En un capítulo anterior: función de costo

Regresión logística:

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

Red neuronal:

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

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

Aprendizaje de redes neuronales

Algoritmo de Backpropagation

Computación de gradiente

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

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

Se necesita computar:

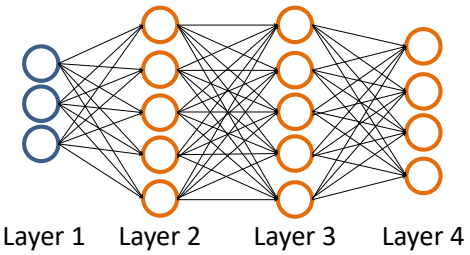
- $J(\Theta)$
- $\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta)$

Computación de gradiente

Dado un ejemplo de entrenamient o (x, y) :

Forward propagation:

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

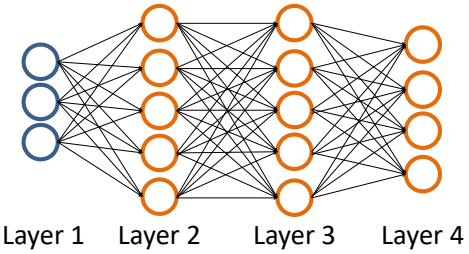


Computación de gradiente: Algoritmo de backpropagation

Intuición: $\delta_j^{(l)} =$ "error" de nodo j en capa l.

Para cada unidad de salida (layer L = 4)

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



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

Algoritmo de backpropagation

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

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

For $i = 1$ to m

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

Realizar forward propagation para computar $a^{(l)}$ para $l = 2, 3, \ldots, L$

Usando $y^{(i)}$, computar $\delta^{(L)} = a^{(L)} - y^{(i)}$

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

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

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

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

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

Aprendizaje de redes neuronales

Inicializacion aleatoria

Valor inicial de Θ

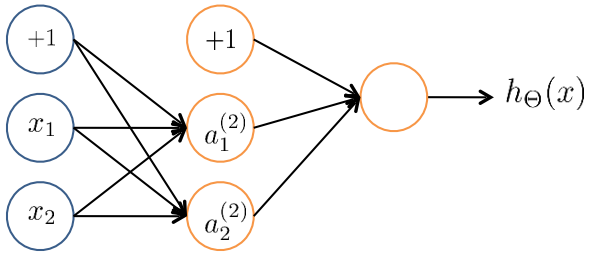
Para descenso de gradiente y metodos de optimizacion avanzada, se necesitan valores iniciales de Θ .

```
optTheta = fminunc(@costFunction,
     initialTheta, options)
```

Considerando descenso de gradiente:

Seteando `initialTheta = zeros(n,1)`?

Inicializacion Zero



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

Despues de cada actualización, los parámetros correspondientes a las entradas (yendo hacia las unidades ocultas) son.. idénticas!

Tarea: ¿por qué?

Inicializacion random : Rompimiento de simetría

Inicializar cada $\Theta_{ij}^{(l)}$ a un valor aleatorio entre $[-\epsilon, \epsilon]$

Code:

