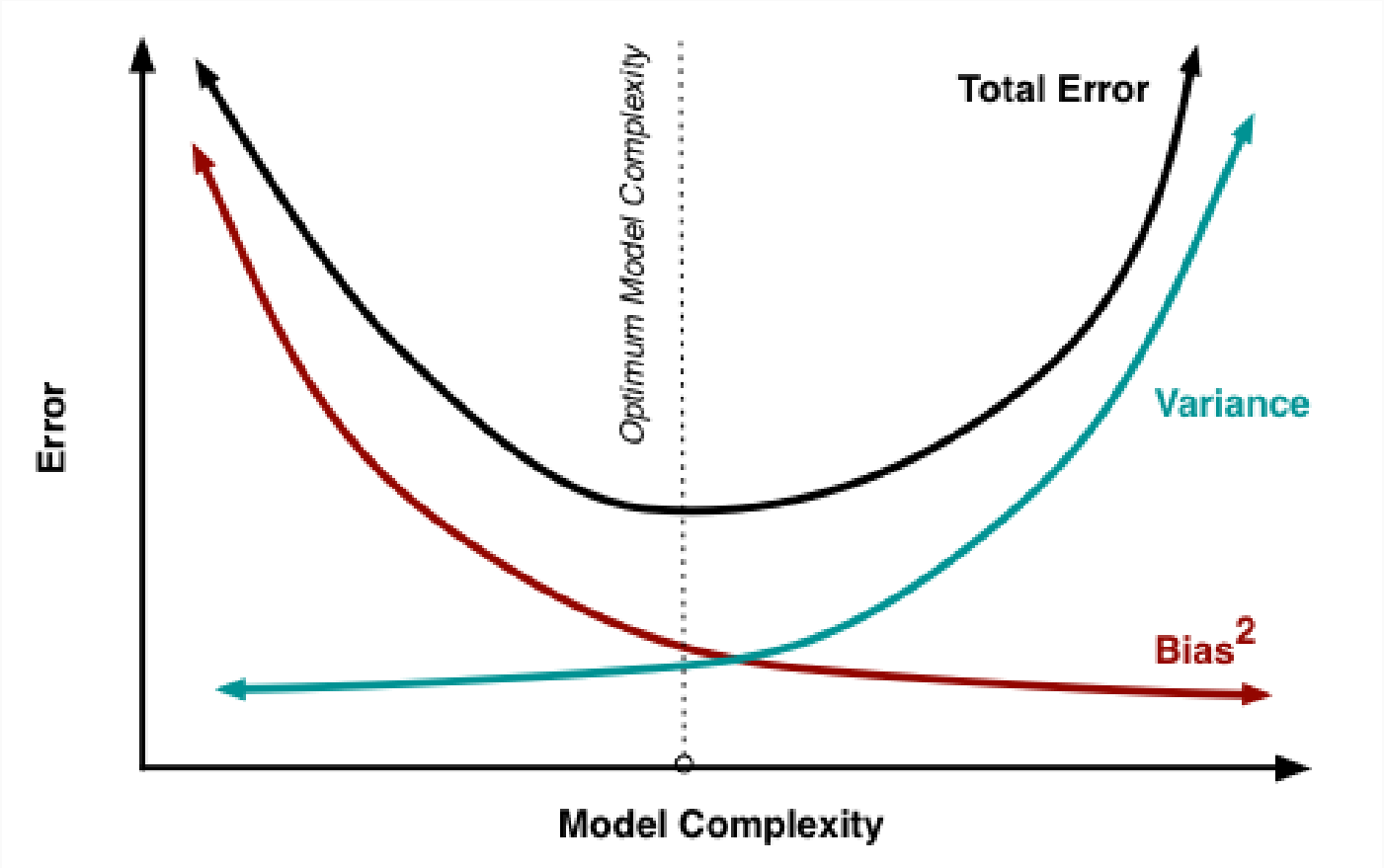
Descriere formală

- Fie un algoritm $\mathbf{L}$ ce realizează clasificare și un set de antrenare $\mathbf{D}$.
  Ipoteza favorită a inducției pentru algoritmul $\mathbf{L}$ este mulțimea minimală de aserțiuni $\mathbf{B}$ astfel încât pentru fiecare clasă c și pentru fiecare exemplu din setul de antrenare $\mathbf{D}$ etichetat cu clasa c ($\mathbf{D_c}$):

$$\forall \mathbf{X_i} \in \mathbf{X}.\ [\mathbf{B} \wedge \mathbf{D_c X_i}] \vdash \mathbf{L}(\mathbf{X_i}, \mathbf{D_c})$$(implicație logică)

