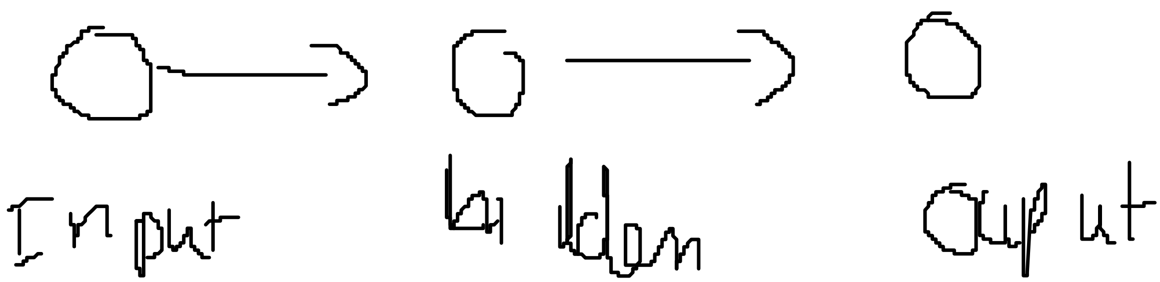
Neural and brain

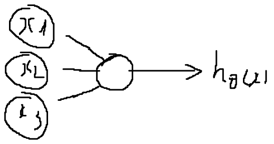
Neural network . thuật toán bắt chước bộ não



Model

Friday, November 26, 2021 10:13 AM

neuron model: logistic unit



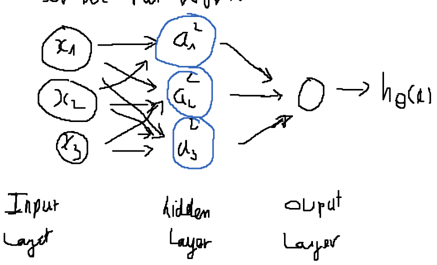
$$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}} \qquad x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \qquad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix}$$

weight (parameter)

Sigmoid (Logistic) activation function

$$a(z) = \frac{1}{1+e^{-z}}, \quad z = \theta^T x$$

Neural network



a_i^j : activation unit i of layer j

θ ma trận trọng số kiểm soát hàm ánh xạ từ một layer

$$a_1^{(2)} = g(\theta_{10}^{(1)} x_0 + \theta_{11}^{(1)} x_1 + \theta_{12}^{(1)} x_2 + \theta_{13}^{(1)} x_3)$$
$$a_2^{(2)} = g(\theta_{20}^{(1)} x_0 + \theta_{21}^{(1)} x_1 + \theta_{22}^{(1)} x_2 + \theta_{23}^{(1)} x_3)$$
$$a_3^{(2)} = g(\theta_{30}^{(1)} x_0 + \theta_{31}^{(1)} x_1 + \theta_{32}^{(1)} x_2 + \theta_{33}^{(1)} x_3)$$
$$h_\theta(x) = a_1^{(3)} = g(\theta_{10} a_0^{(2)} + \theta_{11} a_1^{(2)} + \theta_{12} a_2^{(2)} + \theta_{13} a_3^{(2)})$$

nếu mạng có S_j unit ở layer j, S_{j+1} ở layer $j+1$, thì ma trận $\theta^{(j)}$ có kích cỡ là $S_{j+1} \times (S_j + 1)$

---

vectơ hóa công thức

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

$$z^{(2)} = \theta^{(1)} x$$
$$a^2 = g(z^2)$$
$$\mathbb{R}^3 \qquad \mathbb{R}^3$$

forward propagation

R^3 R^3

thêm $a_0^2 = 1$

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

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

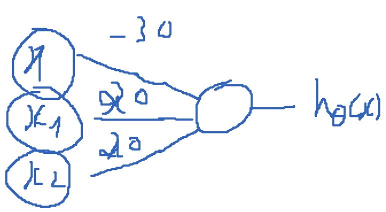
Forward Propagation

Example

x_1 and x_2

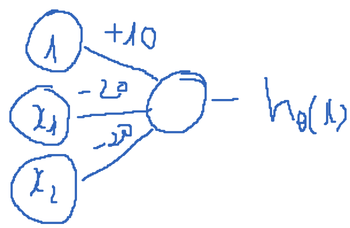
true table

x_1	x_2	x_1 and x_2	
1	1	1	$g(30)\simeq 0$
1	0	0	$g(-10)\simeq 0$
0	1	0	$g(-10)\approx 0$
0	0	0	$g(10)\approx 1$



