



# Sisteme de Vedere Artificială

## Recunoaşterea Formelor

Sorin M. Grigorescu

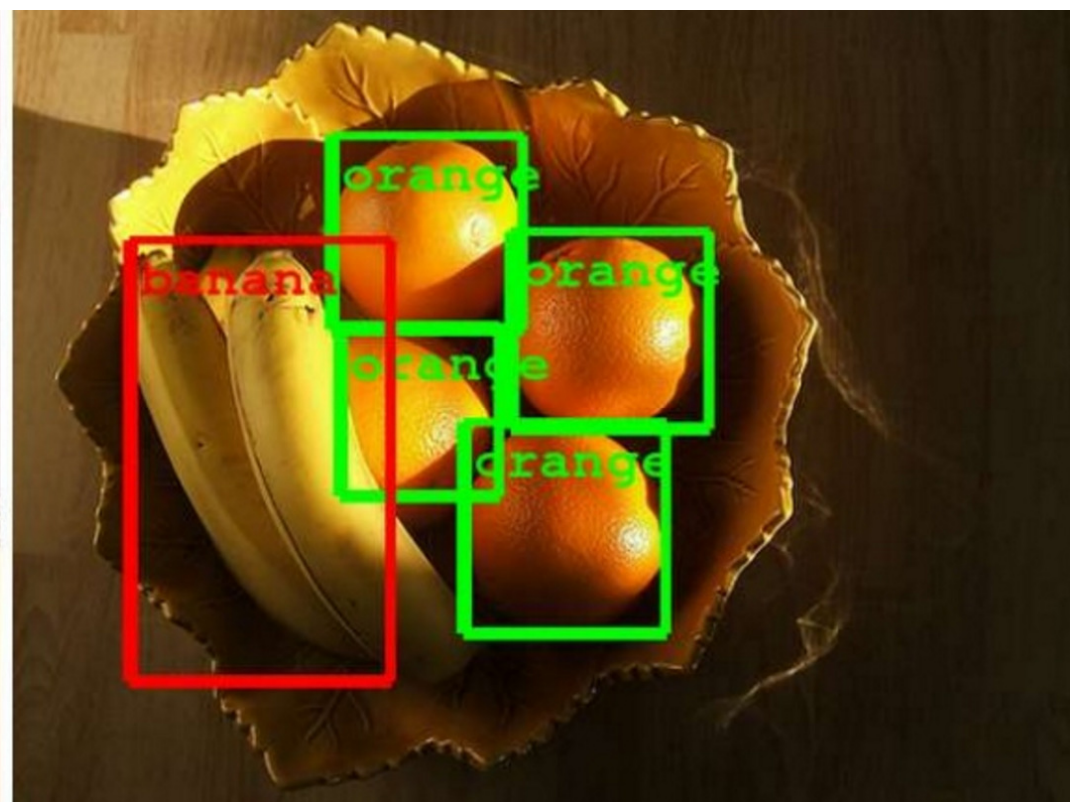
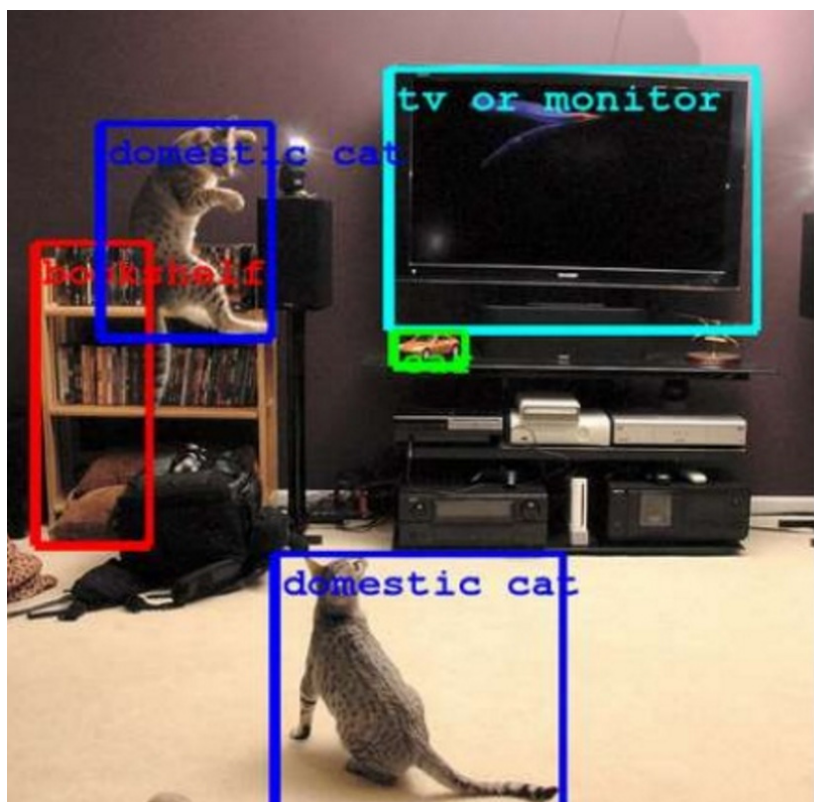


# Cuprins

- Regresia liniară
- Regresia logistică (clasificarea binară)
- Rețele neurale

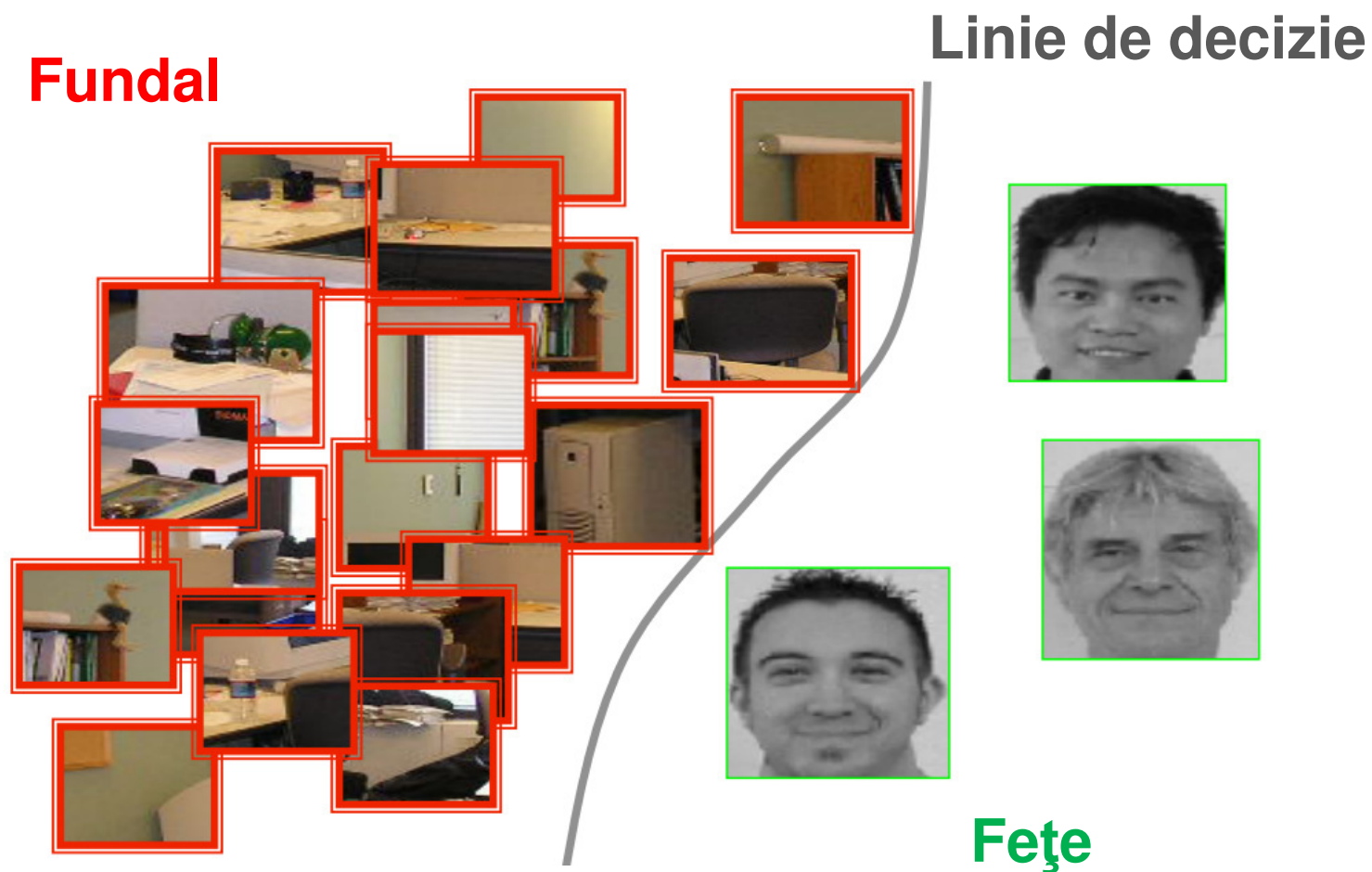


# Recunoașterea formelor în vederea artificială



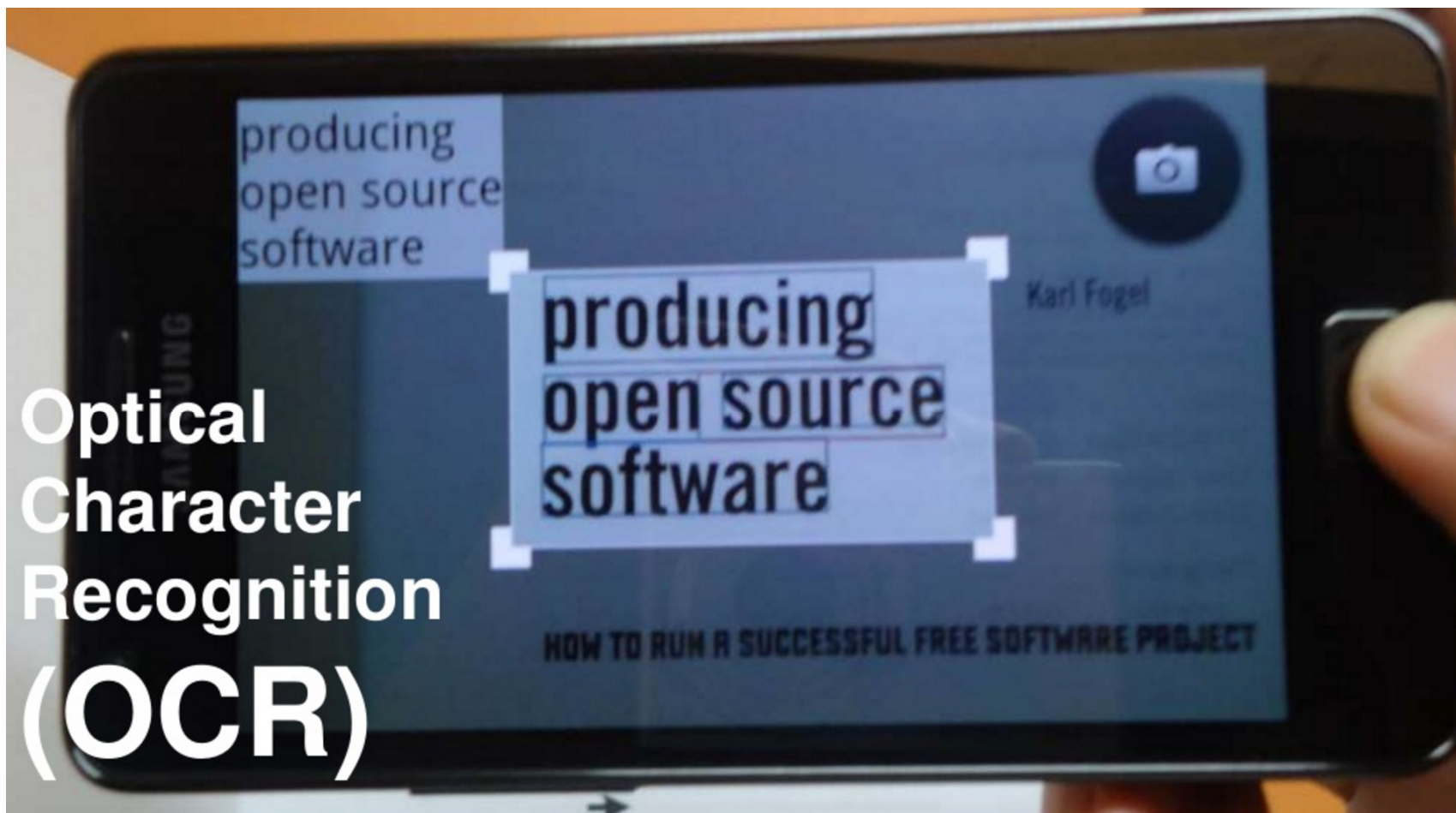


# Recunoașterea formelor în vederea artificială





# Recunoașterea formelor în vederea artificială





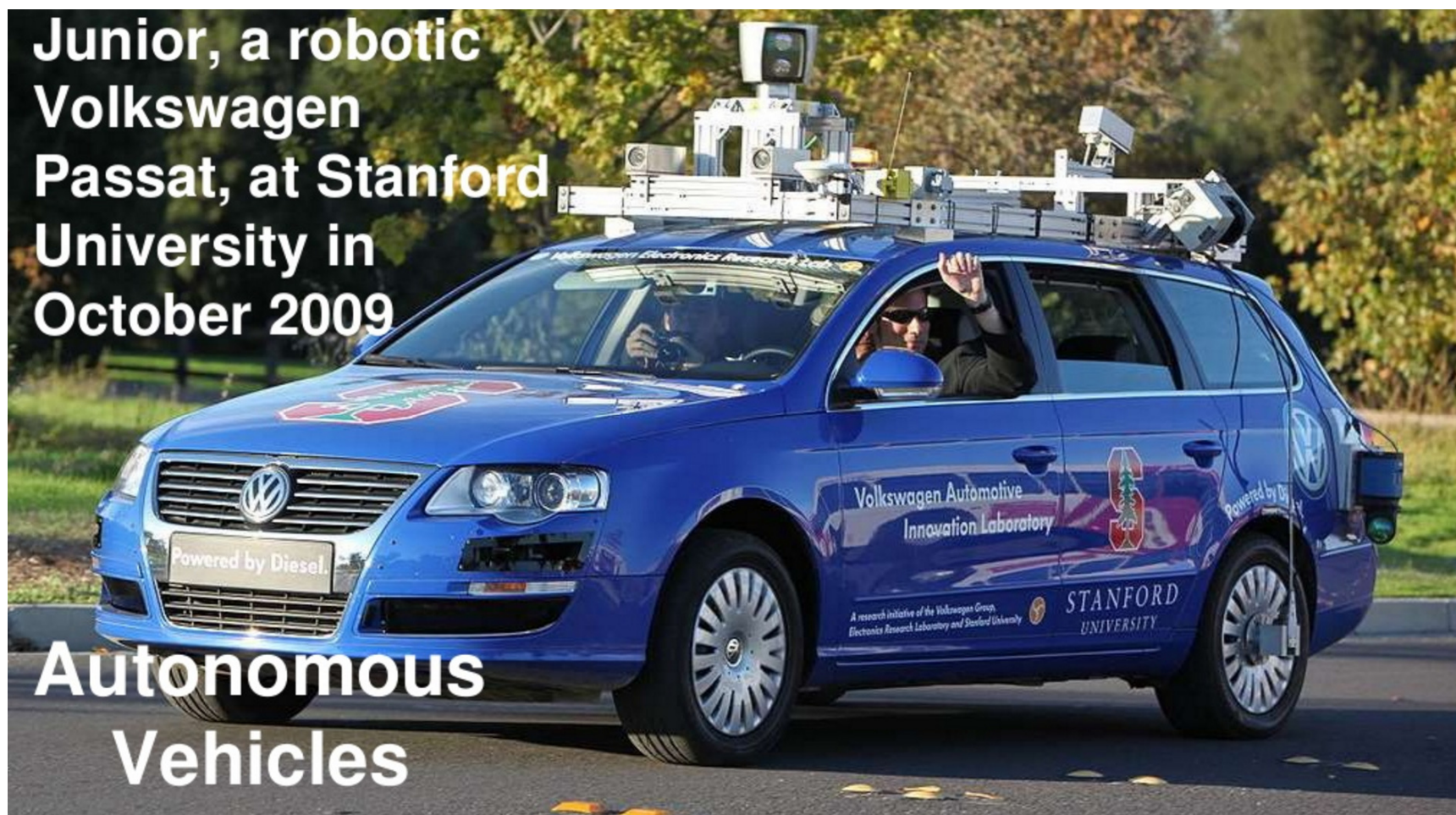
# Recunoașterea formelor în vederea artificială