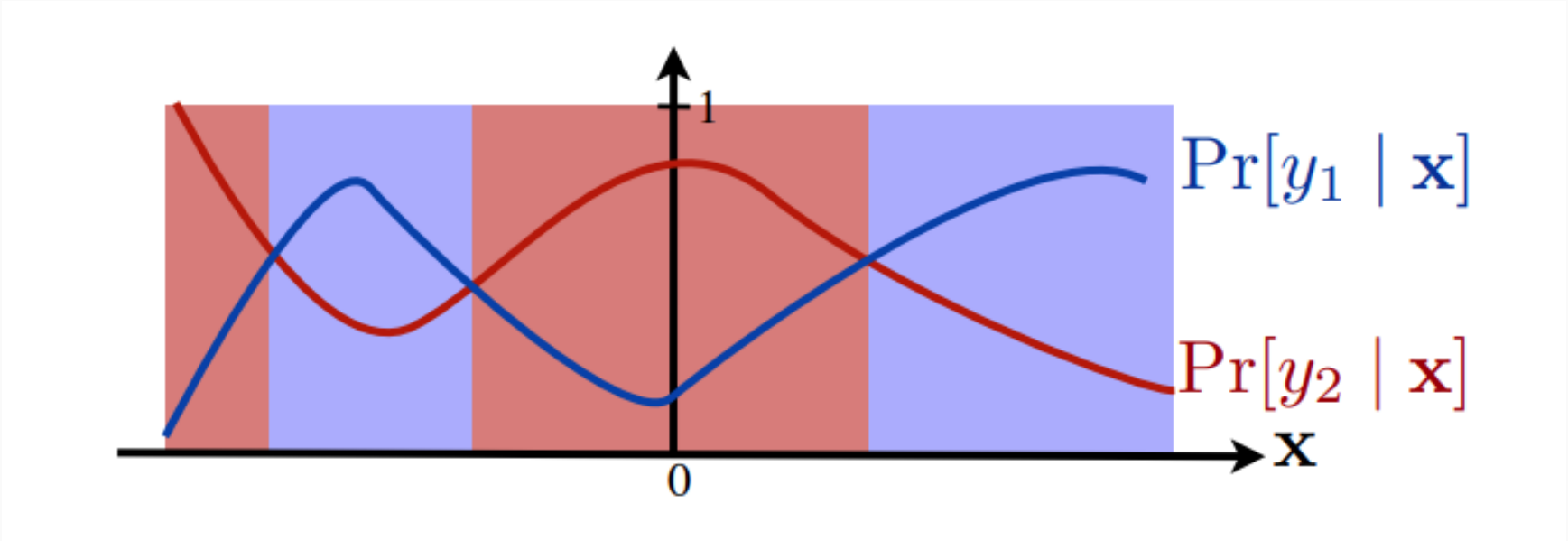
Principiul lamei lui Occam

- preferă explicațiile simple în detrimentul celor complexe;
- selectează ipoteza care implică mulțimea minimală de presupuneri, între ipoteze similare/egal probabile;
- Wiliam of Occam, 1285 – 1347 (filosof englez)



Exemplu de clasificare

- Dorim să clasificăm personajele de film în funcție de modul în care arată. Vom defini o problemă de clasificare binară pentru care cele două clase sunt **Bun** și **Rău**.



- Pentru fiecare exemplu, vom avea la dispoziție următoarele atribute: Sex, Mască, Pelerină, Cravată, Urechi, Luptă, Clasă (eticheta).

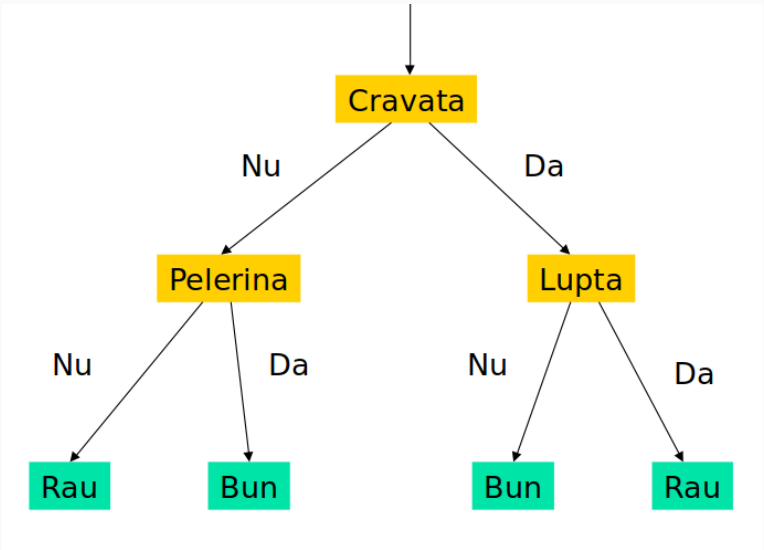
Exemplu de clasificare

Setul de date

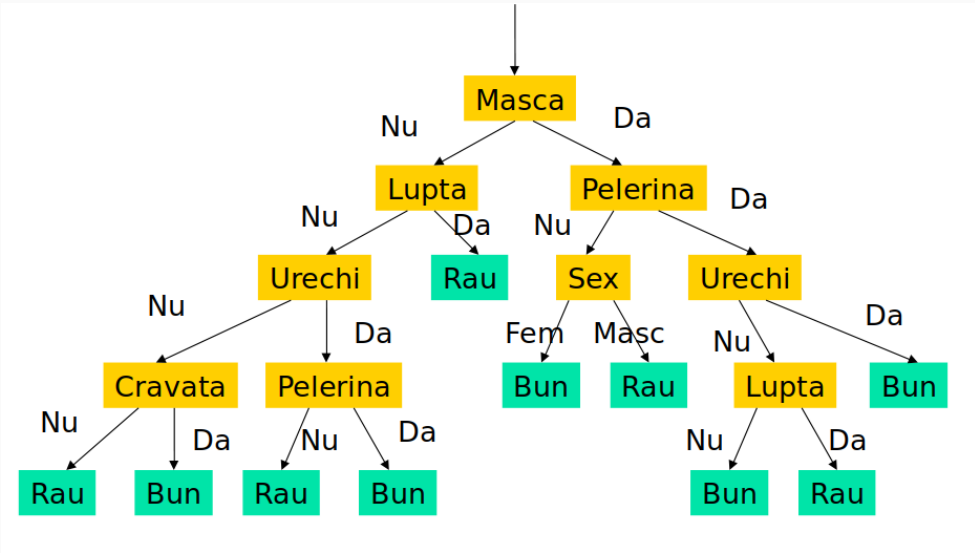
Atribute / Instanțe	Sex	Masca	Pelerina	Cravata	Urechi	Lupta	**Clasa**
	Set de invatare						
Batman	Masc	Da	Da	Nu	Da	Nu	**Bun**
Robin	Masc	Da	Da	Nu	Nu	Nu	**Bun**
Alfred	Masc	Nu	Nu	Da	Nu	Nu	**Bun**
Pinguin	Masc	Nu	Nu	Da	Nu	Da	**Rau**
Catwoman	Fem	Da	Nu	Nu	Da	Nu	**Rau**
Joker	Masc	Nu	Nu	Nu	Nu	Nu	**Rau**
	Date de test						
Batgirl	Fem	Da	Da	Nu	Da	Nu	**??**
Fred	Masc	Da	Nu	Nu	Nu	Nu	**??**