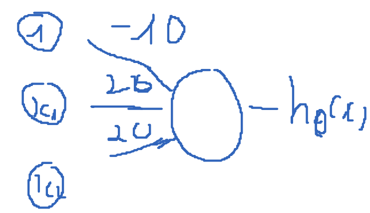
(not x_1) and (not x_2)

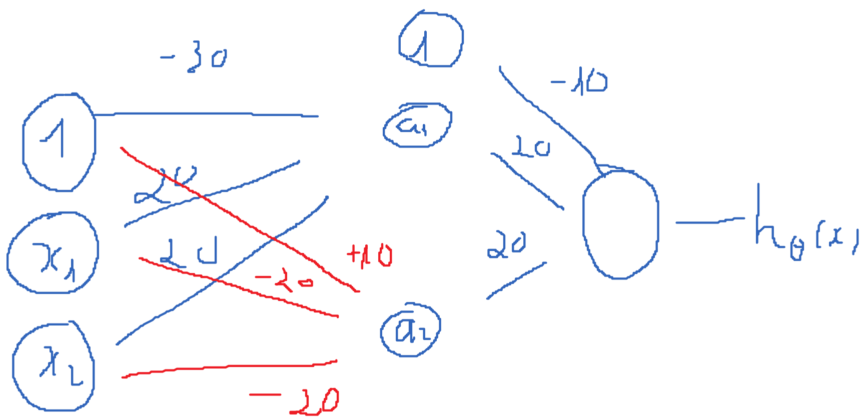
x_1	x_2	p
1	1	0
1	0	0
0	1	0
0	0	1



x_1 or x_2

x_1	x_2	p
1	1	1
1	0	1
0	1	1
0	0	0



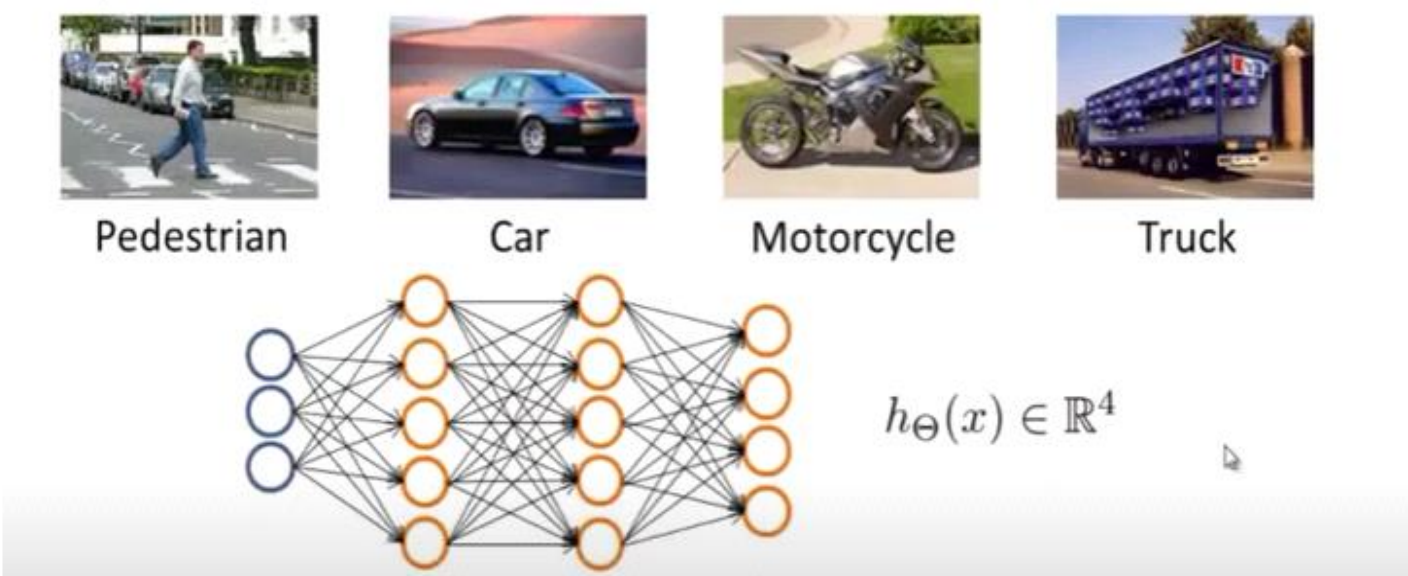


x_1 XNor x_2

x_1	x_2	p
1	1	1
1	0	0
0	1	0
0	0	1

Multiclass

multiple output one vs all



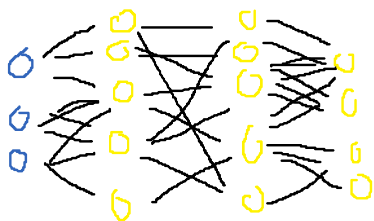
$h_\Theta(x) = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$ Pedestrian ....

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

training set $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), (x^{(3)}, y^{(3)}), \ldots, (x^{(m)}, y^{(m)})$

y one of $\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$

Cost function



$\{(x^1, y^1), (x^2, y^2), (x^3, y^3), \dots, (x^m, y^m)\}$ m example

L = no of Layer

S_l = no of unit not Include bias In layer l

binary classification

$y = 0$ or 1

1 output unit

$y \in R$

$S_L = 1$

multi-class classification K class

$y \in R^K$ $\begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \dots \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$

K output units

$S_L = K$