# Recunoașterea formelor în vederea artificială

Junior, a robotic Volkswagen Passat, at Stanford University in October 2009



Autonomous Vehicles



## Exemplu de învățare supervizată

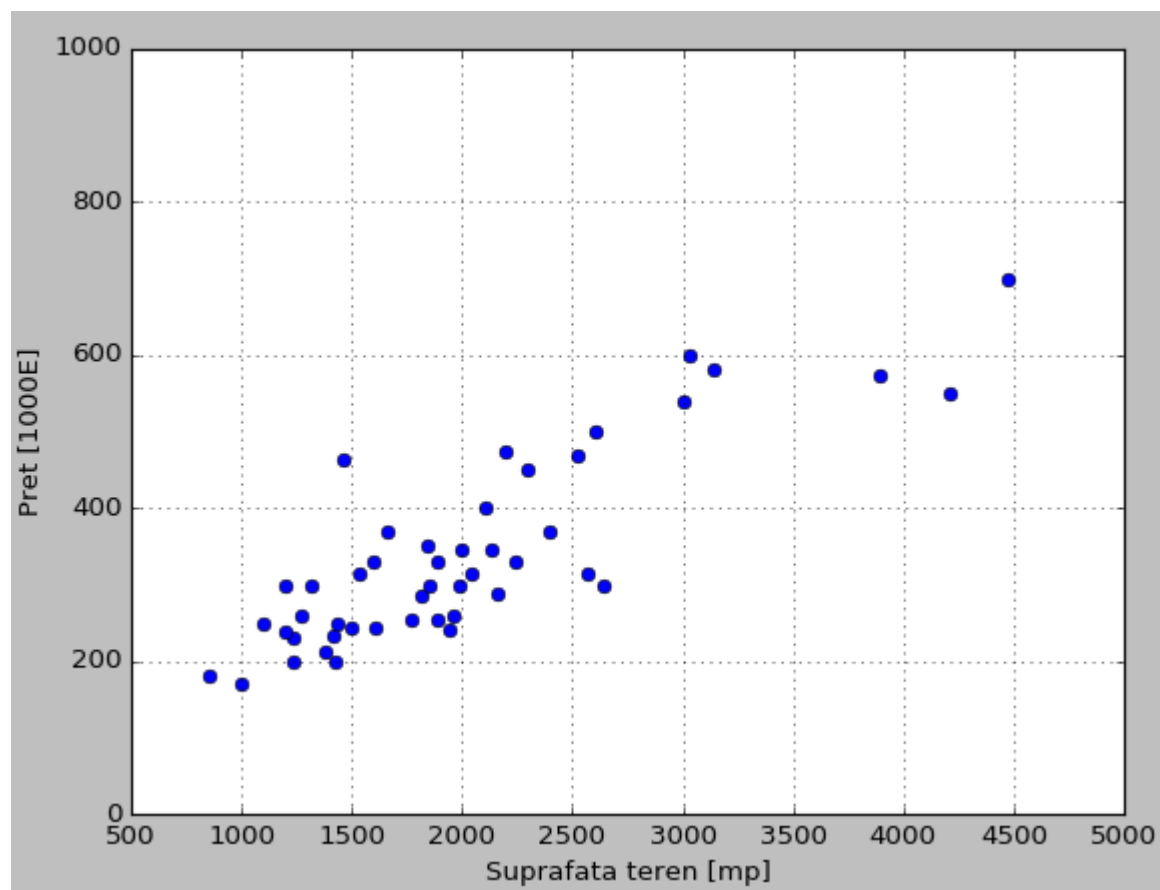
- **Obiectiv: Predicția prețului caselor din zona Braşov, dându-se suprafața terenului**

Suprafață teren [mp]	Preț [1000€]
2104	400
1600	330
2400	369
1416	232
3000	540
⋮	⋮



# Exemplu de învățare supervizată

- **Obiectiv: Predicția prețului caselor din zona Braşov, dându-se suprafața terenului**





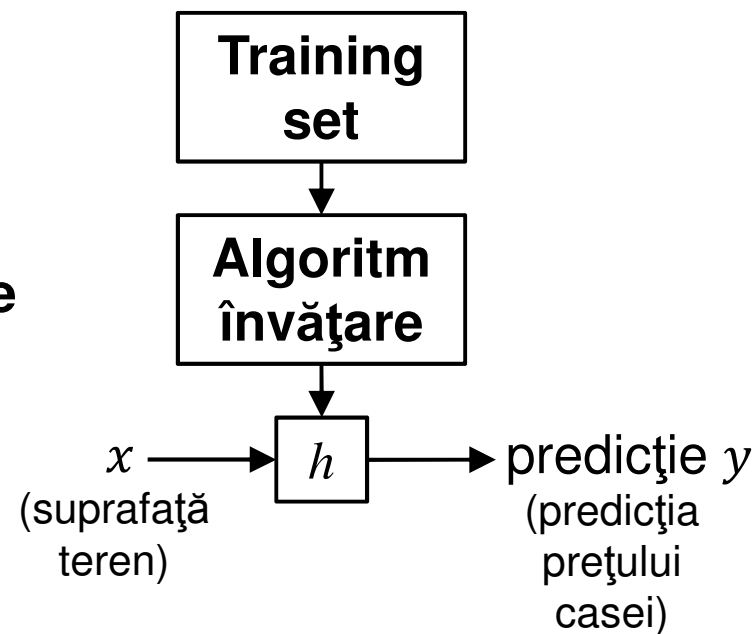
# Notații

- $x^{(i)}$  – vectorul caracteristicilor (features): suprafața terenului
- $y^{(i)}$  – valoarea de ieșire: prețul casei
- $(x^{(i)}, y^{(i)})$  – exemplu de antrenare
- $\{(x^{(i)}, y^{(i)}); i = 1, \dots, m\}$  – baza de date de antrenare (training set), compusă din  $m$  exemple
- $i$  – index către elementele din training set
- $\mathcal{X}$  – spațiul caracteristicilor de intrare
- $\mathcal{Y}$  – spațiul caracteristicilor de ieșire
- În exemplul curent:  $\mathcal{X} = \mathcal{Y} = \mathbb{R}$



# Învățarea supervizată

- Obiectivul algoritmului de învățare este de a învăța funcția  $h_{\theta}: \mathcal{X} \rightarrow \mathcal{Y}$ , în așa fel încât ipoteza  $h_{\theta}(x)$  să reprezinte un predictor "optim" pentru valoarea corespondentă a lui  $y$
- Cazul continuu ( $y \in \mathbb{R}$ ): regresie
- Cazul discret ( $y \in \{0, 1, \dots, k\}$ ): clasificare
- $\theta$  – parametrii modelului

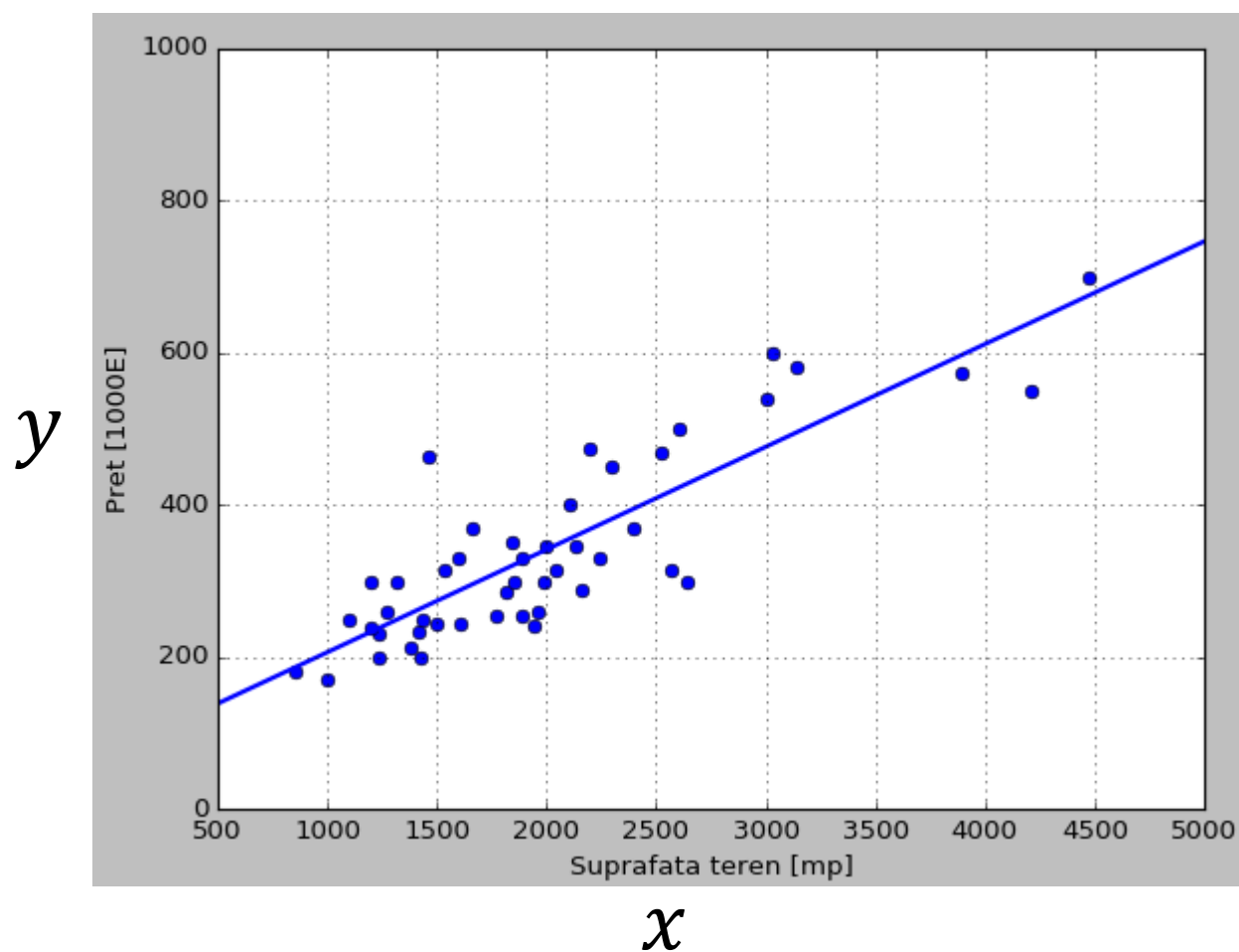




# Regresie liniară

- $h_{\theta}(x)$  – funcție liniară

$$h_{\theta}(x) = \theta_0 + \theta_1 x$$

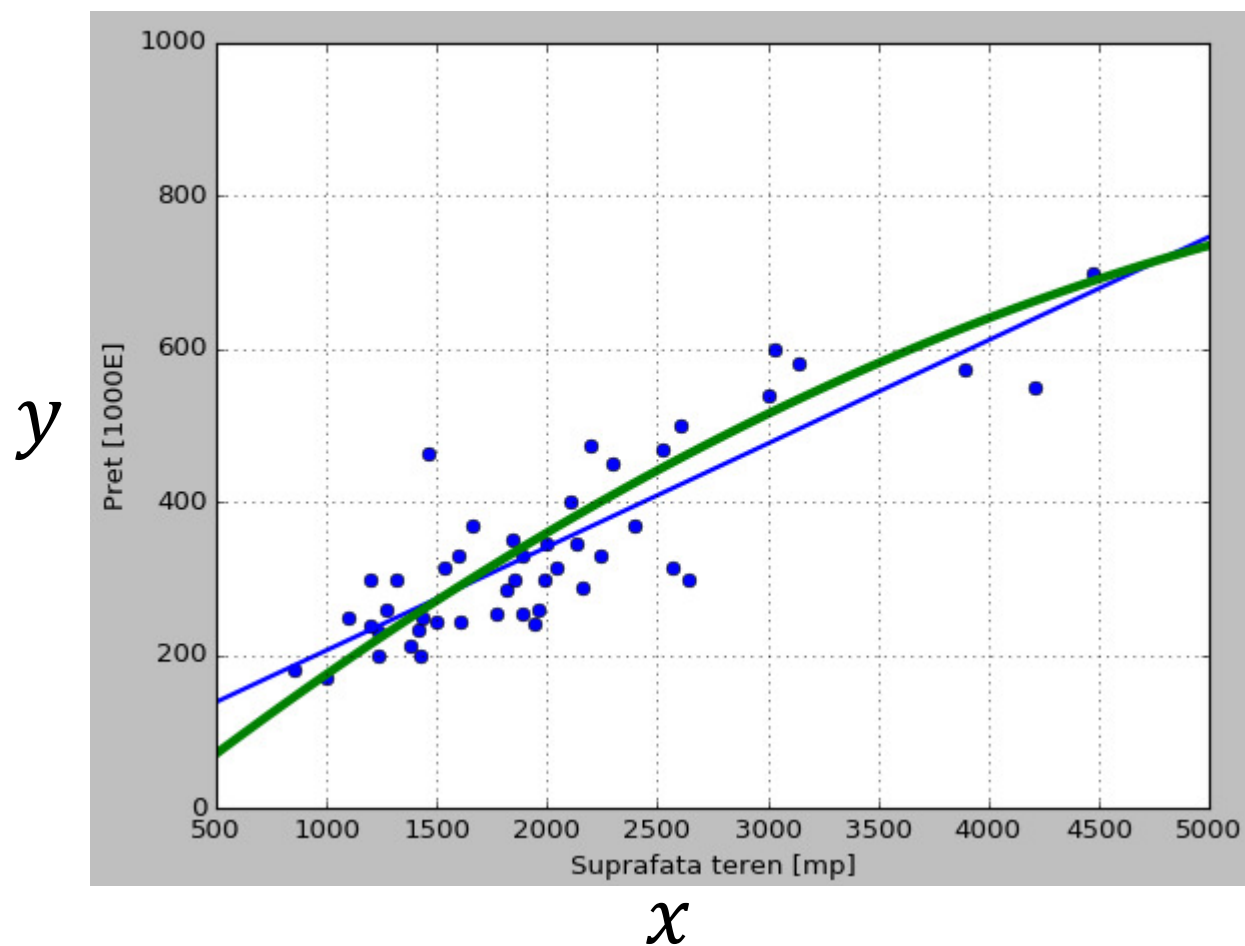




# Regresie liniară

- $h_{\theta}(x)$  – funcție pătratică

$$h_{\theta}(x) = \theta_0 + \theta_1 x + \theta_2 x^2$$





## Cazul $x \in \mathbb{R}^n$

- Cazul numărului de caracteristici multidimensional ( $x \in \mathbb{R}^n$ )
- $n$  – numărul de caracteristici (features)

