20.01.2015

What is vectorization?

$z = w^T x + b$

Non-vecto.

```
z=0
for i in range(n-x)
    z+=w[i]*x[i]
z+=b
```

$w = \begin{bmatrix} \vdots \end{bmatrix}$ $x = \begin{bmatrix} \vdots \end{bmatrix}$

Vectorized

$z = \underbrace{np.dot(w,x)}_{w^T x} + b$

→ GPU
CPU } SIMD - single instruction multiple data

---

non-vec

$u = Av$

$u_i = \sum_j A_{ij} v_j$

```
u = np.zeros((n,1))
for i ...
    for j ...
        u[i]+=A[i][j]*v[j]
```

Vector

```
u = np.dot(A,v)
```

Apply the exponential operation on every element of a matrix/vector

$v = \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix}$ $u = \begin{bmatrix} e^{v_1} \\ e^{v_2} \\ \vdots \\ e^{v_n} \end{bmatrix}$

```
u=np.zeros((n,1))
for i in range(n)
    u[i]=math.exp(v[i])
```

```
import numpy as np
u=np.exp(v)
   np.log(v)
   np.abs(v)
   np.maximum(v,0)
   v**2  → square
   1/v → inverse
```

$, , dw_1 = 0, \quad dw_2 = 0, \quad db = 0$

for $i=1$ to m:

$z^{(i)} = w^T x^{(i)} + b$

$a^{(i)} = \sigma(z^{(i)})$

$J += -[y^{(i)} \log \hat{y}^{(i)} + (1-y^{(i)}) \log(1-\hat{y}^{(i)})]$

$dz^{(i)} = a^{(i)}(1-a^{(i)})$

$dw_1 += x_1^{(i)} dz^{(i)}$ $\rightarrow$ for $j = 1 \ldots n_x$

$dw_j += dw_j$ $\Rightarrow$ $dw = np.zeros((n_x, 1))$

$dw_2 += x_2^{(i)} dz^{(i)}$

$dw += x^{(i)} dz^{(i)}$

$db += dz^{(i)}$

$J = J/m, \quad \underbrace{dw_1 = dw_1/m, \quad dw_2 = dw_2/m}_{dw /= m}, \quad db = db/m$

Vectorizing Logistic Regression

$z^{(1)} = w^T x^{(1)} + b \qquad z^{(2)} = w^T x^{(2)} + b$

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

$X = \begin{bmatrix} x^{(1)} & x^{(2)} & \ldots & x^{(m)} \end{bmatrix} \quad \frac{(n_x, m)}{\mathbb{R}^{n_x \times m}} \qquad w^T \begin{bmatrix} x^{(1)} & x^{(2)} & \ldots & x^{(m)} \end{bmatrix}$

$Z = [z^{(1)} \; z^{(2)} \ldots z^{(m)}] = w^T X + \underbrace{[b \ldots b]}_{1 \times m} = w^T x^{(1)} + b \quad w^T x^{(2)} + b \ldots$

$Z = np.dot(w.T, x) + b$ $\rightarrow$ real number but python convert to (1,1) matrix "Broadcasting"

$A = [a^{(1)}, a^{(2)} \ldots a^{(m)}]$

Vectorizing Logistic Regression

$dz^{(1)} = a^{(1)} - y^{(1)}$ $\quad dz^{(1)} = a^{(2)} - y^{(2)} \ldots$

$dz = [dz^{(1)} \; dz^{(2)} \ldots dz^{(m)}]$

$A = [a^{(1)} \ldots a^{(m)}]$ $\quad Y = [y^{(1)} \ldots y^{(m)}]$

$dz = A - Y = [a^{(1)} - y^{(1)} \quad a^{(2)} - y^{(2)} \ldots]$

$dw = 0$ | $db = 0$

$dw += x^{(1)} dz^{(1)}$ | $db += dz^{(1)}$

$dw += x^{(2)} dz^{(2)}$ | $db += dz^{(2)}$

$\vdots$ | $\vdots$

$dw /= m$ | $db /= m$

Vectorizing

$db = \frac{1}{m} \sum_{i=1}^{m} dz^{(i)}$

$= \frac{1}{m}$ np.sum(dz)

$dw = \frac{1}{m} X dz^T$

$= \frac{1}{m} [x^{(1)} \ldots x^{(m)}] \begin{bmatrix} dz^{(1)} \\ \vdots \\ dz^{(m)} \end{bmatrix}$

$= \frac{1}{m} [x^{(1)} dz^{(1)} + \ldots + x^{(m)} dz^{(m)}]$

Implementing Logistic Regression

$J = 0, \; dw_1 = 0, \; dw_2 = 0, \; db = 0$

For $i = 1$ to m:

$z^{(i)} = w^T x^{(i)} + d$

$a^{(i)} = \sigma(z^{(i)})$

$J += -[y^{(i)} \log a^{(i)} + (1 - y^{(i)}) \log(1 - a^{(i)})]$

$dz^{(i)} = a^{(i)} - y^{(i)}$

$\left.\begin{array}{l} dw_1 += x_1^{(i)} dz^{(i)} \\ dw_2 += x_2^{(i)} dz^{(i)} \end{array}\right] \; dw += x^{(i)} * dz^{(i)}$

