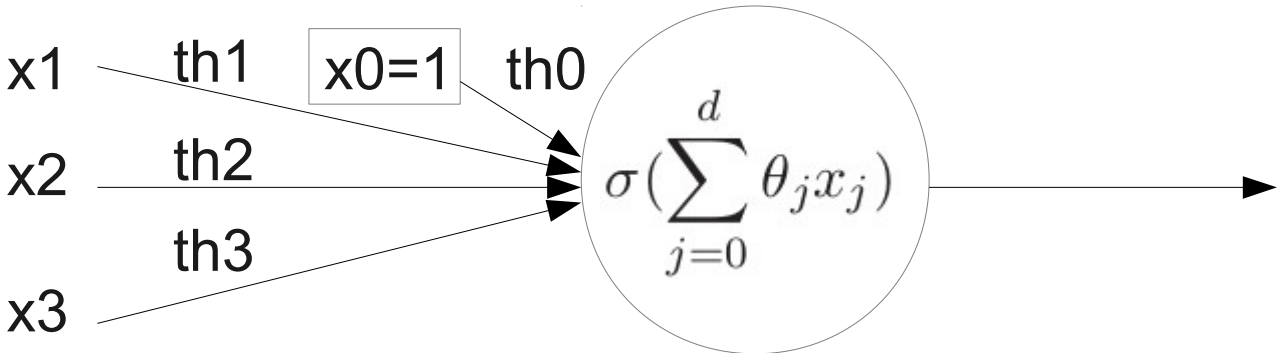
Gépi tanulás a gyakorlatban

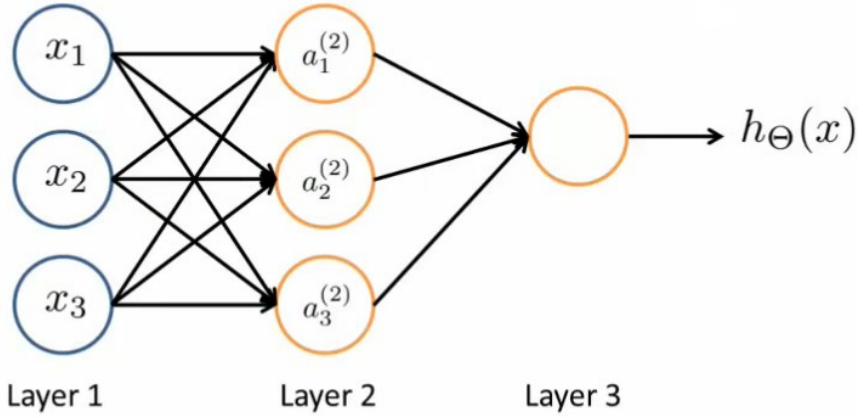
ANN

Perceptron

- Nemlineáris osztályozás megvalósítható a Logisztikus regresszió és a radial basis function (RBF) használatával
  - Előny:
    - Lineáris modellek egyszerűen kiterjeszthetőek
  - Hátrány:
    - Ekkor a „nemlinearitás” az RBF-ben van kódolva, amit a tanítás előtt meg kell adni. Jó lenne olyan módszer, ami inheritensen nemlineáris, azaz a tanulás során épít nem lineáris hipotézist (modellt)
- Perceptron intuíció: A Logisztikus regresszió (és egyéb lineáris tanulók, pl. Perceptron algoritmus) elképzelhetőek egy ideg sejt (neuron) működéséhez hasonlóan:



ANN Reprezentáció



$$\Theta = (\Theta^{(1)}, \Theta^{(2)})$$

$$\Theta^{(1)} = \begin{pmatrix} \Theta_{1,0}^{(1)} & \Theta_{1,1}^{(1)} & \Theta_{1,2}^{(1)} & \Theta_{1,3}^{(1)} \\ \Theta_{2,0}^{(1)} & \Theta_{2,1}^{(1)} & \Theta_{2,2}^{(1)} & \Theta_{2,3}^{(1)} \\ \Theta_{3,0}^{(1)} & \Theta_{3,1}^{(1)} & \Theta_{3,2}^{(1)} & \Theta_{3,3}^{(1)} \end{pmatrix}$$

$$\Theta^{(2)} = \begin{pmatrix} \Theta_{1,0}^{(2)} & \Theta_{1,1}^{(2)} & \Theta_{1,2}^{(2)} & \Theta_{1,3}^{(2)} \end{pmatrix}$$