Exemplu de clasificare

Arbore de decizie – Varianta 1



Arbore de decizie – Varianta 2

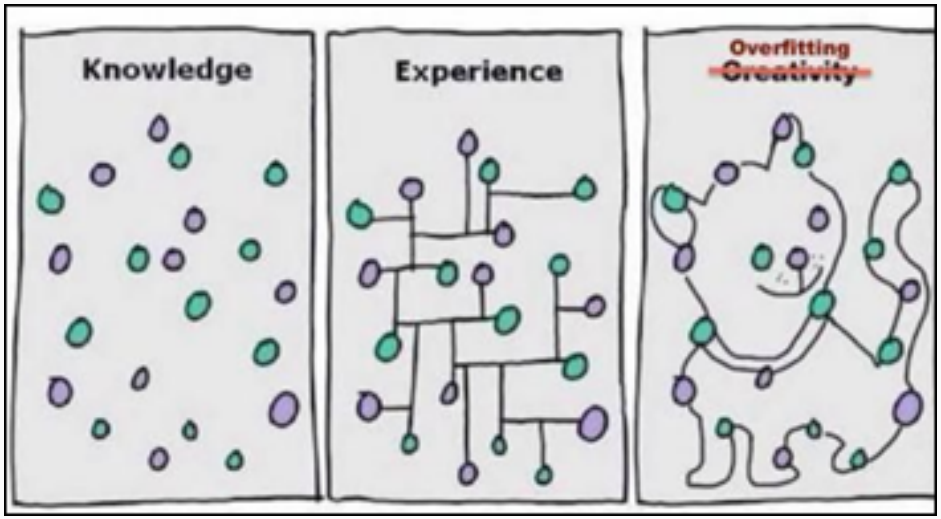


	Date de test						
Batgirl	Fem	Da	Da	Nu	Da	Nu	??
Fred	Masc	Da	Nu	Nu	Nu	Nu	??

Exemplu de clasificare

Observație

Clasificatoarele trebuie să fie suficient de "expresive" pentru a fi în concordanță cu setul de învățare.
Dar clasificatoarele care au o complexitate prea mare pot duce la fenomenul de "overfit" (overfitting)



Overfitting vs Underfitting

Overfiting

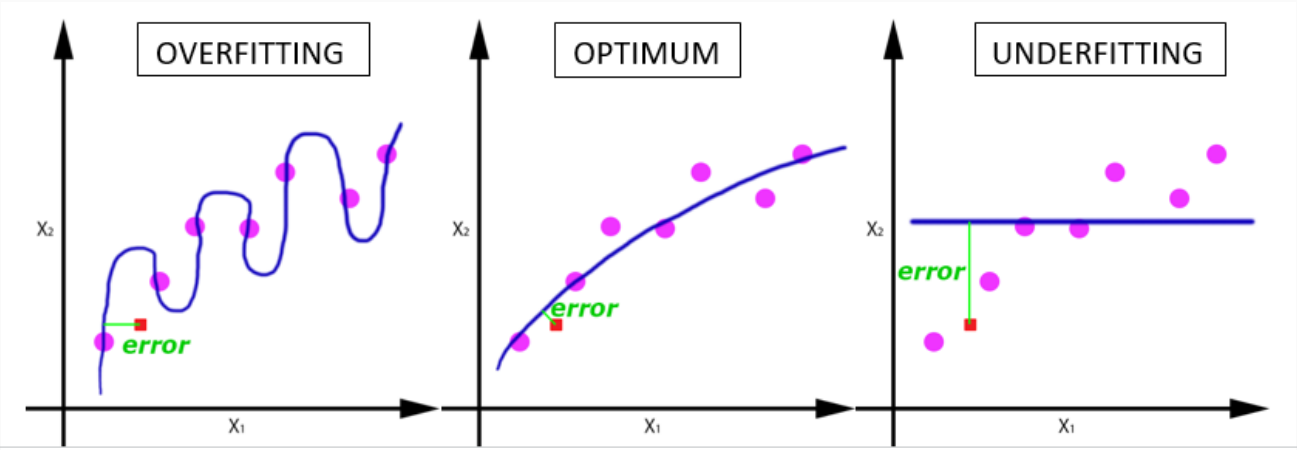
Modelul include zgomot sau șabloane de date nerelevante, devenind mult prea specific. Astfel, NU mai poate generaliza!

Soluții

1. Validare încrucișată
2. Regularizare

Underfitting

Modelul nu se potrivește nici datelor de învățare și nici nu poate generaliza la date noi.



Evitarea overfitting-ului

Regularizare

- Evită overfitting
- Se poate aplica aproape oricărui model bazat pe ML
- Simplifică modelele cu complexitate prea mare prin adăugarea de termeni de penalizare în funcția obiectiv

Cross-validation

- Antrenarea se face pe subseturi de date de învățare și evaluarea pe restul

Metrici de performanță

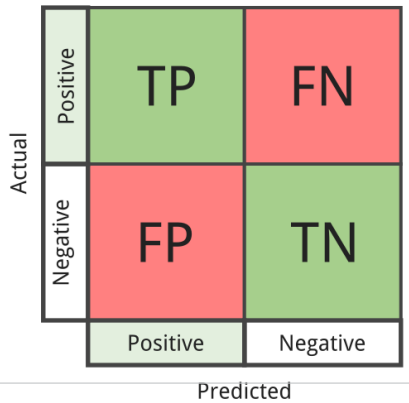
1. **Regresie**
   - **Mean Absolute Error** (MAE) – media diferenței absolute între valori corecte și valori prezise

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

   - **Root Mean Square Error** (RMSE) – rădăcina pătrată a mediei diferențelor pătratelor între valori corecte și valori prezise

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}$$

2. **Clasificare**
   - Confusion matrix – corectitudinea și acuratețea modelului



2. Clasificare

- Accuracy – utilă dacă sunt distribuite uniform clasele;
- Precision
- Recall sau Sensitivity
- Specificity

