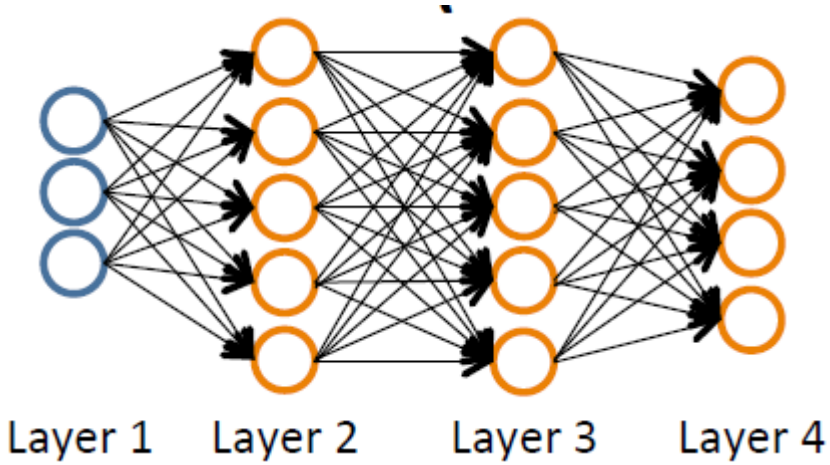
Невронни мрежи - обучение

Доц. д-р Ивайло Пенев

Кат. „Компютърни науки и технологии“

Класификация с невронна мрежа



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

L – брой слоеве в мрежата

s_l – брой върхове (без bias) в слой *l*

K – брой върхове в изходния слой

Двоична класификация
y=0 или 1
1 връх в изходния слой

Класификация в множество класове (К класа)

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

пешеходец кола мотор камион

К върха в изходния слой

Функция на цената (Cost function)

- Логаритмична регресия

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

- Невронна мрежа

$$h_\theta(x)\epsilon R^K \ (h_\theta(x))_i - \text{хипотеза в } i^{\text{ти}} \text{ изход}$$

$$J(\theta)$$

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

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

Пояснения по функцията на цената

- Във функцията на цената са добавени сумирания за множеството изходни върхове в изходния слой. В първата част на уравнението (преди квадратните скоби) имаме допълнителна сума, която итерира всеки изходен връх
- В регуляризацията (след квадратните скоби) имаме множество матрици за theta. Броят на колоните в theta матрицата за поредния връх е равен на броя на върховете в текущия слой (включвайки отклонението - bias). Броят на редовете в theta матрицата за поредния връх е равен на броя на върховете в следващия слой (изключвайки bias)
- Бележки
  - Двойната сума събира цената, изчислена чрез логаритмична регресия, за всеки връх в изходния слой
  - Тройната сума събира всички отделни θ в цялата мрежа
  - Термът *i* в третата сума не се отнася за обучителен пример, а за връх в слой *l*

Алгоритъм Backpropagation

- Backpropagation е алгоритъм за минимизиране на функцията на цената в невронна мрежа (както е градиентното спускане при линейна и логаритмична регресия)
- Функция на цената

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

- Цел - да се намери минимум на функцията на цената $min_\theta J(\theta)$
- Градиент - $\frac{\partial}{\partial\theta_{i,j}^{(j)}}J(\theta)$

Алгоритъм Backpropagation

- Дадено е множество с обучителни примери (training set) - $\left(x^{(1)}, y^{(1)}\right), \left(x^{(2)}, y^{(2)}\right), \ldots, \left(x^{(m)}, y^{(m)}\right)$
- $\Delta_{i,j}^{(l)} := 0$ за всички i, j, l
- За всеки обучителен пример t=1..m

1. $a^{(1)} := x^{(t)}$
2. Изпълняваме forward propagation за изчисляване на $a^{(l)}$ за l=2,3,…,L

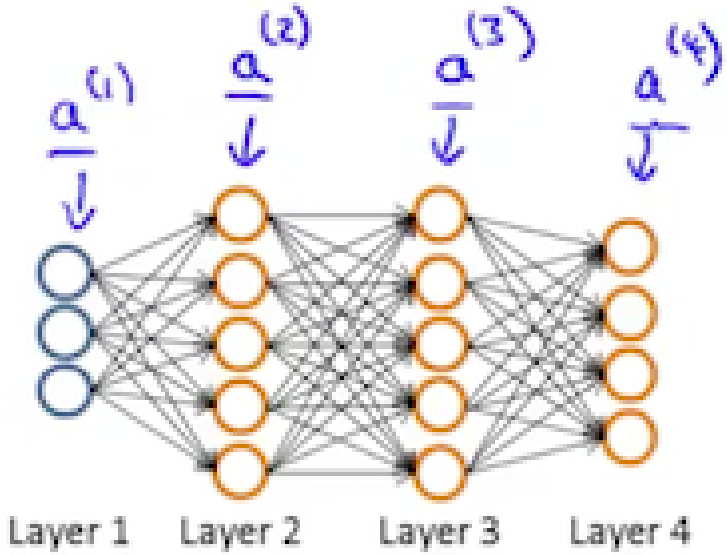
Изчисляване на градиента

Gradient computation

Given one training example (x, y):

Forward propagation:

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



Изчисляване на градиента

- Изчисляваме грешката за последния слой

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

L - общ брой слоеве

$a^{(L)}$ - вектор със стойностите на върховете в последния слой

$y^{(t)}$ - изходна (коректна) стойност (т.е. етикет) за обучителен пример t

- Изчисляваме грешките в предходните слоеве **отдясно наляво**

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

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

- Стойностите на грешката в слой l се получават чрез умножение на грешките от следващия слой с матрицата θ от слой l. Получените стойности се умножават поелементно със стойностите на функция *g'*

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

Изчисляване на градиента

- Изчисляваме $\Delta_{i,j}^{(l)}$

$$\Delta_{i,j}^{(l)} := \Delta_{i,j}^{(l)} + a_j^{(l)} \delta_i^{(l+1)}$$

- С векторизация

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

- Получаваме градиента в матрицата D

$$\frac{\partial}{\partial \theta_{ij}^{(l)}} J(\theta) = D_{i,j}^{(l)} := \frac{1}{m} \left(\Delta_{i,j}^{(l)} + \lambda \theta_{i,j}^{(l)} \right) \text{при } j \neq 0$$

$$\frac{\partial}{\partial \theta_{ij}^{(l)}} J(\theta) = D_{i,j}^{(l)} := \frac{1}{m} \Delta_{i,j}^{(l)} \text{ при } j = 0$$

Пояснения по Backpropagation