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

$h_\theta(x) \in R^K$

$(h_\theta(x))_i$: i th output

Backpropagation

Cost function

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

$$+ \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i}^{s_l}\sum_{j}^{s_{l+1}}\left(\theta_{ij}^{l}\right)^2$$

$\min_\theta J(\theta)$

Δ_j^{l} = error of node j in layer l

$L = 4$

$\Delta_j^4 = a_j^4 - y_j$

$\Delta_j^3 = (\theta^3)^T \Delta_j^4 \;*.\; g'(z^3)$ ~ $a^3 * (1-a^3)$

$\Delta_j^2 = (\theta^2)^T \Delta_j^3 \;*.\; g'(z^2)$ ~ $a^2 * (1-a^2)$

note: $*.$ là nhân từng phần tử 2 ma trận

VD $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} *. \begin{bmatrix} 2 & 2 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 4 \\ 3 & 4 \end{bmatrix}$

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

Algorith

training set $(x^1, y^1), (x^2, y^2), (x^3, y^3), \ldots, (x^4, y^4)$

Set $\Delta_{ij}^{l} = 0$ for all i, j, l

For $i = 1$ to m

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

Perform forward propagation to compute a^{l} for $l = 1, 2, 3, \ldots, L$

Compute $\Delta^L = a^L - y^i$ ...