		Predicted Class: Positive	Predicted Class: Negative	
Actual Class	Positive	True Positive (TP)	False Negative (FN) **Type II Error**	**Sensitivity** $\frac{TP}{(TP+FN)}$
	Negative	False Positive (FP) **Type I Error**	True Negative (TN)	**Specificity** $\frac{TN}{(TN+FP)}$
		Precision $\frac{TP}{(TP+FP)}$	**Negative Predictive Value** $\frac{TN}{(TN+FN)}$	**Accuracy** $\frac{TP+TN}{(TP+TN+FP+FN)}$

Clasificator binar liniar

- Vom avea 2 clase, iar eticheta t a unui exemplu va lua o valoare din mulțimea $\{0, 1\}$.
- Fie $\mathbf{X}$ un exemplu din setul de date pentru care dorim să facem o predicție asupra clasei folosind un clasificator binar liniar:

  $z = \mathbf{W}^T\mathbf{X} + b$, unde $\mathbf{W}$ reprezintă o matrice de ponderi, iar b este un bias

- Pornind de la valoarea lui z, putem determina clasa corespunzătoare lui $\mathbf{X}$ folosind:

$$y = \begin{cases} 1 & \text{dacă } z \geq 0 \\ 0 & \text{dacă } z < 0 \end{cases}$$

- Avem nevoie de o funcție de eroare care să testeze pentru un sample $(\mathbf{X}, t)$ corectitudinea predicției:

$$\mathcal{L}_{0/1}(y, t) = \begin{cases} 0 & \text{dacă } y = t \\ 1 & \text{dacă } y \neq t \end{cases}$$

Scopul antrenării

- Dorim să obținem un clasificator pentru care eroarea totală pentru setul de antrenare să fie minimă.
- Eroarea totală

$$\mathcal{E} = \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(y^i, t^i)$$

- Trebuie descoperit cum actualizăm ponderile clasificatorului (valorile din matricea $\mathbf{W}$) astfel încât să minimizâm eroarea totală.

Soluția: Calculăm gradientul

$$\frac{\partial \mathcal{L}_{0/1}}{\partial w_j} = \frac{\partial \mathcal{L}_{0/1}}{\partial z} \cdot \frac{\partial z}{\partial w_j}$$

Important

$\frac{\partial \mathcal{L}_{0/1}}{\partial z}$ este 0. Acest lucru înseamnă că dacă realizăm o modificare minoră asupra unei ponderi atunci acest lucru nu va afecta eroarea.

Problemă: Funcția de eroare aleasă nu este potrivită. O vom înlocui cu:

Cross Entropy Loss

$$\mathcal{L}_{CE}(y,t) = \begin{cases} -\log y & \text{dacă } t = 1 \\ -\log(1-y) & \text{dacă } t = 0 \end{cases} = -t\log y - (1-t)\log(1-y)$$

- De această dată, avem următoarele formule:
  $z = \mathbf{W}^T\mathbf{X} + b, y = \sigma(z) = \frac{1}{1+e^{-z}}$
- Vom calcula iar gradientul:

$$\frac{\partial \mathcal{L}_{CE}}{\partial w_j} = \frac{\partial \mathcal{L}_{CE}}{\partial z} \cdot \frac{\partial z}{\partial w_j}$$

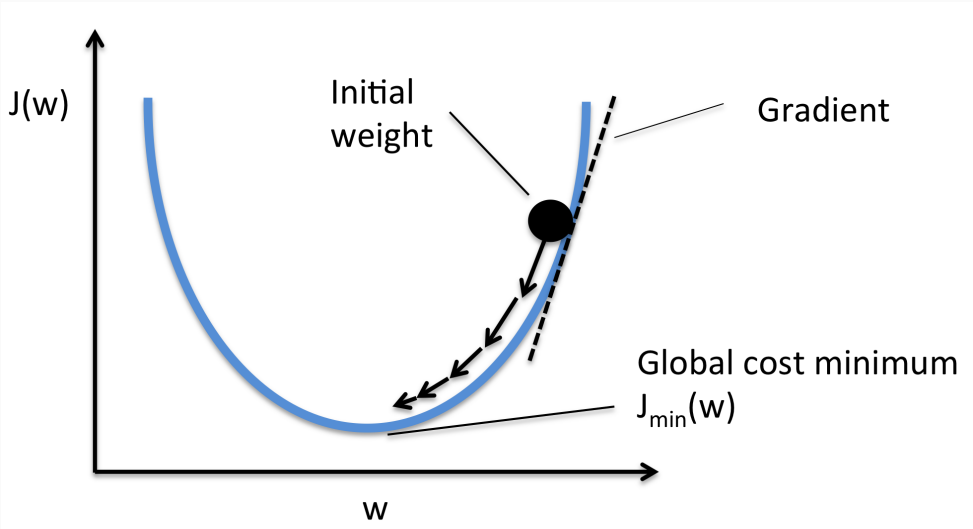
$$\frac{\delta \mathcal{L}_{CE}}{\delta z} = \frac{\delta \mathcal{L}_{CE}}{\delta y} \cdot \frac{\delta y}{\delta z} = \left(-\frac{t}{y} + \frac{1-t}{1-y}\right) \cdot y(1-y) = -(1-y)t + y(1-t) = y - t$$

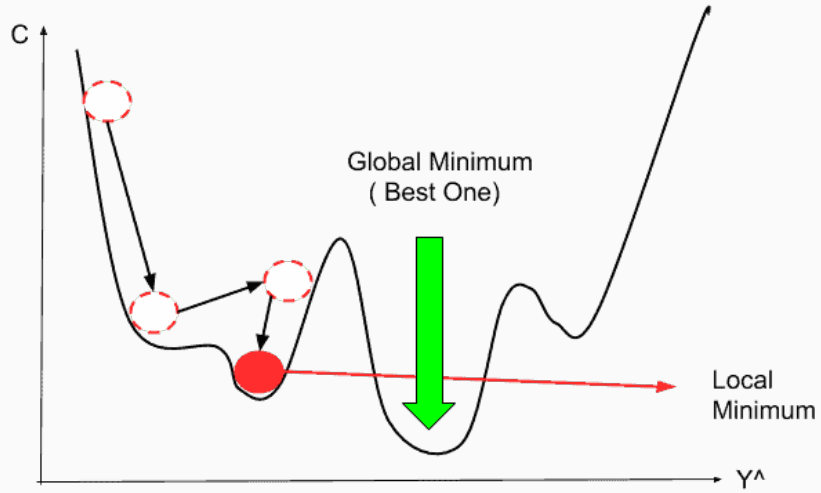
Scăderea gradientului

- Metodă iterativă pentru găsirea minimului unei funcții.
- Pentru a afla minimul unei funcții, se fac pași proporționali cu negativul derivatei/gradientului funcției în punctul curent.
- Sau proporționali cu pozitivul derivatei/gradientului pentru maxim