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = [x_1 \quad x_2]^T$$

$$h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

$x_1$	$x_2$	$y$
Suprafață teren [mp]	Număr dormitoare	Preț [1000€]
2104	3	400
1600	2	330
2400	3	369
1416	2	232
3000	4	540
⋮	⋮	⋮



# Regresie liniară

- Cazul numărului de caracteristici multidimensional ( $x \in \mathbb{R}^n$ )
- $n$  – numărul de caracteristici (features)

$$h_{\theta}(x) = \sum_{i=0}^n \theta_i x_i = \theta^T x$$

$$x_0 = 1$$

- Obiectivul algoritmului de antrenarea: determinarea valorilor parametrilor  $\theta$



## Funcția de cost

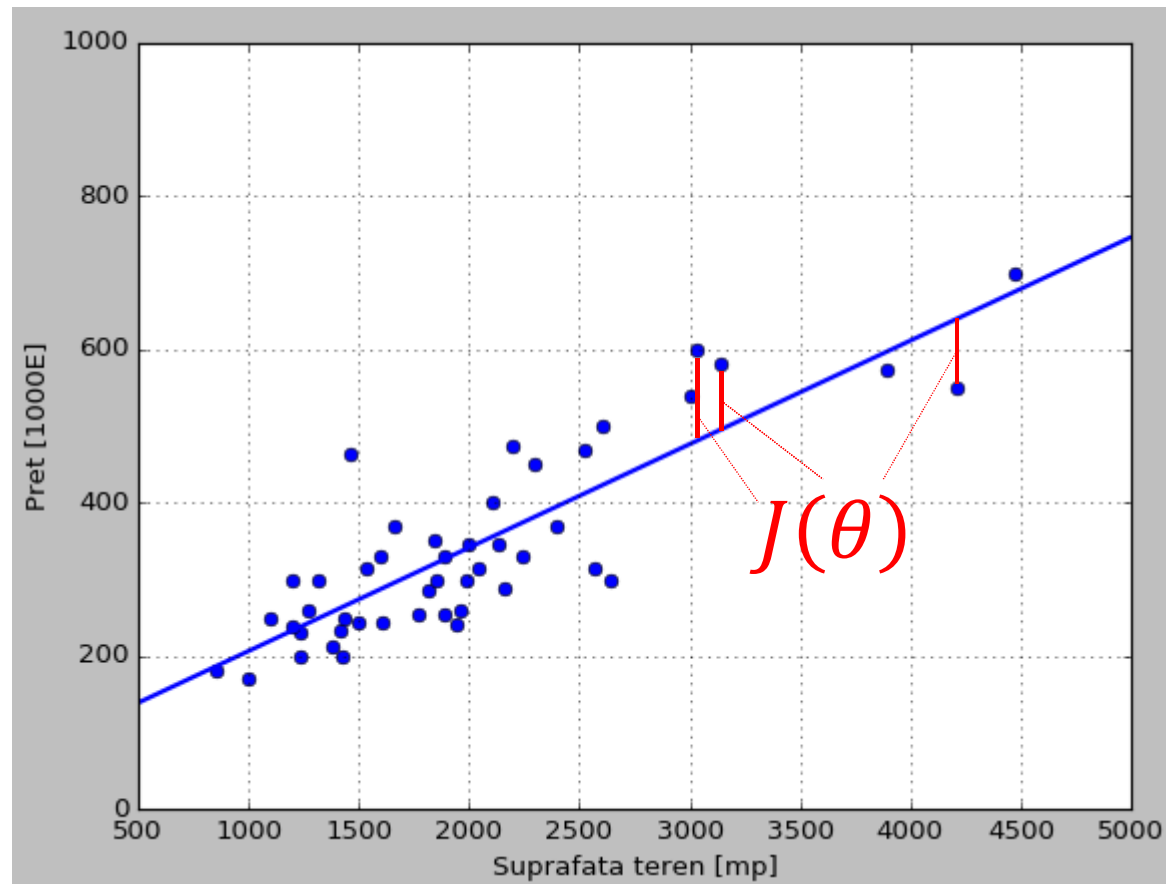
- $J(\theta)$  – funcție de cost (eroarea pătratică) exprimă cât de bine sunt approximate exemplele de antrenare de către  $h_\theta(x)$

$$J(\theta) = \frac{1}{2} \frac{1}{m} \sum_{i=1}^m [h_\theta(x^{(i)}) - y^{(i)}]^2$$

- Obiectivul algoritmului de antrenare este minimizarea funcției  $J(\theta)$ :

$$\underset{\theta}{\text{minimize}} J(\theta)$$

# Funcția de cost



# Minimizarea gradientului (metoda batch)

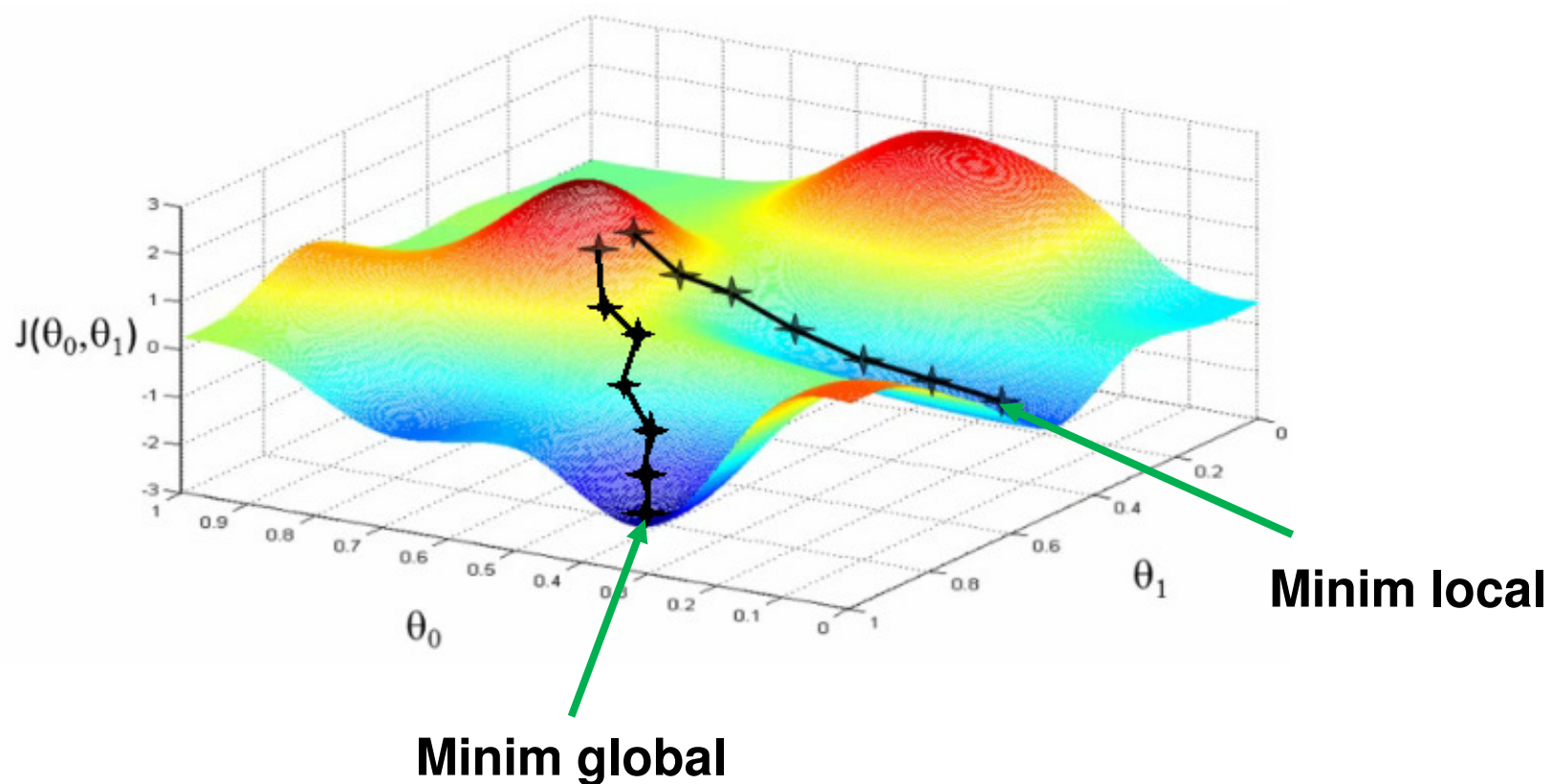


- **Obiectiv:** minimize  $J(\theta)$
- **Inițial valorile pentru  $\theta$  sunt aleatorii (ex.  $\theta = \vec{0} = [0 \ 0 \ \dots \ 0]^T$ )**
- **Modifică  $\theta$  până când  $J(\theta)$  ajunge la o valoare minimă**
  - **Intuiție:** eroarea minimă se găsește în punctul unde derivata funcției  $J(\theta)$  devine zero  $\frac{\partial}{\partial \theta_i} J(\theta) = 0$
  - **Pot exista mai multe puncte în care derivata devine zero (minime locale)**



# Minimizarea gradientului

- **Obiectiv:** minimize  $J(\theta)$





# Minimizarea gradientului

- **Obiectiv:** minimize  $J(\theta)$
- **Noile valori ale parametrilor  $\theta$ :**

$$\theta_i := \theta_i - \alpha \frac{\partial}{\partial \theta_i} J(\theta)$$

$a := b$   
↓  
Operator  
de atribuire

$a = b$   
↓  
Operator  
de egalitate



## Un singur exemplu de antrenare

$$J(\theta) = \frac{1}{2} \frac{1}{m} \sum_{i=1}^m [h_{\theta}(x^{(i)}) - y^{(i)}]^2$$

$$h_{\theta}(x) = \sum_{i=0}^n \theta_i x_i = \theta^T x$$

$$\frac{\partial}{\partial \theta_i} J(\theta) = \frac{\partial}{\partial \theta_i} \frac{1}{2} [h_{\theta}(x) - y]^2 =$$

$$= 2 \frac{1}{2} [h_{\theta}(x) - y] \cdot \frac{\partial}{\partial \theta_i} [h_{\theta}(x) - y] =$$

$$= [h_{\theta}(x) - y] \frac{\partial}{\partial \theta_i} (\theta_0 x_0 + \dots + \theta_i x_i + \dots + \theta_n x_n - y) =$$

$$= [h_{\theta}(x) - y] \cdot x_i$$

**Singurul termen  
care depinde de  $\theta_i$**



# Minimizarea gradientului

- Update parametrii:

$$\theta_i := \theta_i - \alpha \cdot [h_{\theta}(x) - y] \cdot x_i$$

- $\alpha$  – rata de învățare (learning rate):
  - controlează cât de puternic se modifică parametrii  $\theta_i$  între două iterații
  - de obicei se alege manual
  - $\alpha$  prea mic: convergență înceată
  - $\alpha$  prea mic: ”overhooting”



# Minimizarea gradientului

- Pentru mai multe exemple de antrenare
- Repetă până la convergență:

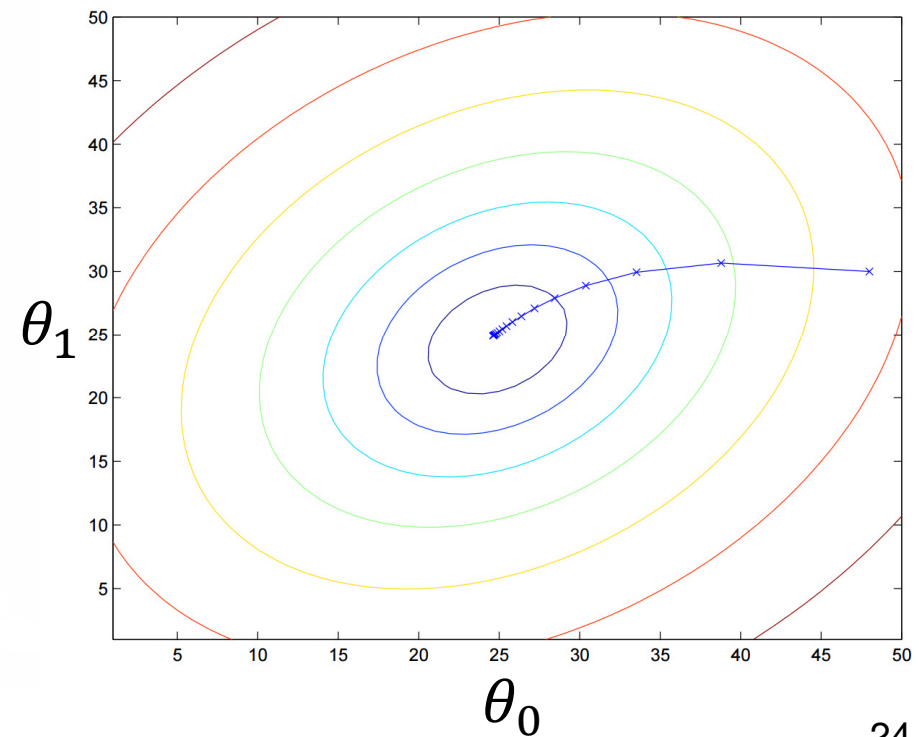
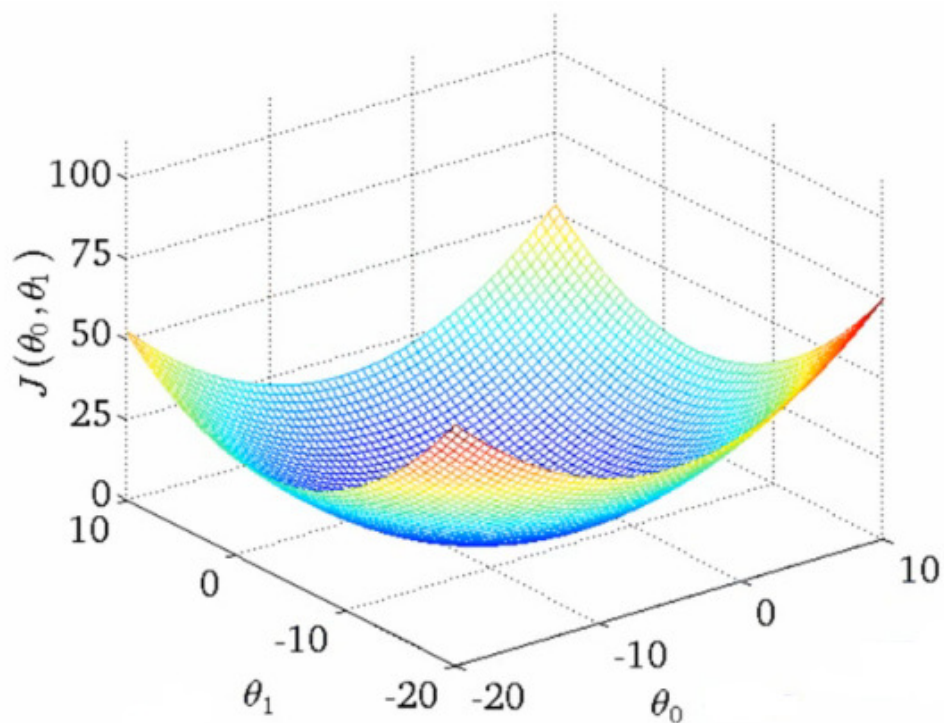
$$\theta_i := \theta_i - \alpha \cdot \sum_{j=1}^m [h_{\theta}(x^{(j)}) - y^{(j)}] \cdot x_i^{(j)}$$

$$\frac{\partial}{\partial \theta_i} J(\theta)$$



# Minimizarea gradientului

- Pe măsură ce ne apropiem de minim, algoritmul va efectua pași din ce în ce mai mici (gradientul (derivata) devine din ce în ce mai mic)

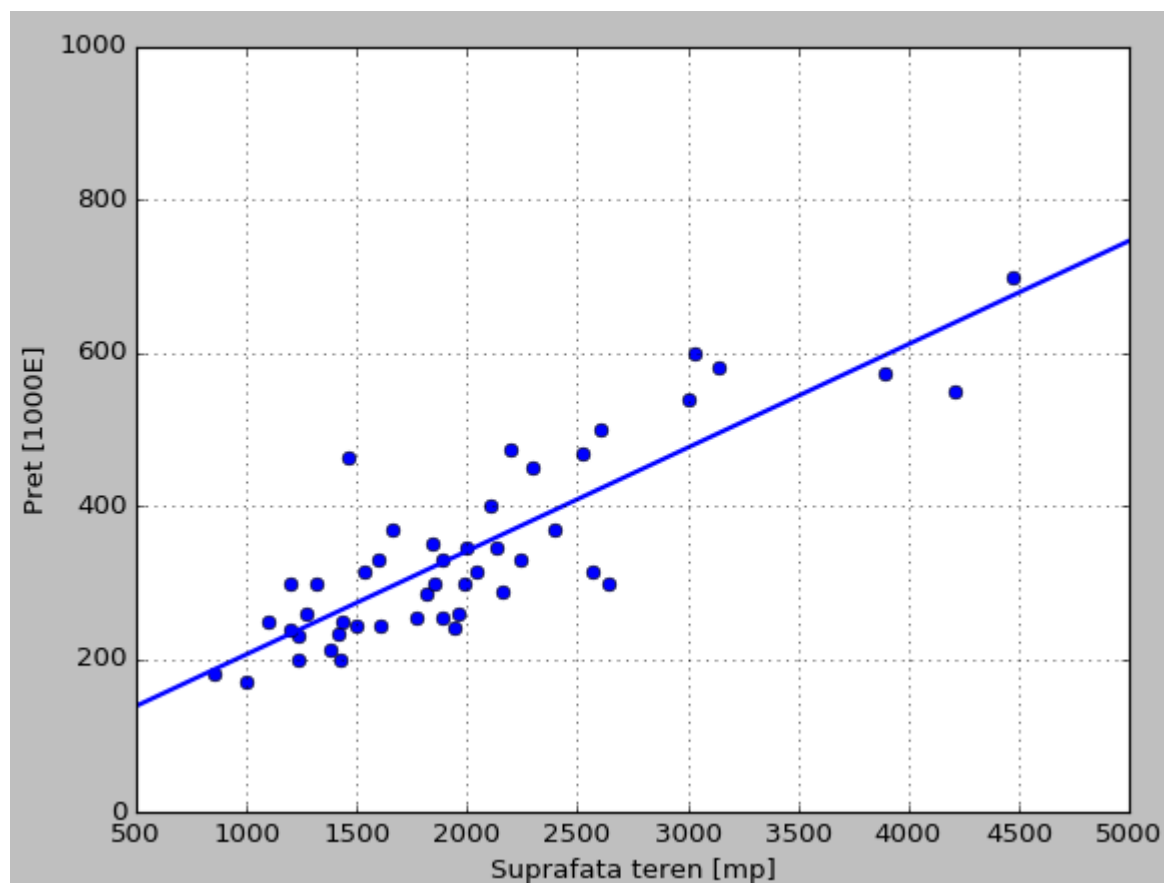




# Aproximare liniară

- $h_{\theta}(x)$  – funcție liniară

$$h_{\theta}(x) = \theta_0 + \theta_1 x$$





# Minimizarea stochastică a gradientului

- Când numărul de exemple de antrenare devine foarte mare (ex. 1 mil de exemple), antrenarea prin metoda batch devine greoaie (parametrii  $\theta$  sunt adaptați doar după calculul gradientului pe întregul set de exemple)
- Minimizarea stochastică a gradientului modifică parametrii  $\theta$  cu fiecare exemplu de antrenare:

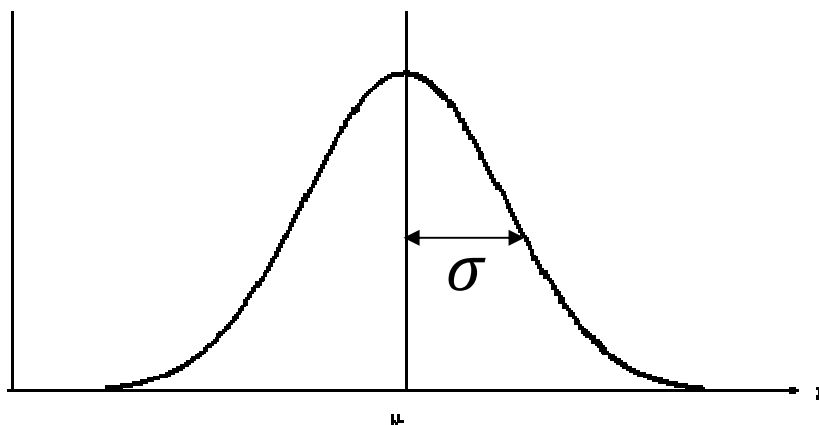
```
Repeat {  
  For  $j = 1$  to  $m$  {  
     $\theta_i := \theta_i - \alpha \cdot [h_{\theta}(x^{(j)}) - y^{(j)}] \cdot x_i^{(j)}$   
  }  
}
```



# Regresie liniară: interpretare probabilistică

- **Fie**  $y^{(i)} = \theta^T x^{(i)} + \varepsilon^{(i)}$
- $\varepsilon^{(i)}$  – termen de eroare (modelează defectele: zgomot aleatoriu)
- $\varepsilon^{(i)} \sim \mathcal{N}(0, \sigma^2)$  – zgomot Gaussian normal distribuit

$$P(\varepsilon^{(i)}) = \frac{1}{\sqrt{2\pi}\sigma} \cdot \exp\left(-\frac{(\varepsilon^{(i)})^2}{2\sigma^2}\right)$$





# Regresie liniară: interpretare probabilistică

- $P(y^{(i)} | x^{(i)}; \theta) \sim \mathcal{N}(\theta^T x^{(i)}, \sigma^2)$  – probabilitatea de apariție a valorii  $y^{(i)}$  dându-se  $x^{(i)}$  și parametrii  $\theta$

$$P(y^{(i)} | x^{(i)}; \theta) = \frac{1}{\sqrt{2\pi}\sigma} \cdot \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right)$$

- Deseori erorile sunt generate de către o distribuție Gaussiană
- Se presupune că  $\varepsilon^{(i)}$  sunt variabile *Independente și Identic Distribuite* (IID): provin de la aceeași distribuție Gaussiană cu aceeași varianță



## Likelihood parametrului $\theta$ : $L(\theta)$

$$L(\theta) = P(\vec{y} | x; \theta)$$

- $L(\theta)$  – vizualizăm  $P(\vec{y} | x; \theta)$  ca o funcție dependentă de  $\theta$ , păstrând fix  $x$  și  $y$ 
  - Probabilitatea parametrilor  $\theta$  și a datelor  $(x, y)$

$$\begin{aligned} L(\theta) = P(\vec{y} | x; \theta) &= \prod_{i=1}^m P(y^{(i)} | x^{(i)}; \theta) = \\ &= \prod_{i=1}^m \frac{1}{\sqrt{2\pi}\sigma} \cdot \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right) \end{aligned}$$



# Maximum Likelihood Estimation

- **Obiectiv: determinarea lui  $\theta$  maximizând cât mai mult posibil probabilitatea de existență a datelor**
- **Determină  $\theta$  în așa fel încât  $L(\theta) = P(\vec{y} | x; \theta)$  să fie maxim**

$$\begin{aligned}l(\theta) &= \log L(\theta) = \\ &= \log \prod_{i=1}^m \frac{1}{\sqrt{2\pi}\sigma} \exp(\dots) = \sum_{i=1}^m \log \left[ \frac{1}{\sqrt{2\pi}\sigma} \exp(\dots) \right] = \\ &= m \log \frac{1}{\sqrt{2\pi}\sigma} + \sum_{i=1}^m -\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\end{aligned}$$



# Maximum Likelihood Estimation

- **Compuțațional, este mai ușoară maximizarea lui  $l(\theta)$ , în locul lui  $L(\theta)$  (adunari în locul înmulțirilor)**
- **Maximizarea lui  $l(\theta)$  este același lucru cu minimizarea funcției pătratice  $J(\theta)$ :**