$a_i^{(j)}$ = "activation" of unit i in layer j

$\Theta^{(j)}$ = matrix of weights controlling function mapping from layer j to layer $j+1$

$$a_1^{(2)} = g(\Theta_{1,0}^{(1)} \cdot x_0 + \Theta_{1,1}^{(1)} \cdot x_1 + \Theta_{1,2}^{(1)} \cdot x_2 + \Theta_{1,3}^{(1)} \cdot x_3)$$

$$a_2^{(2)} = g(\Theta_{2,0}^{(1)} \cdot x_0 + \Theta_{2,1}^{(1)} \cdot x_1 + \Theta_{2,2}^{(1)} \cdot x_2 + \Theta_{2,3}^{(1)} \cdot x_3)$$

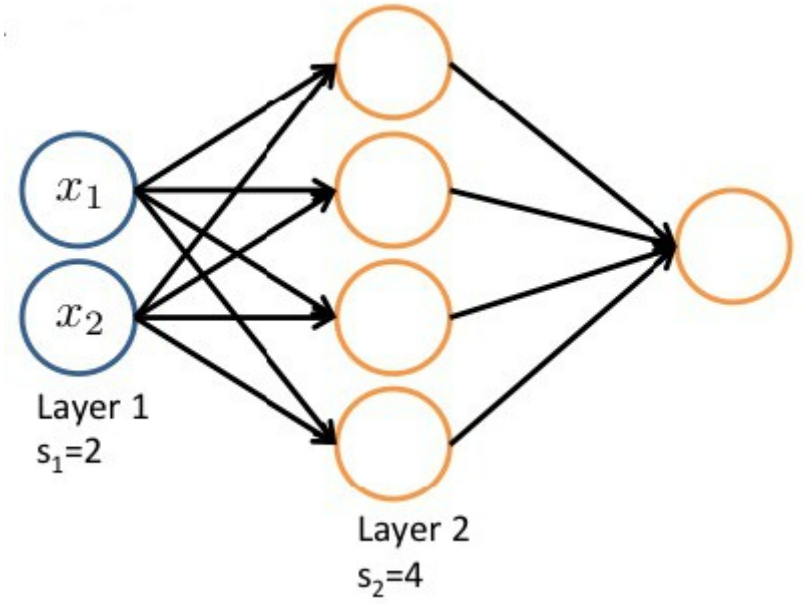
$$a_3^{(2)} = g(\Theta_{3,0}^{(1)} \cdot x_0 + \Theta_{3,1}^{(1)} \cdot x_1 + \Theta_{3,2}^{(1)} \cdot x_2 + \Theta_{3,3}^{(1)} \cdot x_3)$$

$$h_\Theta(x) = a_1^{(3)} = g(\Theta_{1,0}^{(2)} \cdot a_0^{(2)} + \Theta_{1,1}^{(2)} \cdot a_1^{(2)} + \Theta_{1,2}^{(2)} \cdot a_2^{(2)} + \Theta_{1,3}^{(2)} \cdot a_3^{(2)})$$

ANN reprezentáció

1. Kvíz:

Mekkora a dimenziója a következő ANN rejtett rétegéhez tartozó paraméter mátrixnak? Feltesszük, hogy a rejtett réteg neuronjaihoz tartozik x_0 bias bemenet, ami az ábrán nem látszik.