Scopul: Minimizăm o funcție de cost $J(\theta)$, unde θ sunt ponderile modelului.

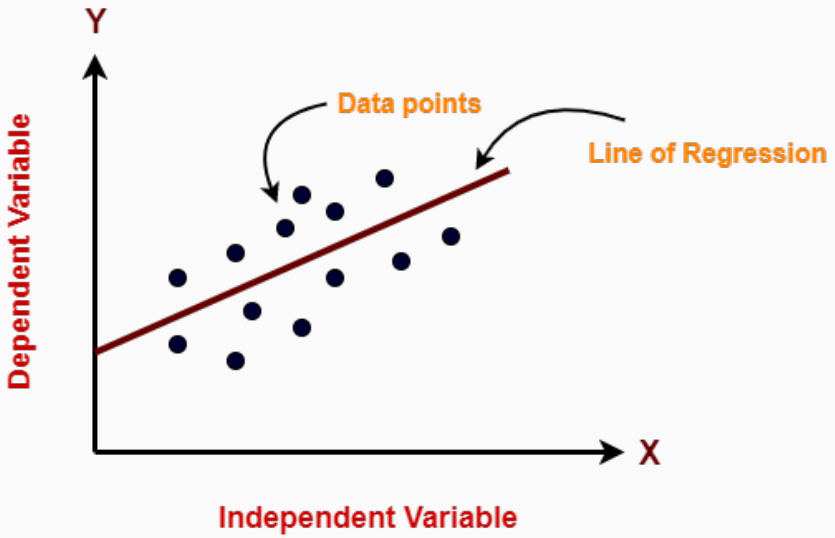
Problemă: Nu garantează găsirea valorii optime globale.





Proprietăți

- Algoritm de învățare supervizată;
- Învață valori continue pe baza exemplelor de învățare



Formularea problemei

- Dorim să prezicem un scalar t pornind de la un scalar x.
- Avem la dispoziție un set de puncte $\{(x^{(i)}, t^{(i)})\}_{i=1}^{N}$

Modelul

- y este o funcție liniară de x: $y = wx + b$
- y – predicția modelului ; w – pondere ; b – bias ; w și b – reprezintă parametri modelului

Funcția de eroare (Loss function)

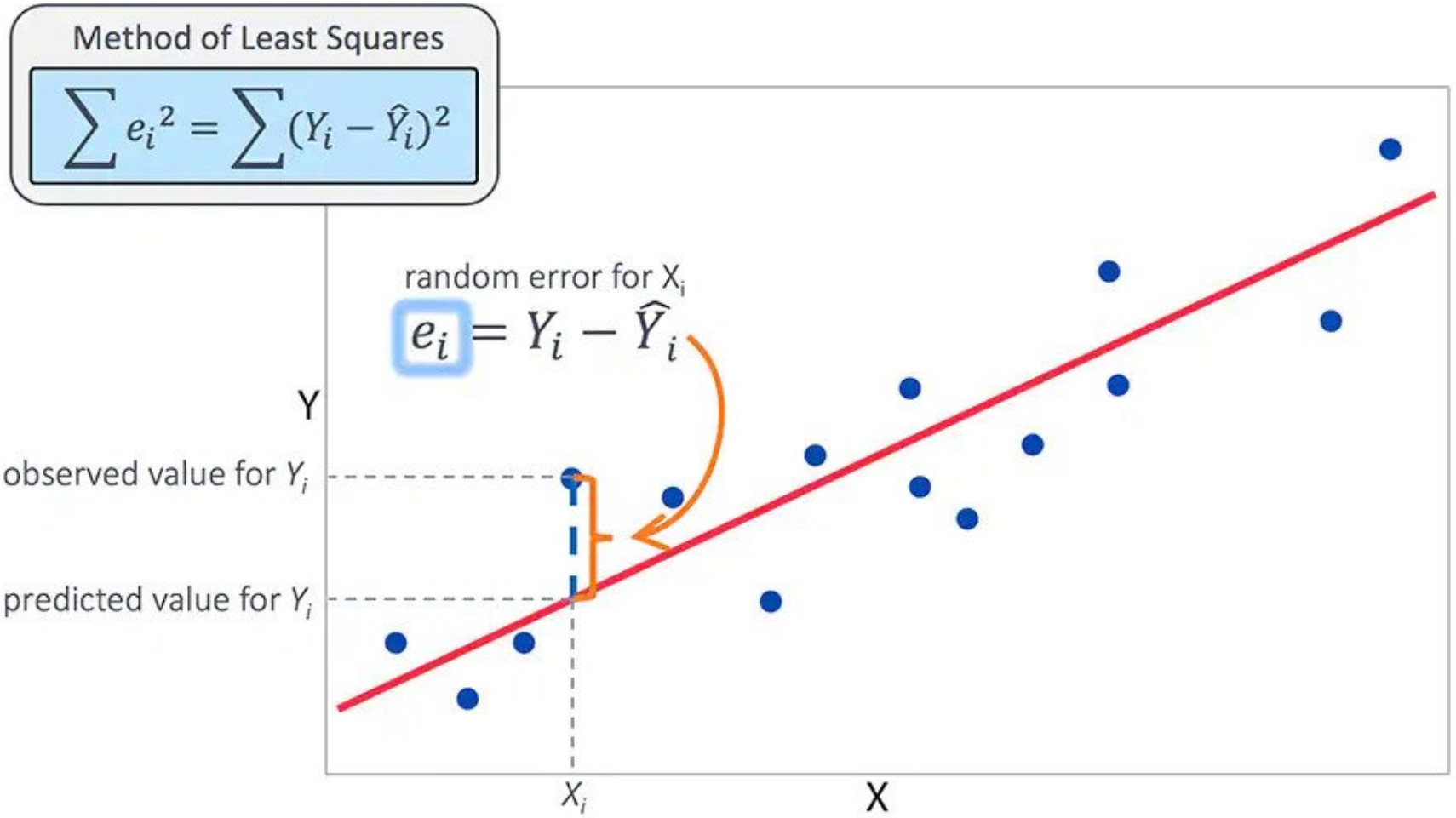
$$\mathcal{L}(y, t) = \frac{1}{2}(y - t)^2$$