$$J(\theta) = \frac{\sum_{i=1}^m (y^{(i)} - \theta^T x^{(i)})^2}{2}$$

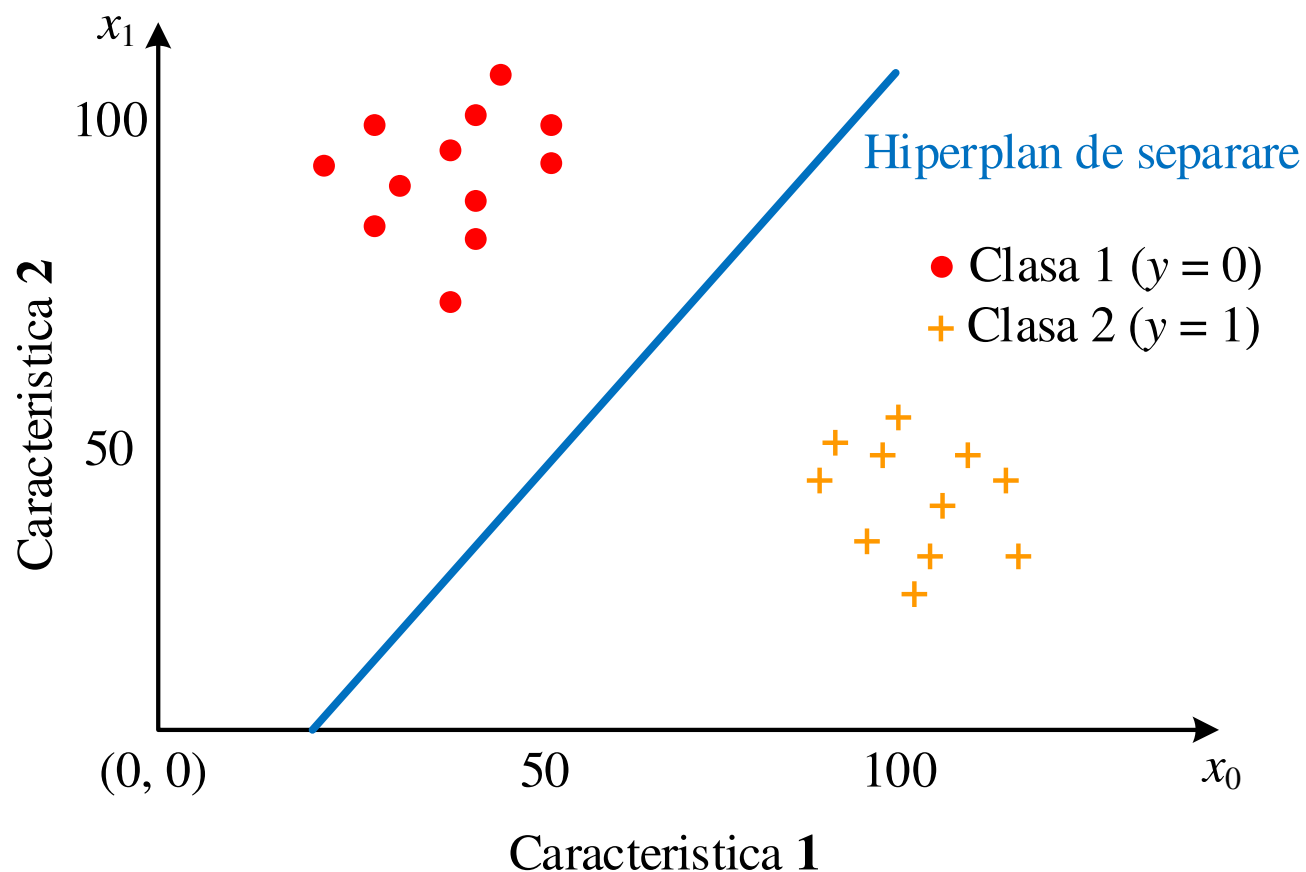


**în acest model  
varianța  $\sigma^2$  nu este  
luată în considerare**



# Clasificarea binară $y \in \{0,1\}$

- Obiectiv: determinarea hiperplanului optim de separare între cele două clase

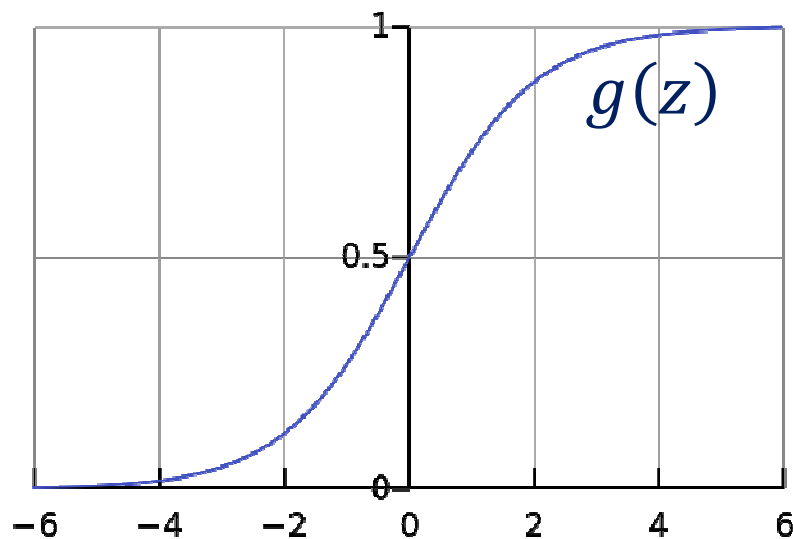




# Clasificarea binară $y \in \{0,1\}$

- Ipoteza  $h_{\theta}(x) \in [0,1]$  va indica o probabilitate
- Funcția sigmoid sau logistică

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





## Clasificarea binară $y \in \{0,1\}$

- **Probabilitatea ca  $y = 1$ :**  $P(y = 1 | x; \theta) = h_{\theta}(x)$
- **Probabilitatea ca  $y = 0$ :**  $P(y = 0 | x; \theta) = 1 - h_{\theta}(x)$

$$P(y | x; \theta) = h_{\theta}(x)^y \cdot (1 - h_{\theta}(x))^{(1-y)}$$

$$\begin{aligned} L(\theta) &= P(\vec{y} | x; \theta) = \prod_{i=1}^m P(y^{(i)} | x^{(i)}; \theta) = \\ &= \prod_{i=1}^m h(x^{(i)})^{y^{(i)}} \left(1 - h(x^{(i)})^{(1-y^{(i)})}\right) \end{aligned}$$



# Maximum Likelihood Estimation

$$l(\theta) = \log L(\theta)$$

$$l(\theta) = \sum_{i=1}^m y^{(i)} \log h(x^{(i)}) + (1 - y^{(i)}) \log(1 - h(x^{(i)}))$$

- Maximizarea gradientului:

$$\theta := \theta + \alpha \cdot \nabla_{\theta} l(\theta)$$

$$\frac{\partial}{\partial \theta_j} l(\theta) = \sum_{i=1}^m (y^{(i)} - h(x^{(i)})) \cdot x_j^{(i)}$$



## Distribuția Bernoulli $\phi$

- Distribuția Bernoulli  $\phi$  modelează variabilele aleatorii ce pot avea doar două valori

$$P(y = 1; \phi) = \phi$$

$$h_{\theta}(x) = E[y | x; \theta] = P(y = 1; \phi)$$

$$= \phi$$

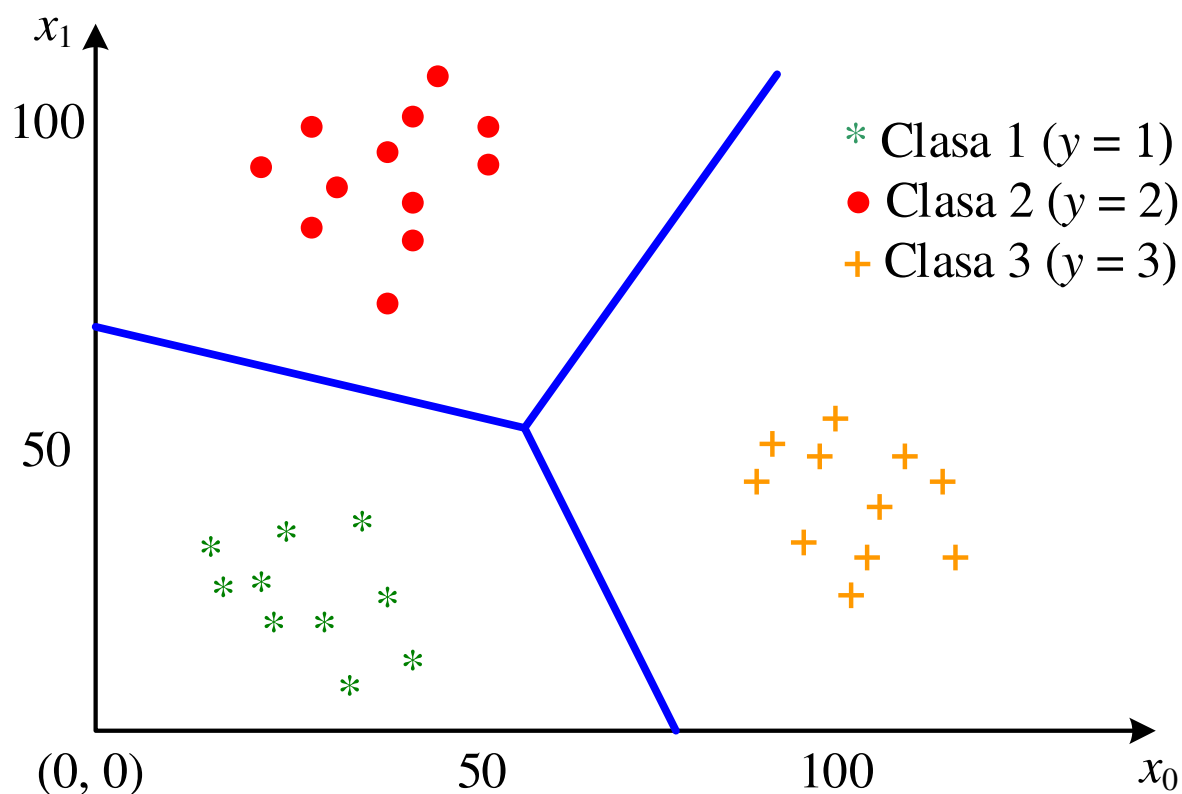
$$= \frac{1}{1 + e^{-\theta^T x}}$$



# Clasificarea multclasă

- $k$  posibile clase de obiecte pentru variabila de ieșire  $y$

$$y \in \{1, 2, \dots, k\}$$





# Clasificarea multclasă

- Parametrii:  $\phi_1, \phi_2, \dots, \phi_{k-1}$

$$P(y = i) = \phi_i$$

- Ultimul parametru din listă:  $\phi_k = 1 - (\phi_1 + \phi_2 + \dots + \phi_{k-1})$

$$T(y) \in \mathbb{R}^{k-1}$$

$$T(1) = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad T(2) = \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix} \quad T(k-1) = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \quad T(k) = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$



# Clasificarea multclasă

- **Funcție indicator:**

$$1\{\text{Adevărat}\} = 1$$

$$1\{\text{Fals}\} = 0$$

$$1\{2 = 3\} = 0$$

$$1\{1 + 1 = 2\} = 1$$

$T(y)_i$  – **elementul  $i$  al vectorul  $T(y)$**

$$T(y)_i = 1\{y = i\}$$

$$P(y) = \phi_1^{1\{y=1\}} \phi_2^{1\{y=2\}} \dots \phi_k^{1\{y=k\}} =$$

$$= \phi_1^{T(y)_1} \phi_2^{T(y)_2} \dots \phi_{k-1}^{T(y)_{k-1}} \phi_k^{1 - \sum_{j=1}^{k-1} T(y)_j}$$



## Clasificarea multclasă

$$\phi_i = \frac{e^{\theta_i^T x}}{1 + \sum_{j=1}^{k-1} e^{\theta_j^T x}}$$

$$h_{\theta}(x) = E[T(y) \mid x; \theta]$$

$$= E \left[ \begin{array}{c} 1\{y = 1\} \\ \vdots \\ 1\{y = k - 1\} \end{array} \mid x; \theta \right] = \begin{bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_{k-1} \end{bmatrix}$$

$1\{y = 1\}$  – probabilitatea ca  $y = 1$ , dată de  $\phi_1$



# Regresie Softmax

$$h_{\theta}(x) = \begin{bmatrix} \frac{e^{\theta_1^T x}}{1 + \sum_{j=1}^{k-1} e^{\theta_j^T x}} \\ \vdots \\ \frac{e^{\theta_{k-1}^T x}}{1 + \sum_{j=1}^{k-1} e^{\theta_j^T x}} \end{bmatrix}$$

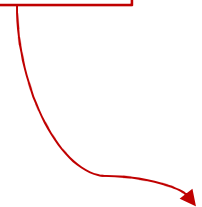
- Generalizare a clasificării binare (doua clase)



# Maximum Likelihood Estimation

- Determinarea parametrilor modelului via  $L(\theta)$

$$\begin{aligned} L(\theta) &= \prod_{i=1}^m P(y^{(i)} \mid x^{(i)}; \theta) \\ &= \prod_{i=1}^m \phi_1^{1\{y=1\}} \phi_2^{1\{y=2\}} \dots \phi_k^{1\{y=k\}} \end{aligned}$$



$$\phi_1 = \frac{e^{\theta_1^T x}}{1 + \sum_{j=1}^{k-1} e^{\theta_j^T x}}$$

$$\theta_1, \theta_2, \dots, \theta_{k-1} \in \mathbb{R}^{n+1}$$

# Maximum log-Likelihood Estimation



$$\begin{aligned} l(\theta) &= \sum_{i=1}^m P(y^{(i)} | x^{(i)}; \theta) \\ &= \sum_{i=1}^m \log \prod_{l=1}^k \left( \frac{e^{\theta_l^T x^{(i)}}}{1 + \sum_{j=1}^k e^{\theta_j^T x^{(i)}}} \right)^{1\{y=l\}} \end{aligned}$$



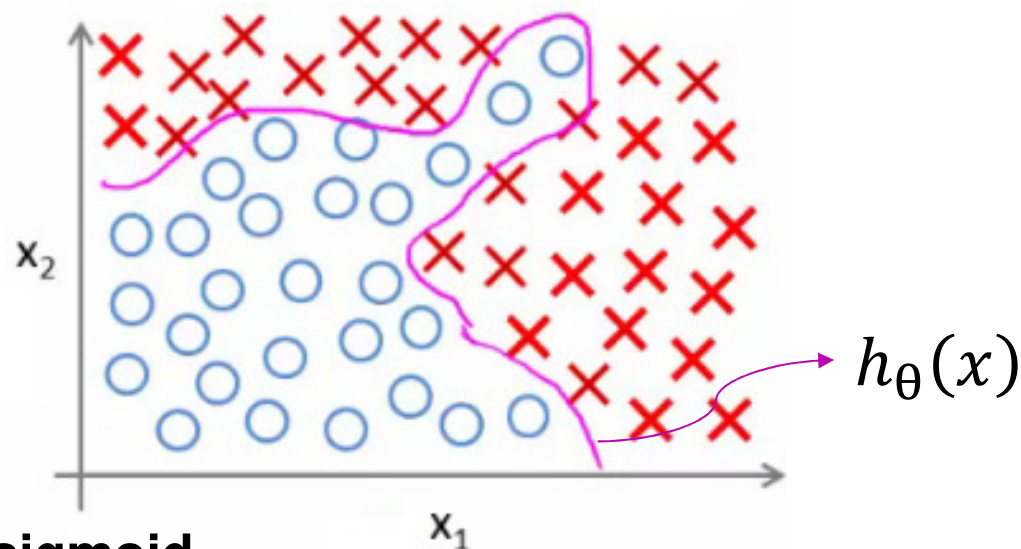
# Rețele Neurale



## Ipoteze $h_{\theta}(x)$ neliniare

- Necesitatea învățării unor ipoteze neliniare  $h_{\theta}(x)$  complexe
- Crearea unui algoritm de regresie logistică complex:

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

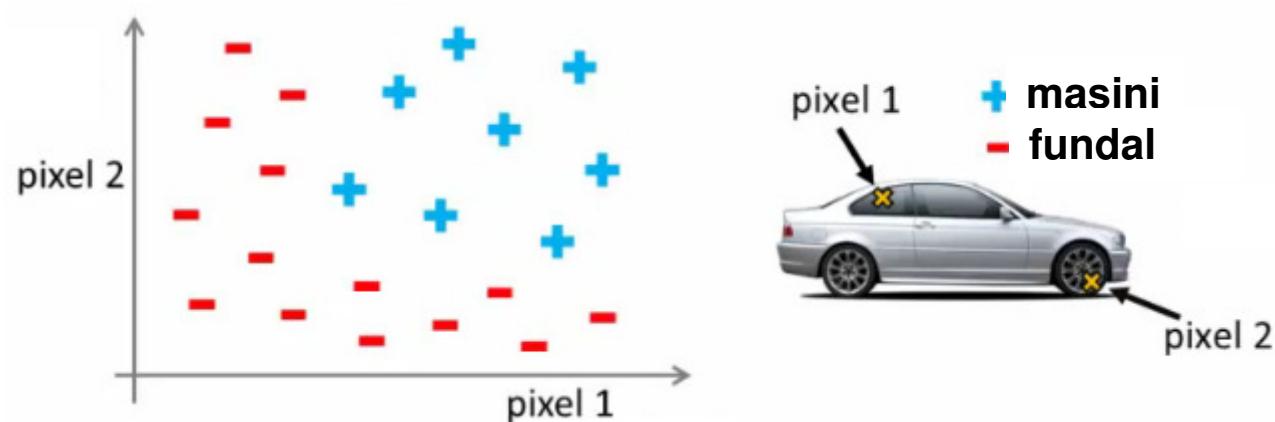


- $g(\cdot)$  – funcție sigmoid
- $g(\cdot)$  foarte dificil de construit pentru un număr mare de features (pericol de overfitting)