1. 2x4
2. 4x2
3. 3x4
4. 4x3

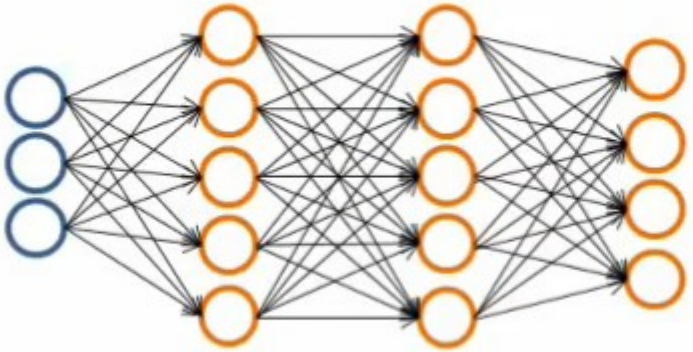
Több osztályos osztályozás – ANN


Pedestrian


Car


Motorcycle


Truck



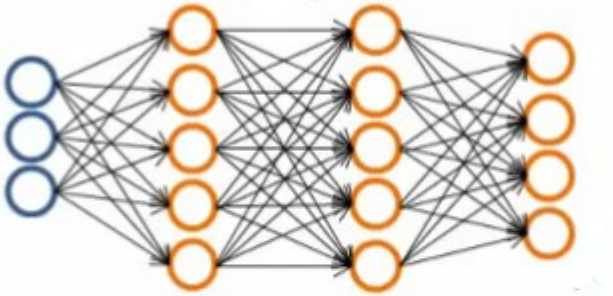
$h_\Theta(x) \in \mathbb{R}^4$

Want $h_\Theta(x) \approx \begin{bmatrix}1\\0\\0\\0\end{bmatrix}$, $h_\Theta(x) \approx \begin{bmatrix}0\\1\\0\\0\end{bmatrix}$, $h_\Theta(x) \approx \begin{bmatrix}0\\0\\1\\0\end{bmatrix}$, etc.

when pedestrian when car when motorcycle

Backpropagation

Neural Network (Classification)



$\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\}$

$L =$ total no. of layers in network

$s_l =$ no. of units (not counting bias unit) in layer l

Binary classification

$y = 0$ or 1

K=1 output unit

Multi-class classification (K classes)

$y \in \mathbb{R}^K$ E.g. $\begin{bmatrix}1\\0\\0\\0\end{bmatrix}$, $\begin{bmatrix}0\\1\\0\\0\end{bmatrix}$, $\begin{bmatrix}0\\0\\1\\0\end{bmatrix}$, $\begin{bmatrix}0\\0\\0\\1\end{bmatrix}$

pedestrian car motorcycle truck

K output units

Backpropagation – költség függvény

Logistic regression:

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

Neural network:

$$h_\Theta(x) \in \mathbb{R}^K \quad (h_\Theta(x))_i = i^{th} \text{ output}$$

$$J(\Theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log((h_\Theta(x^{(i)}))_k + (1 - y_k^{(i)}) \log(1 - (h_\Theta(x^{(i)}))_k)\right]$$

$$+\frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(\Theta_{ji}^{(l)})^2$$

$$\min_\Theta J(\Theta)$$

Need code to compute:

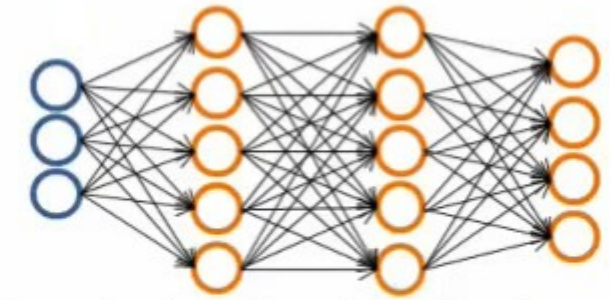
- $J(\Theta)$
- $\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta)$

ANN – Backpropagation

Given one training example (x, y):

Forward propagation:

$$a^{(1)} = x$$
$$z^{(2)} = \Theta^{(1)} a^{(1)}$$
$$a^{(2)} = g(z^{(2)}) \quad (\text{add } a_0^{(2)})$$
$$z^{(3)} = \Theta^{(2)} a^{(2)}$$
$$a^{(3)} = g(z^{(3)}) \quad (\text{add } a_0^{(3)})$$
$$z^{(4)} = \Theta^{(3)} a^{(3)}$$
$$a^{(4)} = h_\Theta(x) = g(z^{(4)})$$



Backpropagation

Intuition: $\delta_j^{(l)}$ = "error" of node j in layer l.

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

$$\delta^{(3)} = (\Theta^{(3)})^T \delta^{(4)} .* g'(z^{(3)})$$
$$\delta^{(2)} = (\Theta^{(2)})^T \delta^{(3)} .* g'(z^{(2)})$$

$$\frac{\partial}{\partial \Theta_{i,j}^{(l)}} J(\Theta) = a_j^{(l)} \delta_i^{(l+1)}$$

ANN - Backpropagation

Backpropagation algorithm

Training set $\{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$

Set $\triangle_{ij}^{(l)} = 0$ (for all l, i, j).

For $i = 1$ to m

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

Perform forward propagation to compute $a^{(l)}$ for $l = 2, 3, \ldots, L$

Using $y^{(i)}$, compute $\delta^{(L)} = a^{(L)} - y^{(i)}$

Compute $\delta^{(L-1)}, \delta^{(L-2)}, \ldots, \delta^{(2)}$

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

$D_{ij}^{(l)} := \frac{1}{m} \triangle_{ij}^{(l)} + \lambda \Theta_{ij}^{(l)}$ if $j \neq 0$

$D_{ij}^{(l)} := \frac{1}{m} \triangle_{ij}^{(l)}$ if $j = 0$