$db += dz^{(i)}$

$J = J/m, \; dw_1 = dw_1/m, \; dw_2 = dw_2/m$

$db = db/m$

for iter in range(1000):

$Z = w^T X + b$

= np.dot(w.T, X) + b

$A = \sigma(Z)$

$dZ = A - Y$

$dw = \frac{1}{m} X dZ^T$

$db = \frac{1}{m}$ np.sum(dZ)

$w := w - \alpha dw$

$b := b - \alpha db$

tek bir gradient dec. gerçekleşir

costing example

$$
\begin{array}{c|cccc}
 & \text{Apples} & \text{Beef} & \text{Eggs} & \text{Potatoes} \\
\hline
\text{Carb} & 56.0 & 0.0 & 4.4 & 68.0 \\
\text{Protein} & 1.2 & 104.0 & 52.0 & 8.0 \\
\text{Fat} & 1.8 & 135.0 & 99.0 & 0.9
\end{array}
= A_{(3,4)}
$$

59cal

import numpy as np

A = np.array([[56.0, 0.0, 4.4, 68.0],
[1.2, 104.0, 52.0, 8.0],
[1.8, 135.0, 99.0, 0.9]])

col = A.sum(axis=0) → vertically "0"

print(col)

→ [59. 239. 155.4 29.9]

percentage = 100*A/col.reshape(1,4)

technically don't need

import numpy as np

a=np.random.randn(5) → [- - - -]

Saye veri matris deh

(5,1) → olmalı

a= np.random.randn(5)

Don't use it { a.shape =(5,)
"rank 1 array" → raw veya column vector değil

a = np.random.randn(5,1) → column vector

(1,5) → raw vector

assert(a.shape==(5,1))

col = A.sum(axis=0) — vertically

axis=1 → horizontally

$$\begin{bmatrix}1\\2\\3\\4\end{bmatrix} + \begin{bmatrix}100\\100\\100\\100\end{bmatrix} 100 = \begin{bmatrix}101\\102\\103\\104\end{bmatrix}$$

$$\underbrace{\begin{bmatrix}1&2&3\\4&5&6\end{bmatrix}}_{(m,n)} + \begin{bmatrix}100 & 200 & 300\end{bmatrix} \begin{matrix}100 & 200 & 300\end{matrix} = \begin{bmatrix}101 & 202 & 303\\104 & 205 & 306\end{bmatrix}$$

(1,n) ~ (m,n)

$$\underbrace{\begin{bmatrix}1&2&3\\4&5&6\end{bmatrix}}_{(m,n)} + \underbrace{\begin{bmatrix}100\\200\end{bmatrix}}_{(m,1)} \begin{matrix}100 & 100\\200 & 200\end{matrix} \Rightarrow \begin{bmatrix}101 & 102 & 103\\204 & 205 & 206\end{bmatrix}$$

(m,n)

General Principle

$(m,n) \;\; \begin{matrix}+\\-\\*\\/\end{matrix} \;\; \begin{matrix}(1,n) \rightsquigarrow (m,n)\\ (m,1) \rightsquigarrow (m,n)\end{matrix}$

(m,1) + R

$$\begin{bmatrix}1\\2\\3\end{bmatrix} + 100 = \begin{bmatrix}101\\102\\103\end{bmatrix}$$

$[1\ 2\ 3] + 100 = [101\ 102\ 103]$

(2,3) $(2,1)^T$ (1,3) → (2,3)

element-wise $^T(3,1)$ (1,3) (3,2)

(4,3) (2,1) → hata

dot (4,3)(3,4) → (4,4)

Logistic regression cost function

$\hat{y} = \sigma(w^T x + b)$ where $\sigma(z) = \frac{1}{1+e^{-z}}$

interpret $\hat{y} = p(y=1|x)$

$$\left.\begin{array}{l} \text{if } y=1 : p(y|x) = \hat{y} \\ \text{if } y=0 : p(y|x) = 1-\hat{y} \end{array}\right\} p(y|x)$$

$p(y|x) = \hat{y}^y (1-\hat{y})^{(1-y)}$

If $y=1$: $\hat{y}$

If $y=0$: $(1-\hat{y})$

maximum

$\uparrow \log p(y/x) = \log \hat{y}^y (1-\hat{y})^{(1-y)} = y \log \hat{y} + (1-y) \log(1-\hat{y})$

$= -\mathcal{L}(\hat{y}, y) \downarrow$ minimize

Cost on m examples

$\log p(\text{labels in training set}) = \log \prod_{i=1}^{m} p(y^{(i)}|x^{(i)})$

$\log p(\ldots) = \sum_{i=1}^{m} \underbrace{\log p(y^{(i)}/x^{(i)})}_{-\mathcal{L}(\hat{y}^{(i)}, y^{(i)})}$ maximum likelihood estimation

$= -\sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)})$

Cost : $J(w,b) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)})$

minimize