## Ipoteze $h_{\theta}(x)$ neliniare

- În recunoașterea formelor din imagini vom avea un număr mare de clase (ex. Imagini gri de dimensiune  $50 \times 50\text{px} \rightarrow n = 2500$ )



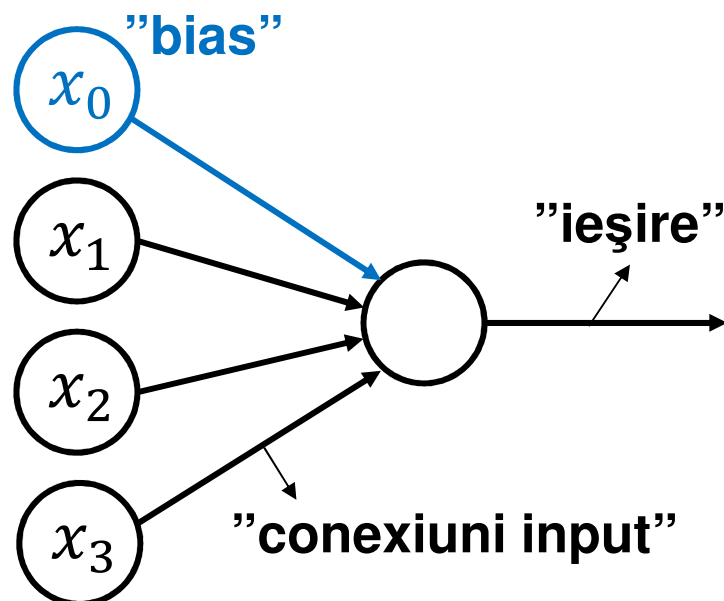
$$x = \begin{bmatrix} \text{intensitatea pixelului 1} \\ \text{intensitatea pixelului 2} \\ \vdots \\ \text{intensitatea pixelului 2500} \end{bmatrix}$$



# Structura unui neuron

- Recent au redevenit populare datorită posibilităților oferite de calculul paralel prin GPU

- Modelul unui neuron:



$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$

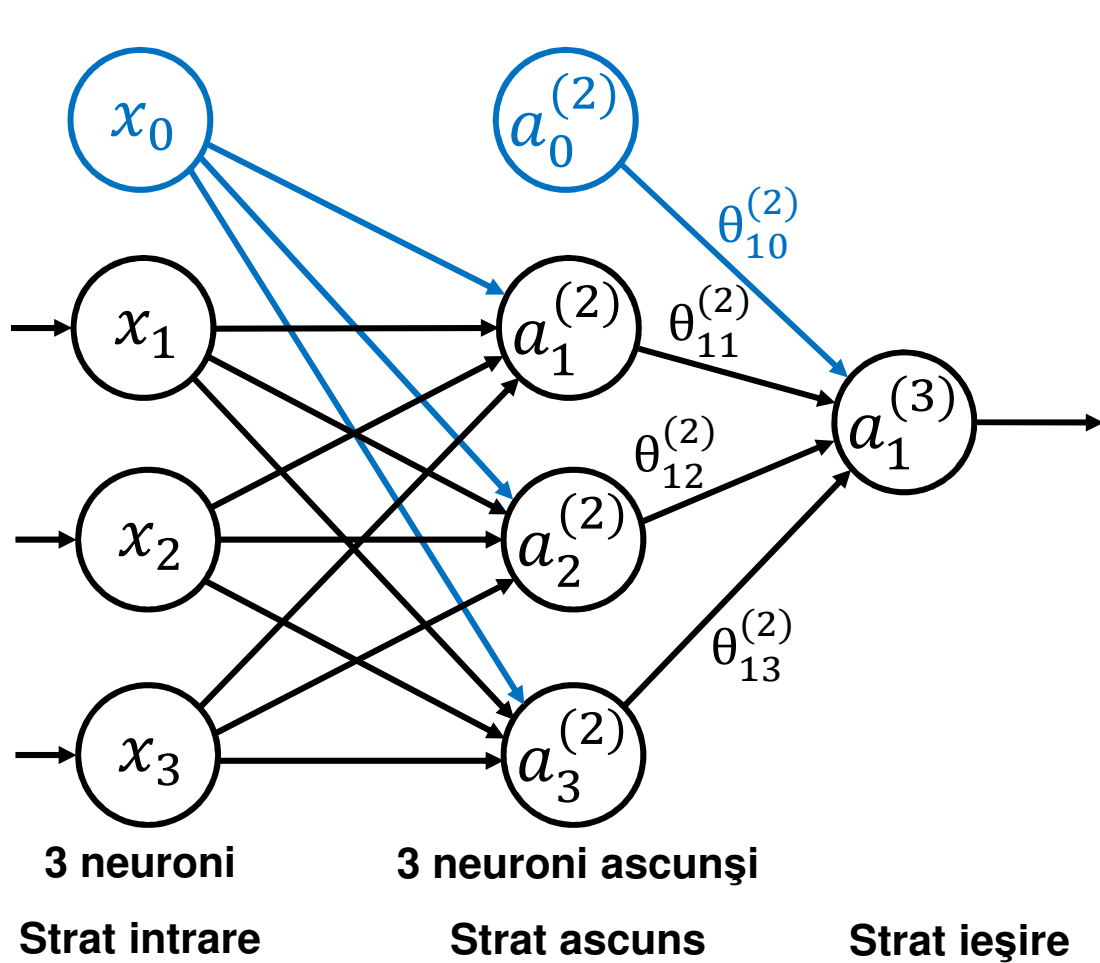
$$g(x) = \frac{1}{1 + e^{-\theta^T x}}$$

$x_0 = 1$  (de obicei)



# Rețea Neurală

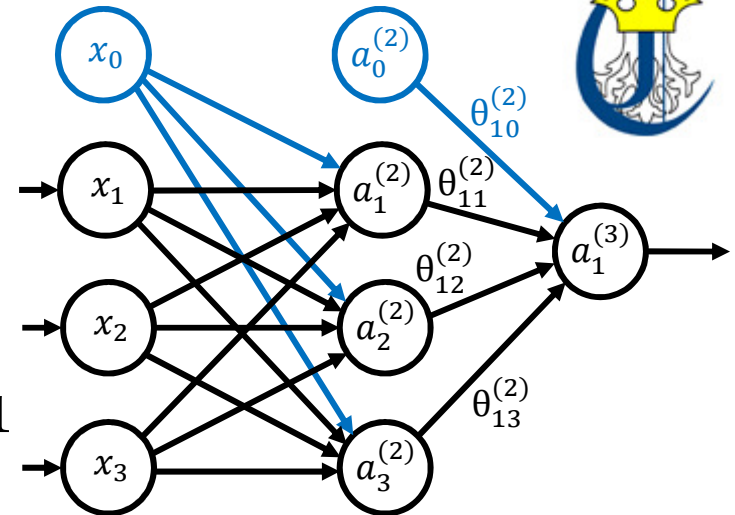
- $\theta$  – parametri, sau ponderi



$$x_0 = 1$$
$$a_0^{(2)} = 1$$

# Rețea Neurală

- $a_i^{(j)}$  – “activarea” neuronului  $i$  din layer-ul  $j$
- $\theta^{(j)}$  – matricea ponderilor ce controlează funcția de mapare din stratul  $j$  în stratul  $j + 1$



$$a_1^{(2)} = g \left( \theta_{10}^{(1)} x_0 + \theta_{11}^{(1)} x_1 + \theta_{12}^{(1)} x_2 + \theta_{13}^{(1)} x_3 \right)$$

$$a_2^{(2)} = g \left( \theta_{20}^{(1)} x_0 + \theta_{21}^{(1)} x_1 + \theta_{22}^{(1)} x_2 + \theta_{23}^{(1)} x_3 \right)$$

$$a_3^{(2)} = g \left( \theta_{30}^{(1)} x_0 + \theta_{31}^{(1)} x_1 + \theta_{32}^{(1)} x_2 + \theta_{33}^{(1)} x_3 \right)$$

$$h_{\theta}(x) = a_1^{(3)} = g \left( \theta_{10}^{(2)} a_0^{(2)} + \theta_{11}^{(2)} a_1^{(2)} + \theta_{12}^{(2)} a_2^{(2)} + \theta_{13}^{(2)} a_3^{(2)} \right)$$

$$\Theta^{(1)} \in \mathbb{R}^{3 \times 4}$$



## Rețea Neurală

- Dacă rețeaua neurală are  $s_j$  neuroni în stratul  $j$  și  $s_{j+1}$  neuroni în stratul  $j + 1$ , atunci  $\Theta^{(j)}$  va avea dimensiunea:

$$s_{j+1} \times (s_j + 1)$$

- $h_{\Theta}(x)$  – variind  $\Theta$  (diferiți parametrii), obținem diferite funcții de mapare între variabilele de intrare  $x$  și variabila de ieșire  $y$



# Reprezentarea vectorială

$$z^{(2)} = \Theta^{(1)} x$$

$$a_0^{(2)} = 1$$

$$a_1^{(2)} = g(z_1^{(2)})$$

$$a_2^{(2)} = g(z_2^{(2)})$$

$$a_3^{(2)} = g(z_3^{(2)})$$

$$z^{(3)} = \Theta^{(2)} a^{(2)}$$

$$h_{\theta}(x) = a^{(3)} = g(z^{(3)})$$

$$\begin{aligned} a_1^{(2)} &= g(\theta_{10}^{(1)} x_0 + \theta_{11}^{(1)} x_1 + \theta_{12}^{(1)} x_2 + \theta_{13}^{(1)} x_3) \\ a_2^{(2)} &= g(\theta_{20}^{(1)} x_0 + \theta_{21}^{(1)} x_1 + \theta_{22}^{(1)} x_2 + \theta_{23}^{(1)} x_3) \\ a_3^{(2)} &= g(\theta_{30}^{(1)} x_0 + \theta_{31}^{(1)} x_1 + \theta_{32}^{(1)} x_2 + \theta_{33}^{(1)} x_3) \end{aligned}$$

$\Theta^{(1)}$  (blue arrow pointing to the first three rows)

$$h_{\theta}(x) = a_1^{(3)} = g(\theta_{10}^{(2)} a_0^{(2)} + \theta_{11}^{(2)} a_1^{(2)} + \theta_{12}^{(2)} a_2^{(2)} + \theta_{13}^{(2)} a_3^{(2)} + 1)$$

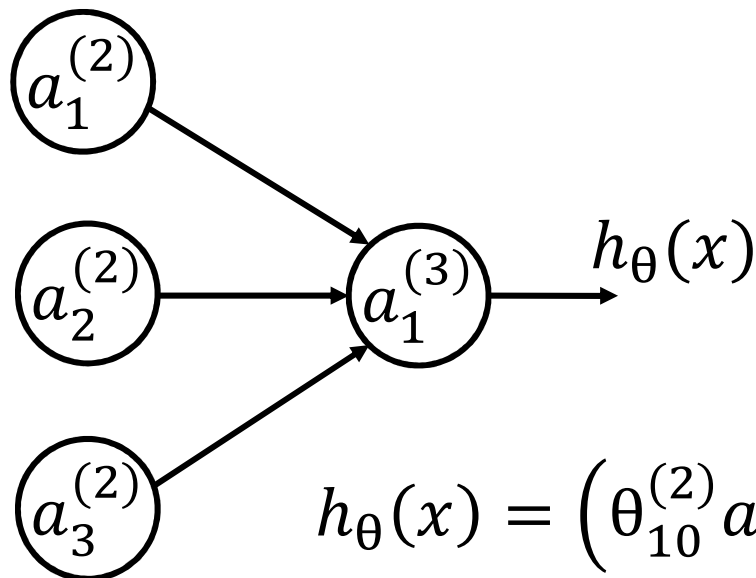
$\Theta^{(2)}$  (blue arrow pointing to the first four terms),  $z^{(3)}$  (green arrow pointing to the last three terms)

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad z^{(2)} = \begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \\ z_3^{(2)} \end{bmatrix}$$



# Învățarea propriilor features

- Similar regresiei logistice; în locul utilizării feature-urilor originale  $(x_1, x_2, x_3)$ , rețeaua neurală folosește noile feature-uri învățate  $(a_1^{(2)}, a_2^{(2)}, a_3^{(2)})$

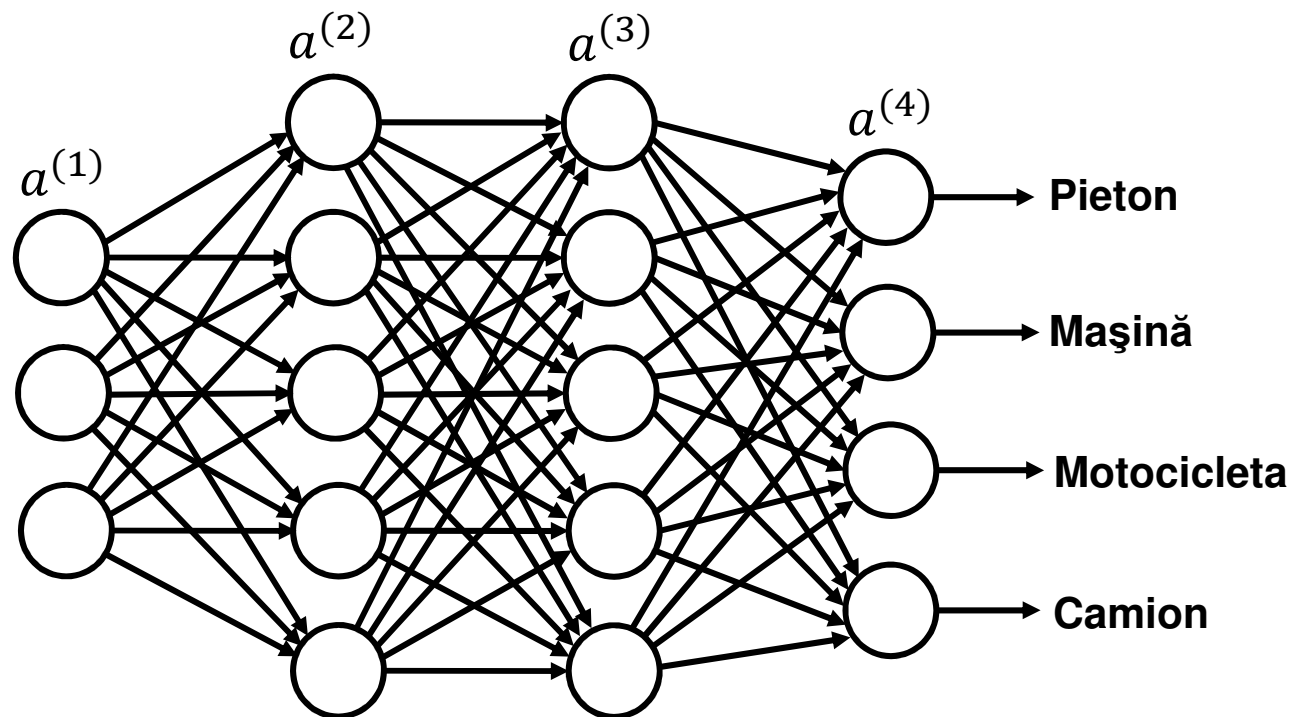


$$h_{\theta}(x) = \left( \theta_{10}^{(2)} a_0^{(2)} + \theta_{11}^{(2)} a_1^{(2)} + \theta_{12}^{(2)} a_2^{(2)} + \theta_{13}^{(2)} a_3^{(2)} \right)$$