Eroarea totală / Funcția de cost (Cost function)

- funcția de eroare *mediată* peste toate exemplele de antrenare

$$\mathcal{E}(w, b) = \frac{1}{2N}\sum_{i=1}^{N}(y^{(i)} - t^{(i)})^2 = \frac{1}{2N}\sum_{i=1}^{N}(wx^{(i)} + b - t^{(i)})^2$$

Regresie liniară



Regresie liniară – Gradient descent

Scopul: Dorim să minimizăm funcția de cost $\mathcal{E}(w, b)$

$$\mathcal{E}(w, b) = \frac{1}{2N}\sum_{i=1}^{N}(y^{(i)} - t^{(i)})^2 = \frac{1}{2N}\sum_{i=1}^{N}(wx^{(i)} + b - t^{(i)})^2$$

1. Inițializăm ponderile w și b
2. Repetă până la convergență

$$w \leftarrow w - \alpha\frac{\partial}{\partial w}\mathcal{E}(w, b)$$

$$b \leftarrow b - \alpha\frac{\partial}{\partial b}\mathcal{E}(w, b)$$

α – viteza de învățare

Regresie liniară – Gradient descent

Scopul: Dorim să minimizăm funcția de cost $\mathcal{E}(w, b)$

$$\mathcal{E}(w, b) = \frac{1}{2N} \sum_{i=1}^{N} (y^{(i)} - t^{(i)})^2 = \frac{1}{2N} \sum_{i=1}^{N} (wx^{(i)} + b - t^{(i)})^2$$

1. Inițializăm ponderile w și b
2. Repetă până la convergență

$$w \leftarrow w -$$

Regresie liniară – Gradient descent

Scopul: Dorim să minimizăm funcția de cost $\mathcal{E}(w, b)$

$$\mathcal{E}(w, b) = \frac{1}{2N} \sum_{i=1}^{N} (y^{(i)} - t^{(i)})^2 = \frac{1}{2N} \sum_{i=1}^{N} (wx^{(i)} + b - t^{(i)})^2$$

1. Inițializăm ponderile w și b
2. Repetă până la convergență

$$w \leftarrow w - \alpha \frac{1}{N} \sum_{i=1}^{N} (wx^{(i)} + b - t^{(i)}) \cdot x^{(i)}$$

$$b \leftarrow b -$$

Regresie liniară – Gradient descent

Scopul: Dorim să minimizăm funcția de cost $\mathcal{E}(w, b)$

$$\mathcal{E}(w, b) = \frac{1}{2N}\sum_{i=1}^{N}(y^{(i)} - t^{(i)})^2 = \frac{1}{2N}\sum_{i=1}^{N}(wx^{(i)} + b - t^{(i)})^2$$

1. Inițializăm ponderile w și b
2. Repetă până la convergență

$$w \leftarrow w - \alpha\frac{1}{N}\sum_{i=1}^{N}(wx^{(i)} + b - t^{(i)}) \cdot x^{(i)}$$

$$b \leftarrow b - \alpha\frac{1}{N}\sum_{i=1}^{N}(wx^{(i)} + b - t^{(i)})$$

α – viteza de învățare

Observații

- Pe măsură ce ne apropiem de minimul local, se vor face pași din ce în ce mai mici.
- Regim de învățare de tip batch — utilizează toate exemplele de învățare.
- α – viteza de învățare
- Daca α prea mică $\Rightarrow$ converge încet algoritmul de învățare.
- Daca α prea mare poate să nu găsească minimul.
- Dacă inițializăm w și b chiar cu valorile minimului $\Rightarrow \mathcal{E}$ nu se mai modifică (derivata parțială este 0)

Regresie liniară multi-variabile

Modificare: Vom considera o mulțime de atribute $x_1, x_2, \ldots, x_m$.

- Este întâlnită cu numele de **multi-variable regression**.
- Nu există diferențe între varianta anterioară și această (este doar mai complicat de realizat vizualizarea).

Modelul liniar

$$y = \sum_{j=1}^{j=m} w_j x_j + b$$

Implementare

```
y = b
for j in range(M):
  y += w[j] * x[j]
```

Modificare: Vom considera o mulțime de atribute $x_1, x_2, \ldots, x_m$.

- Este întâlnită cu numele de **multi-variable regression**.
- Nu există diferențe între varianta anterioară și această (este doar mai complicat de realizat vizualizarea).

Modelul liniar

$$y = \sum_{j=1}^{j=m} w_j x_j + b$$

Implementare

```
y = b
for j in range(M):
  y += w[j] * x[j]
```

Important

Variantă ineficientă, deoarece sunt încete loop-urile în Python.
Pentru simplitate și eficiență, vom aplica operația de vectorizare.