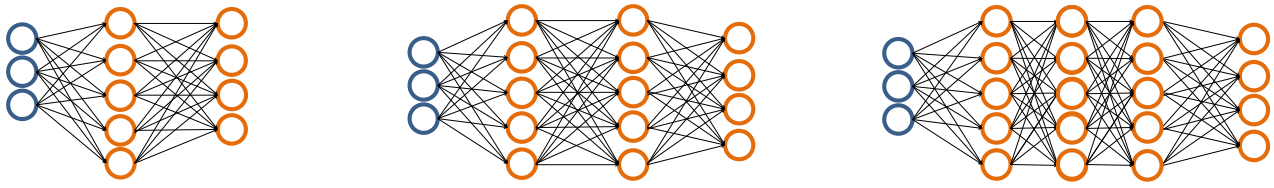
```
Theta1 =  rand(10,11)*(2*INIT_EPSILON)
          - INIT_EPSILON;

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

Aprendizaje de redes neuronales

Mezclando los ingredientes

Entrenando una red neuronal

Escoger una arquitectura de red neuronal (conectividad entre neuronas)



Nro de unidades de entrada $x^{(i)}$: Dimensiones de features
Nro de unidades de salida: Numero de clases
Razonable: 1 capa oculta, o si hay mas capas ocultas, deberian tener el mismo numero de unidades salida para cada una de ellas (usualmente a mas es mejor)

Entrenando una red neuronal (6 pasos)

1. Inicializar aleatoriamente los pesos
2. Implementar propagacion forward para hallar $h_\Theta(x^{(i)})$ para $x^{(i)}$
3. Implementar codigo para computar funcion de costo $J(\Theta)$
4. Implementar backprop para hallar derivadas parciales $\frac{\partial}{\partial \Theta_{jk}^{(l)}} J(\Theta)$

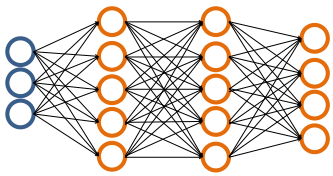
```
for i = 1:m {
```

Realizar forward propagation y backpropagation
Usando ejemplo $(x^{(i)}, y^{(i)})$
(Obtener activaciones $a^{(l)}$ y terminos delta $\delta^{(l)}$ para $l = 2, \ldots, L$).

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

}

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

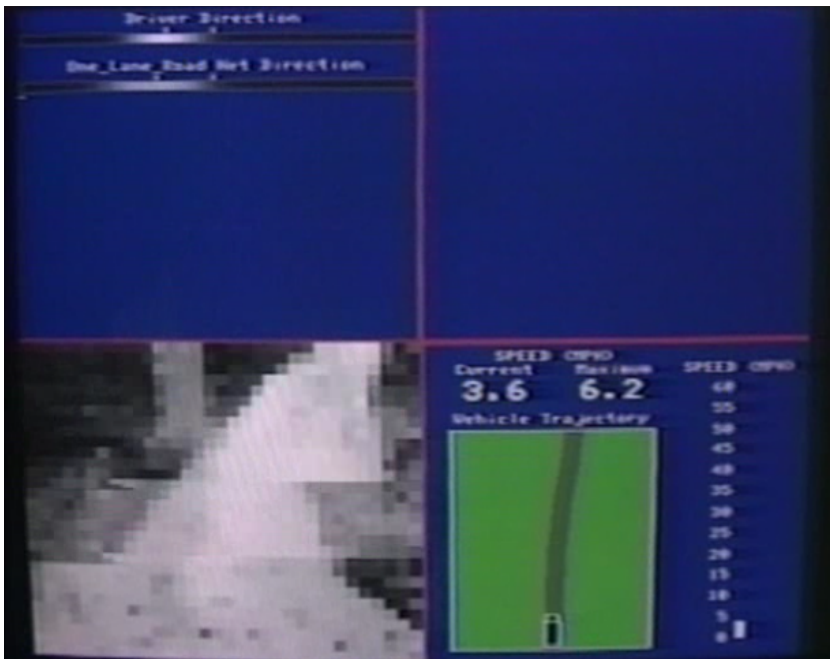


Entrenando una red neuronal

5. Usar chequeo de gradiente para comparar $\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta)$ computado usando vs. usando estimado numerico de gradiente de $J(\Theta)$. Luego desactivar chequeo de gradiente.
6. Usar descenso de gradiente o metodo avanzado de optimizacion con backpropagation para minimizar $J(\Theta)$ como funcion de parametros Θ.

Aprendizaje de redes neuronales:

Aplicación: Conducción autónoma



http://www.youtube.com/watch?v=jet4vwPUfh8

[Dean Pomerleau]