Stratul 2



# Clasificarea multclasă



$$h_{\theta}(x) \in \mathbb{R}^4$$

$$h_{\theta}(x) \sim \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Pieton

$$h_{\theta}(x) \sim \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

Mașină

$$h_{\theta}(x) \sim \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

Motocicleta

$$h_{\theta}(x) \sim \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Camion



# Clasificarea multclasă

- Clasa  $y^{(i)}$  a datelor de antrenare  $\{(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)})\}$  are una din formele:

$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Pieton

$$\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

Mașină

$$\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

Motocicleta

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Camion

- ieșirea rețelei neurale:

$$h_{\theta}(x^{(i)}) \sim y^{(i)}$$



# Funcția de cost în clasificarea binară

## *Regresie Logistică*

$$J(\theta) = -\frac{1}{m} \left[ \sum_{i=1}^m y^{(i)} \log h_{\theta}(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_{\theta}(x^{(i)})) \right] + \frac{\lambda}{2m} \sum_{j=1}^n \theta_j^2$$



# Funcția de cost în clasificarea multclasă

## *Rețele Neurale*

- $L$  – numărul total de straturi ale rețelei
- $s_l$  – numărul de neuroni (fără neuronul bias) al stratului  $l$
- $k$  clase  $\rightarrow k$  neuroni de ieșire

$$h_{\theta}(x) \in \mathbb{R}^k \quad k \geq 3 \quad s_l = k$$

- Funcția de cost este o generalizare a funcției de cost utilizată în regresia logistică (fără regularizare a termenului bias  $\theta_0$ )

# Funcția de cost în clasificarea multclasă

## Rețele Neurale



- ieșirea rețelei sunt vectori în domeniul  $\mathbb{R}^K$  ( $K = 1$  pentru clasificarea binară)
- $(h_{\theta}(x))_i$  – ieșirea  $i$  (ieșirea neuronului  $i$ ); elementul  $i$  al vectorului de ieșire din domeniul  $\mathbb{R}^K$

$$J(\theta) = -\frac{1}{m} \left[ \sum_{i=1}^m \sum_{k=1}^K y_k^{(i)} \log h_{\theta}(x^{(i)})_k + (1 - y_k^{(i)}) \log (1 - h_{\theta}(x^{(i)})_k) \right] +$$
$$+ \frac{\lambda}{2m} \sum_{l=1}^{L-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} (\theta_{ji}^{(l)})^2$$

$$\text{Ex: } y_k = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

# Funcția de cost în clasificarea multclasă

## Rețele Neurale



Sumă de-a lungul celor  $K$  neuroni de ieșire

$$J(\theta) = -\frac{1}{m} \left[ \sum_{i=1}^m \sum_{k=1}^K y_k^{(i)} \log h_{\theta}(x^{(i)})_k + (1 - y_k^{(i)}) \log (1 - h_{\theta}(x^{(i)})_k) \right] +$$
$$+ \frac{\lambda}{2m} \sum_{l=1}^{L-1} \sum_{i=1}^{s_l} \sum_{j=1}^{s_{l+1}} (\theta_{ji}^{(l)})^2$$

**Termen de regularizare**

**Nu se iau în considerare neuronii de bias ( $i = 0$ )**



# Antrenarea unei rețele neurale

- **Obiectiv: găsirea parametrilor  $\theta$  ce minimizează  $J(\theta)$ :**

$$\min_{\theta} J(\theta)$$

- **Pentru găsirea parametrilor  $\theta$  (prin minimizarea gradientului) este necesar calculul funcției de cost, împreună cu derivatele sale parțiale:**

$$J(\theta); \quad \frac{\partial}{\partial \theta_{ij}^{(l)}} J(\theta)$$

- $\theta_{ij}^{(l)} \in \mathbb{R}$  – un parametru al rețelei neurale
- **Propagarea înainte:**



# Propagarea înainte într-o rețea neurală

- Luăm în considerare un singur exemplu de antrenare  $(x, y)$
- Propagarea înainte:

$$a^{(1)} = x \quad (\text{stratul de intrare})$$

$$z^{(2)} = \theta^{(1)} a^{(1)}$$

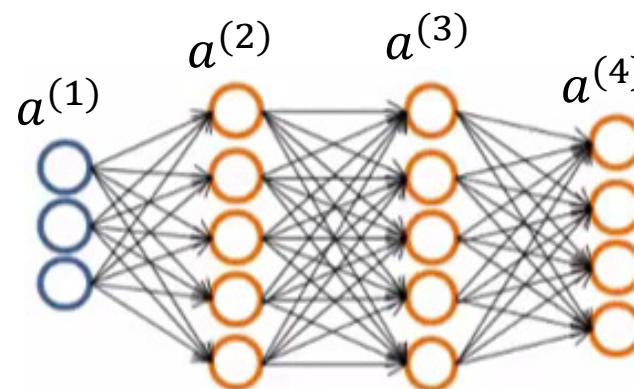
$$a^{(2)} = g(z^{(2)}) \quad (\text{adaugă } a_0^{(2)} \text{ ca și bias})$$

$$z^{(3)} = \theta^{(2)} a^{(2)}$$

$$a^{(3)} = g(z^{(3)}) \quad (\text{adaugă } a_0^{(3)} \text{ ca și bias})$$

$$z^{(4)} = \theta^{(3)} a^{(3)}$$

$$a^{(4)} = h_{\theta}(x) = g(z^{(4)})$$





# Calculul gradientului prin *Backpropagation*

- Algoritmul de propagare înapoi a erorii este utilizat pentru calculul

derivatelor parțiale  $\frac{\partial}{\partial \theta_{ij}^{(l)}} J(\theta)$

- **Intuiție backpropagation:** pentru fiecare nod vom calcula un termen

$\delta_j^{(l)}$  ce reprezintă eroarea neuronului  $j$  din stratul  $l$

- $a_j^{(l)}$  – activarea neuronului  $j$  din stratul  $l$

- $\delta_j^{(l)}$  va captura eroarea din activarea  $a_j^{(l)}$  a neuronului  $j$  din stratul  $l$



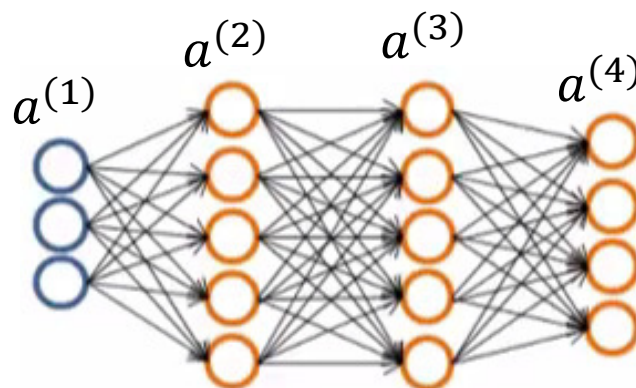
# Calculul gradientului prin *Backpropagation*

- Exemplu pentru o rețea cu 4 neuroni de ieșire
- Pentru fiecare neuron de ieșire ( $L = 4$ ):

$$\delta_j^{(4)} = \boxed{a_j^{(4)}} - y_j$$

*h<sub>θ</sub>(x)<sub>j</sub>*

- $y_j$  – elementul  $j$  al vectorului  $y$  din setul de date de antrenare





# Propagarea înapoi a erorii

- Varianta vectorizată:

$$\delta^{(4)} = a^{(4)} - y$$

- $\delta$  în straturile anterioare ale rețelei:

$$\delta^{(3)} = \left( (\theta^{(3)})^T \delta^{(4)} \right) .* g'(z^{(3)})$$

$$\delta^{(2)} = \left( (\theta^{(2)})^T \delta^{(3)} \right) .* g'(z^{(2)})$$

- $.*$  – multiplicare la nivel de element

- $g'(z^{(3)})$  – derivata funcției de activare  $g$  evaluată la valorile de intrare date de  $z^{(3)}$

$$g'(z^{(3)}) = a^{(3)} .* (1 - a^{(3)})$$

$$g'(z^{(2)}) = a^{(2)} .* (1 - a^{(2)})$$



## Propagarea înapoi a erorii

- Derivata funcției de activare  $g$  în stratul  $l$

$$g'(z^{(l)}) = a^{(l)} .* (1 - a^{(l)})$$

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

$$g'(z) = \frac{\partial g(z)}{\partial z} = - \left( \frac{1}{1 + e^{-z}} \right)^2 \frac{\partial}{\partial z} (1 + e^{-z})$$

- Ecuația completă pentru propagarea înapoi a erorii:

$$\delta^{(l)} = \left( (\theta^{(l)})^T \delta^{(l+1)} \right) .* a^{(l)} .* (1 - a^{(l)})$$



# Propagarea înapoi a erorii

- Putem calcula termenii derivatelor parțiale prin multiplicarea valorilor de activare și a erorilor pentru fiecare exemplu de antrenare

$$\frac{\partial J(\theta)}{\partial \theta_{ij}^{(l)}} = \frac{1}{m} \sum_{t=1}^m a_j^{(t)(l)} \delta_i^{(t)(l+1)}$$

- $\delta^{l+1}$  și  $a^{l+1}$  sunt vectori de dimensiune  $s_{l+1}$
- $a^l$  este un vector de  $s_l$  elemente
- Multiplicând  $a^l$  cu  $\delta^{l+1}$  vom obține o matrice de dimensiune  $s_{l+1} \times s_l$ , ce are aceeași dimensiune cu  $\theta^{(l)}$
- Cu alte cuvinte, procesul produce un termen gradient pentru fiecare element din  $\theta^{(l)}$



## Propagarea înapoi a erorii

- Termenul  $\delta^{(1)}$  este inexistent datorită faptului că el corespunde stratului de intrare (nu dorim să modificăm feature-urile de intrare)
- Propagarea înapoi a erorii de la stratul de ieșire:

$$\delta^{(4)} \rightarrow \delta^{(3)} \rightarrow \delta^{(2)}$$

$$\frac{\partial}{\partial \theta_{ij}^{(l)}} J(\theta) = a_j^{(l)} \delta_i^{(l+1)}$$

( $\lambda = 0$  – fără regularizare)



# Propagarea înapoi a erorii

- Setul de antrenare:  $\{(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)})\}$

- Pentru toți  $i, j, l$ , set:  $\Delta_{ij}^{(l)} = 0$

→ folosiți în calculul  $\frac{\partial}{\partial \theta_{ij}^{(l)}} J(\theta)$

- $\Delta$  – matrice indexată prin  $ij$  ( $\Delta$  este o matrice acumulator pe care o folosim la acumularea termenilor derivatelor parțiale)

- $D_{ij}^{(l)}$  – derivatele parțiale

$$D_{ij}^{(l)} = \frac{\partial J(\theta)}{\partial \theta_{ij}^{(l)}}$$



# Propagarea înapoi a erorii

**For**  $i = 1$  **to**  $m$  {

Set  $a^{(1)} = x^{(i)}$

Calcul  $a^{(l)}$  pentru  $l = 2, 3, \dots, L$  prin propagare înainte

Utilizând  $y^{(i)}$ , se calculează  $\delta^{(L)} = a^{(L)} - y^{(i)}$

Calcul  $\delta^{(L-1)}, \delta^{(L-2)}, \dots, \delta^{(2)}$

$$\Delta_{ij}^{(l)} := \Delta_{ij}^{(l)} + a_j^{(l)} \delta_i^{(l+1)}$$

}

**vectorial:**  $\Delta^{(l)} := \Delta^{(l)} + \delta^{(l+1)} (a^{(l)})^T$

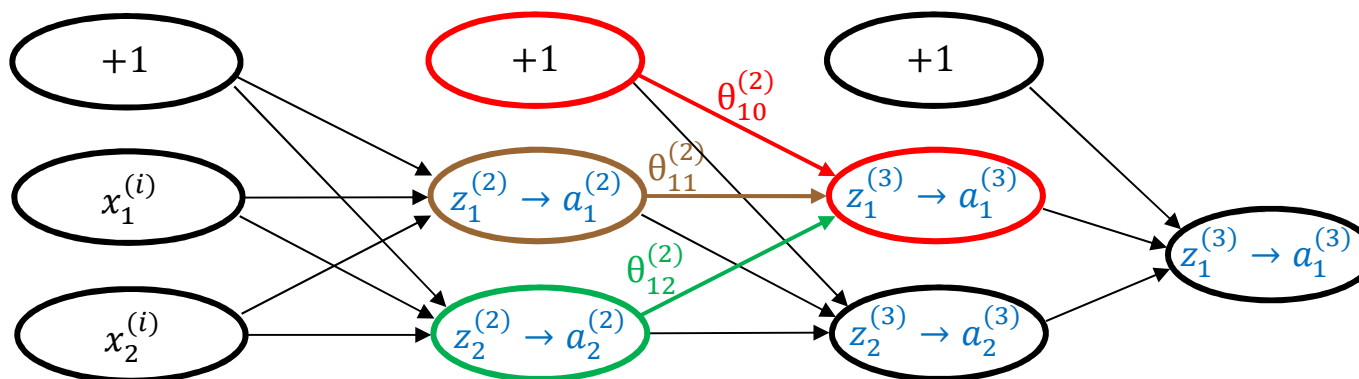
$$D_{ij}^{(l)} := \frac{1}{m} \Delta_{ij}^{(l)} + \lambda \theta_{ij}^{(l)} \quad \text{dacă } j \neq 0$$

$$D_{ij}^{(l)} := \frac{1}{m} \Delta_{ij}^{(l)} \quad \text{dacă } j = 0$$

$j = 0$  – termenul bias



# Propagarea înainte



$$z_1^{(3)} = \theta_{10}^{(2)} \cdot 1 + \theta_{11}^{(2)} a_1^{(2)} + \theta_{12}^{(2)} a_2^{(2)}$$