Vectorizare

$$\mathbf{w} = (w_1, w_2, \ldots, w_m)^T$$

$$\mathbf{x} = (x_1, x_2, \ldots, x_m)$$

$$y = \mathbf{w}^T\mathbf{x} + b$$

Implementare

```
import numpy as np
y = np.dot(w, x) + b
```

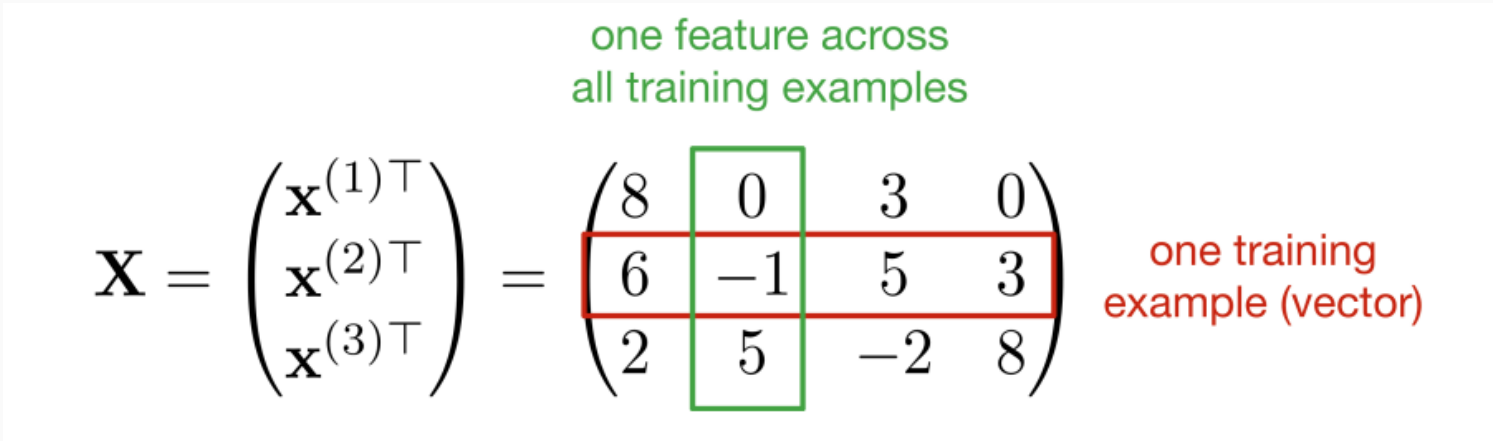
Important

Putem să utilizăm biblioteci eficiente care efectuează calcule algebrice în forma matriceală (ex. `numpy`).
Înmulțirea matricelor se realizează foarte eficient pe GPU.

Vectorizare

Important

Putem optimiza și mai mult. Organizăm toate exemplele într-o matrice $\mathbf{X}$ (câte un rând pentru fiecare exemplu de antrenare).



$$\mathbf{X} = \begin{pmatrix} \mathbf{x}^{(1)\top} \\ \mathbf{x}^{(2)\top} \\ \mathbf{x}^{(3)\top} \end{pmatrix} = \begin{pmatrix} 8 & 0 & 3 & 0 \\ 6 & -1 & 5 & 3 \\ 2 & 5 & -2 & 8 \end{pmatrix}$$

Modelul liniar

$$\mathbf{X}\mathbf{w} + b\mathbf{1} = \begin{bmatrix} \mathbf{w}^T\mathbf{x}^{(1)} + b \\ \mathbf{w}^T\mathbf{x}^{(2)} + b \\ \vdots \\ \mathbf{w}^T\mathbf{x}^{(N)} + b \end{bmatrix} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(N)} \end{bmatrix}$$

Modelul liniar

$$\mathbf{X}\mathbf{w} + b\mathbf{1} = \begin{bmatrix} \mathbf{w}^T\mathbf{x}^{(1)} + b \\ \mathbf{w}^T\mathbf{x}^{(2)} + b \\ \vdots \\ \mathbf{w}^T\mathbf{x}^{(N)} + b \end{bmatrix} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(N)} \end{bmatrix}$$

Gradient descent

- Dacă $\frac{\partial \mathcal{E}}{\partial w_j} > 0 \Rightarrow$ crescând valoarea lui w_j o să crească și eroarea.
- Dacă $\frac{\partial \mathcal{E}}{\partial w_j} < 0 \Rightarrow$ crescând valoarea lui w_j o să scadă eroarea.

$$w_j \leftarrow w_j - \alpha \frac{\partial \mathcal{E}}{\partial w_j} = w_j - \frac{\alpha}{N} \sum_{i=1}^{N} (y^{(i)} - t^{(i)}) x_j^{(i)}$$

Important

Funcția de cost pentru regresie liniară este o funcție convexă.

Gradient descent

- Dacă funcția de cost este strict convexă (dacă domeniul parametrilor este strict convex, i.e., nu există restricții asupra coeficienților, nu există regularizare, etc.) atunci scăderea gradientului va avea o soluție unică și aceasta va îndeplini criteriul de optim global.
- În caz contrar, pot exista mai multe soluții.
- Pentru ca GD să conveargă repede, este bine ca valorile de atribute să fie cam în același interval. Putem realiza acest lucru folosind normalizare.

Regresie logistică

Proprietăți

- Algoritm de învățare supervizată
- Utilizată în clasificare

Tipuri de regresie logistică

1. Regresie logistică binomială (binară)
2. Regresie logistică multinomială (multi-clasă)

Greșit

- Am văzut anterior că nu este o idee bună să folosim o variantă modificată de regresie liniară pentru a face clasificare.