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

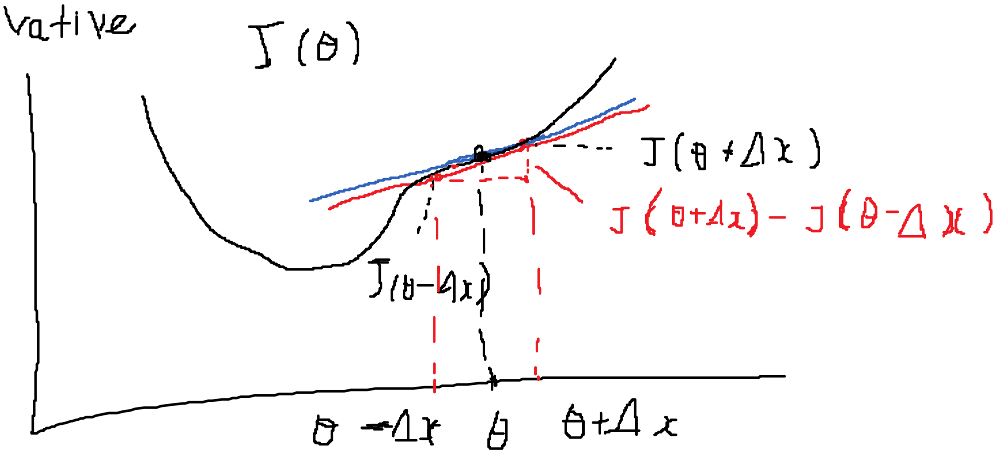
Δ^{l+1}

$\Rightarrow \quad \Delta^{l}_{ij} = \Delta^{l}_{ij} + a^{l}_{j} \Delta_{1}$

$\Delta^{l} = \Delta^{l} + \Delta^{(l+1)} \cdot a^{(l)T}$

Gradient checking

derevative



$$\frac{\partial J(\theta)}{\partial \theta} = \frac{J(\theta+\Delta x) - J(\theta-\Delta x)}{2\Delta x}$$

Vol $J(\theta)$; $\theta = [\theta_1, \theta_2, \theta_3, \theta_4, \ldots, \theta_n]$ $\qquad \Delta x \approx 10^{-4} \simeq 0$

$$\frac{\partial J(\theta)}{\partial \theta_1} = \frac{J(\theta_1+\Delta x, \theta_2, \theta_3, \ldots, \theta_n) - J(\theta_1-\Delta x, \theta_2, \ldots, \theta_n)}{2\Delta x}$$

$$\frac{\partial J\theta}{\partial \theta_2} = \frac{J(\theta_1, \theta_2+\Delta x, \theta_3, \theta_4, \ldots, \theta_n) - J(\theta_1, \theta_2-\Delta x, \ldots, \theta_n)}{2\Delta x}$$

....

$$\frac{\partial J(\theta)}{\partial \theta_n} = \frac{J(\theta_1, \theta_2, \ldots, \theta_n+\Delta x) - J(\theta_1, \theta_2, \ldots, \theta_n-\Delta x)}{2\Delta x}$$

Octave

```
for i = 1:n,
   thetaPlus = theta;
   thetaPlus(i) = thetaPlus(i) + EPSILON;
   thetaMinus = theta;
   thetaMinus(i) = thetaMinus(i) - EPSILON;
   gradApprox(i) = (J(thetaPlus) - J(thetaMinus))
                   /(2*EPSILON);
end;
```

Step

use back propagation to compute Dvec

use Gradient checking to compute Gradapprox

Make sure Dvec ≈ Gradapprox

Disable Gradient checking then run back propagation

Random initialization

If the dimensions of Theta1 is 10x11, Theta2 is 10x11 and Theta3 is 1x11.

/matrix 10x11 random(0,1)

```
Theta1 = rand(10,11) * (2 * INIT_EPSILON) - INIT_EPSILON;
Theta2 = rand(10,11) * (2 * INIT_EPSILON) - INIT_EPSILON;
Theta3 = rand(1,11) * (2 * INIT_EPSILON) - INIT_EPSILON;
```

Setup neural network

Cách để chọn cấu trúc network

No of input : Dimension of feature $x^{(i)}$

No of output : Number of classes

hidden layer : default 1, or > 1 layer, have same no of hidden unit (càng nhiều càng tốt)

Các bước training

1 random initialization các trọng số θ (weight)
2. dùng forward propagation để tính $h_\theta(x^{(i)})$ cho $y^{(i)}$
3. xây dựng code tính cost function $J(\theta)$
4. khai triển back prop để tính đạo hàm $\frac{\partial}{\partial \theta_{ij}} J(\theta)$
5 dùng Gradient checking để kiểm tra
6. dùng Gradient descent để minimize $J(\theta)$