- Funcția de cost pentru un singur exemplu  $(x^{(i)}, y^{(i)})$ , un singur neuron de ieșire și fără regularizare ( $\lambda = 0$ ):

$$\text{cost}(i) = y^{(i)} \log h_{\theta}(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_{\theta}(x^{(i)}))$$

$$\text{cost}(i) \approx (h_{\theta}(x^{(i)}) - y^{(i)})^2$$



## Propagarea înainte

$$\text{cost}(i) = y^{(i)} \log h_{\theta}(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_{\theta}(x^{(i)}))$$

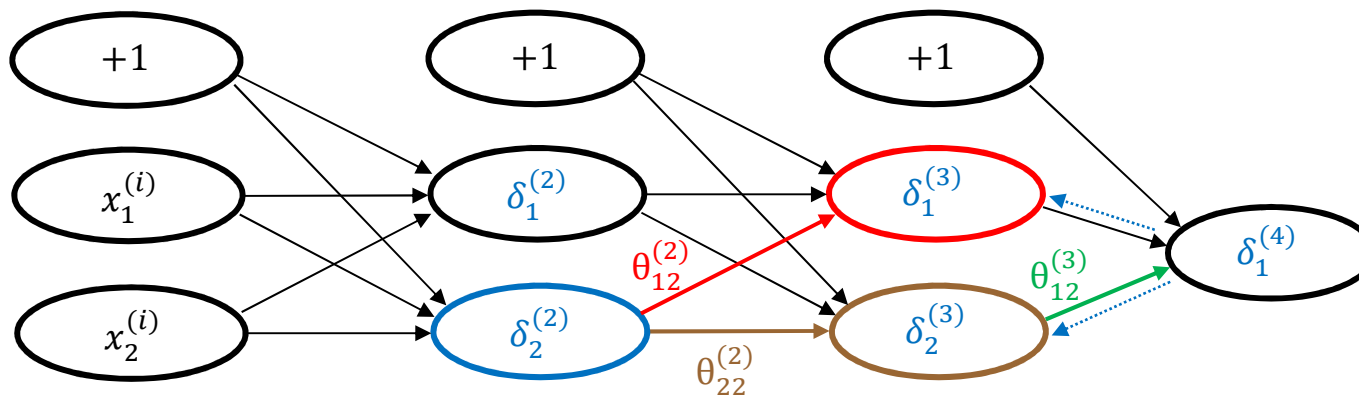
$$\text{cost}(i) \approx (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

- **Funcția de cost ne spune cât de bine estimează rețeaua neurală exemplul  $i$**
- **Derivatele: cât de mult trebuie să modificăm ponderile  $\theta$  ale rețelei neurale:**

$$\delta_1^{(4)} = y^{(i)} - a_1^{(4)}$$



# Propagarea înapoi a erorii



$$\delta_1^{(4)} = y^{(i)} - a_1^{(4)}$$

$$\delta_2^{(2)} = \theta_{12}^{(2)} \delta_1^{(3)} + \theta_{22}^{(2)} \delta_2^{(3)}$$

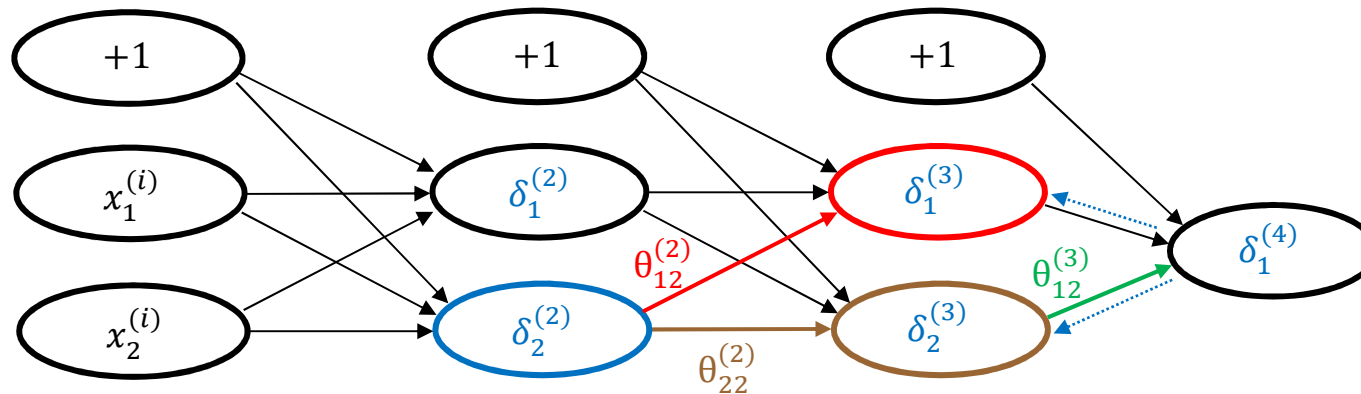
$$\delta_2^{(3)} = \theta_{12}^{(3)} \delta_1^{(4)}$$

- $\delta_j^{(l)}$  - eroarea pentru activarea  $a_j^{(l)}$  a neuronului  $j$  din stratul  $l$

- **Formal:** 
$$\delta_j^{(l)} = \frac{\partial}{\partial z_j^{(l)}} cost(i)$$



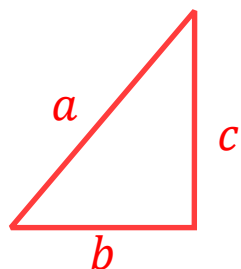
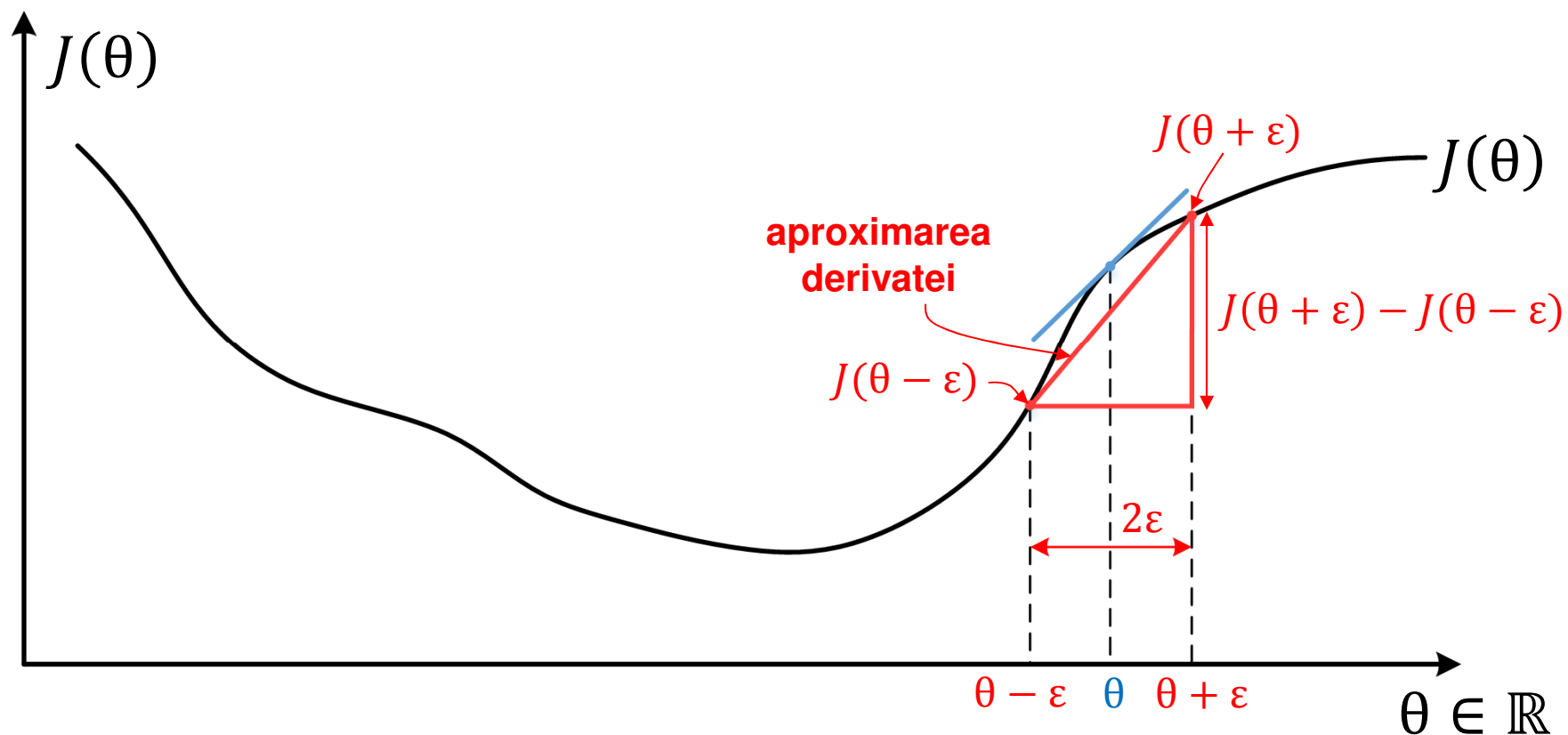
# Propagarea înapoi a erorii



- $\delta$ - cu cât dorim sa modificam ponderile rețelei pentru a schimba valorile intermediare  $z$
- În funcție de modul de implementare al algoritmului de propagare înapoi a erorii, se pot calcula valorile  $\delta$  și pentru neuronii bias
- Parametrii  $\theta$  nu mai sunt vectori, ca în cazul regresiei logistice, ci matrici



# Verificarea gradientului numeric



$$\text{panta}(a) = \frac{c}{b}$$

$$\frac{d}{d\theta} J(\theta) \approx \frac{J(\theta + \epsilon) - J(\theta - \epsilon)}{2\epsilon}$$



## Verificarea gradientului

- Derivata este aproximativ panta tangentei în punctul în care se calculează derivata
- Dacă  $\varepsilon$  este foarte mic, aproximarea devine derivata
- Interpretare adițională a derivatei:

$$\frac{d}{d\theta}J(\theta) \approx \frac{J(\theta + \varepsilon) - J(\theta)}{\varepsilon}$$



# Verificarea gradientului numeric

- Matricea de ponderi  $\theta$  poate fi descompusă sub forma unui vector

$\theta \in \mathbb{R}^n$  ce conține ponderile  $\theta^{(1)}, \theta^{(2)}, \theta^{(3)}$

$$\theta = [\theta_1, \theta_2, \theta_3, \dots, \theta_n]$$

$$\frac{\partial}{\partial \theta_1} J(\theta) \approx \frac{J(\theta_1 + \varepsilon, \theta_2, \theta_3, \dots, \theta_n) - J(\theta_1 - \varepsilon, \theta_2, \theta_3, \dots, \theta_n)}{2\varepsilon}$$

$$\frac{\partial}{\partial \theta_2} J(\theta) \approx \frac{J(\theta_1, \theta_2 + \varepsilon, \theta_3, \dots, \theta_n) - J(\theta_1, \theta_2 - \varepsilon, \theta_3, \dots, \theta_n)}{2\varepsilon}$$

⋮

$$\frac{\partial}{\partial \theta_n} J(\theta) \approx \frac{J(\theta_1, \theta_2, \theta_3, \dots, \theta_n + \varepsilon) - J(\theta_1, \theta_2, \theta_3, \dots, \theta_n - \varepsilon)}{2\varepsilon}$$



## Verificarea gradientului numeric

- Gradientul este aproximativ  $D_{ij}$ , obținut prin propagarea înapoi a erorii
- Implementare:
  - Implementarea propagării înapoi a erorii și calculul  $D$
  - Implementarea gradientului numeric
  - Verificarea că propagarea înapoi a erorii și gradientul numeric livrează aproximativ aceleași valori
  - Oprirea gradientului numeric (este încet la calcul) și antrenarea rețelei neurale utilizând propagarea înapoi a erorii



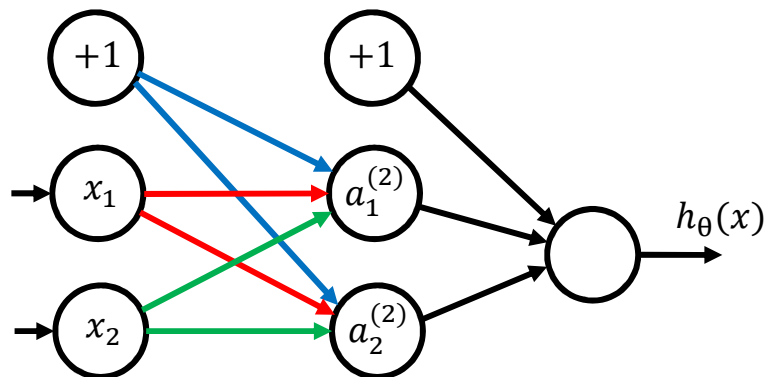
# Inițializarea ponderilor $\theta$ cu zero

- În cazul în care valorile inițiale pentru  $\theta_{ij}^{(l)}$  sunt zero, atunci fiecare neuron ascuns ( $a_1, a_2$ ) va calcula aceeași funcție pe baza valorilor de intrare (la fiecare iterație de antrenare, ponderile  $\theta_{ij}^{(l)}$  vor fi egale între ele):

$$a_1^{(2)} = a_2^{(2)}$$

$$\delta_1^{(2)} = \delta_2^{(2)}$$

$$\frac{\partial}{\partial \theta_{01}^{(1)}} J(\theta) = \frac{\partial}{\partial \theta_{02}^{(2)}} J(\theta)$$



$$\theta_{01}^{(1)} = \theta_{02}^{(1)}$$

$$\theta_{11}^{(1)} = \theta_{12}^{(1)}$$

$$\theta_{21}^{(1)} = \theta_{22}^{(1)}$$



## Inițializarea aleatoare a ponderilor $\theta$

- Inițializarea fiecărei ponderi  $\theta_{ij}^{(l)}$  cu o valoare aleatoare în intervalul  $[-\varepsilon, \varepsilon]$  (e.g.  $-\varepsilon \leq \theta_{ij}^{(l)} \leq \varepsilon$ )
- Ponderile vor fi inițializate cu valori mici, apropiate de zero