Ipoteza de învățare

$$0 \leq h_\theta(\mathbf{x}) \leq 1, h(\mathbf{x}) = g(\theta^T \mathbf{x})$$

$h_\theta(\mathbf{x})$ estimează probabilitatea ca $y = 1$ pentru o intrare $\mathbf{x}$

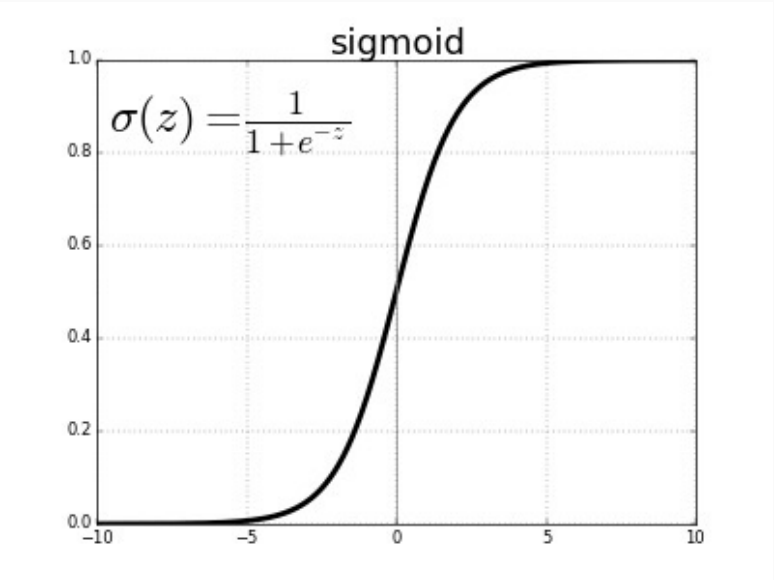
$$h_\theta(\mathbf{x}) = P(y = 1 | \mathbf{x}; \theta)$$

Regresie logistică

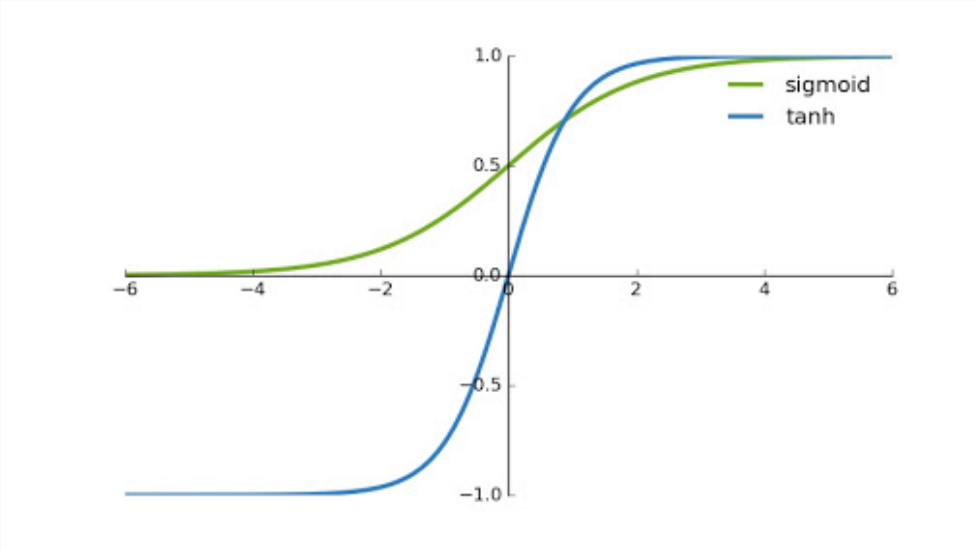
$$g(z) = \frac{1}{1 + e^{-z}}$$

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

Sigmoid



Tangenta hiperbolică



Ipoteza de învățare

$0 \leq h_\theta(x) \leq 1$ – presupunem o relație liniară între 2 atribute x_1 și x_2 și "log odds" (l) al clasei y

$$l = \ln \frac{p}{1-p} = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

$$\frac{p}{1-p} = e^{\theta_0 + \theta_1 x_1 + \theta_2 x_2}$$

$$p = \frac{1}{1 + e^{-(\theta_0 + \theta_1 x_1 + \theta_2 x_2)}}$$

Funcția de cost

Dacă am utiliza aceeași funcție de cost ca în cazul regresiei liniare, atunci funcția nu ar mai fi convexă (deci nu avem un unic minim global). De aceea, avem nevoie să definim o altă funcție de cost.

Funcția de cost

$$Cost(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{dacă } y = 1 \\ -\log(1 - h_\theta(x)) & \text{dacă } y = 0 \end{cases}$$

Înseamnă că dacă $h_\theta(x) = 0$ costul tinde la infinit deci predicția este $P(y = 1|x; \theta) = 0$

Scriem compact:

$$Cost(h_\theta(x), y) = -y \log(h_\theta(x)) - (1 - y) \log(1 - h_\theta(x))$$

Funcția de cost:

$$\mathcal{E}(\theta) = -\frac{1}{2N} \sum_{i=1}^{N} y^{(i)} \log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))$$

Clasificare “**one-versus-all**”, K clase

Clasa **1** $h^1_\theta(x) = P(Y=1 \mid X; \theta_1)$ – determina setul de parametrii θ_1

Clasa **2** $h^2_\theta(x) = P(Y=2 \mid X; \theta_2)$ – determina setul de parametrii θ_2

etc.

Pentru o instanta necunoscuta x, determina $h^i_\theta(x)$ pt fiecare clasa si alege clasa pentru care $h^i_\theta(x)$ are valoarea maxima

Pentru K clase, se executa regresie logistica pentru K-1 modele de regresie logistica binara, din care o clasa este aleasa ca pivot si celelalte K-1 sunt clasificate in functie de pivot. Daca clasa K este aleasa ca pivot atunci:

$$\ln \frac{\Pr(Y_i = 1)}{\Pr(Y_i = K)} = \theta_1 \cdot \mathbf{X}_i$$

$$\ln \frac{\Pr(Y_i = 2)}{\Pr(Y_i = K)} = \theta_2 \ \mathbf{X}_i$$

$$\cdots\cdots$$

$$\ln \frac{\Pr(Y_i = K-1)}{\Pr(Y_i = K)} = \theta_{K-1} \cdot \mathbf{X}_i$$

Soluții pentru minimizare

1. Reducerea numarului de atribute
   - Manual
   - Algoritmi de selecție a modelului

Observație

Poate reduce overfitting, dar elimină informație.

2. Regularizare
   - Păstrez toate atributele, dar reduc dimensiunea parametrilor $\theta_1, \theta_2, \ldots, \theta_m$.

Observație

Potrivită dacă avem multe atribute, fiecare contribuind cu puțin/la fel la predicție.