[2-2. Python and Vectorization]

<Vectorization>

* 벡터화: 코딩에서 for문을 제거하는 기술

ex) $z = w^T x + b$

(Non-vectorization)

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

(Vectorization)

```
z = np.dot(w, x) +b
```

* SIMD (Single Instruction Multiple Data) : 병렬 프로세스의 한 종류로, 하나의 명령어로 여러개의 값을 동시에 처리하는 방식으로 벡터 연산을 가능하게 함

ex2) $v = \begin{bmatrix} v_1 \\ \cdots \\ v_n \end{bmatrix}$ $u = \begin{bmatrix} e^{v_1} \\ \cdots \\ e^{v_n} \end{bmatrix}$

(non-vectorization)

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

(vectorization)

```
u= np.exp(v)
```

<Vectorizing Logistic Regression>

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

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

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

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

$z^{(3)} = w^T x^{(3)} + b$

$a^{(3)} = \sigma(z^{(3)})$

...

$\Rightarrow X = \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix}$

$[z^{(1)} \ z^{(2)} \cdots z^{(m)}] = w^T X + [b \ b \cdots b]$

$= [\underbrace{w^T x^{(1)} + b}_{= z^{(1)}} \quad \underbrace{w^T x^{(2)} + b}_{= z^{(2)}} \cdots \underbrace{w^T x^{(m)} + b}_{= z^{(m)}}]$

(vectorization coding)

```
z= np.dot(np.transpose(w), x) + b
```

< Vectorizing Logistic Regression's Gradient Descent >

i) dz

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

$\Rightarrow dz = [dz^{(1)} \quad dz^{(2)} \quad \cdots \quad dz^{(m)}]$

$A = [a^{(1)} \quad a^{(2)} \quad \cdots \quad a^{(m)}]$

$Y = [y^{(1)} \quad y^{(2)} \quad \cdots \quad y^{(m)}]$

$\Rightarrow dz = A - Y$

$= [a^{(1)} - y^{(1)} \quad a^{(2)} - y^{(2)} \quad \cdots \quad a^{(m)} - y^{(m)}]$

ii) dw

$dw = 0$

$dw \mathrel{+}= x^{(1)} dz^{(1)}$

$dw \mathrel{+}= x^{(2)} dz^{(2)}$

$\cdots$

$dw \mathrel{+}= x^{(m)} dz^{(m)}$

$dw \mathrel{/}= m$

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

$$= \frac{1}{m} \begin{bmatrix} | & & | \\ x^{(1)} & \cdots & x^{(m)} \\ | & & | \end{bmatrix} \begin{bmatrix} dz^{(1)} \\ \vdots \\ dz^{(m)} \end{bmatrix}$$

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

iii) db

$db = 0$

$db \mathrel{+}= dz^{(1)}$

$db \mathrel{+}= dz^{(2)}$

$\cdots$

$db \mathrel{+}= dz^{(m)}$

$db \mathrel{/}= m$

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

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

$$= np.sum(dz)$$

non - vectorizing

```
J = 0, dw_1 = 0, dw_2 = 0, db = 0
for i = 1 to m:
```

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

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

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

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

$$\left.\begin{aligned} dw_1 &\mathrel{+}= x_1^{(i)} dz^{(i)} \\ dw_2 &\mathrel{+}= x_2^{(i)} dz^{(i)} \end{aligned}\right\} \Rightarrow dw \mathrel{+}= x^{(i)} * dz^{(i)}$$

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

```
J = J/m, dw_1 = dw_1/m, dw_2 = dw_2/m
db = db/m
```

Vectorizing

$\rightarrow Z = w^T X + b = \text{np.dot}(w^T, X) + b$

$A = \sigma(Z)$

$dZ = A - Y$

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

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

$w = w - \alpha dw$

$b = b - \alpha d$

<Broadcasting in Python>

ex1)

$$A = \begin{bmatrix} 56.0 & 0.0 & 4.4 & 68.0 \\ 1.2 & 104.0 & 52.0 & 8.0 \\ 1.8 & 135.0 & 99.0 & 0.9 \end{bmatrix} \rightarrow$$

```
cal = A.sum(axis=0)
percentage = 100*A/cal.reshape(1,4)
```

$\uparrow$ (3,4) $\uparrow$ (1,4)

⇒ (3,4) 행렬을 (1,4)로 나눔

ex2)

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

ex3)

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

$(1,n) \xrightarrow{\text{확장}} (m,n)$

$$\begin{bmatrix} 100 & 200 & 300 \\ 100 & 200 & 300 \end{bmatrix}$$

ex4)

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

$(m,1) \xrightarrow{\text{확장}} (m,n)$

$$\begin{bmatrix} 100 & 100 & 100 \\ 200 & 200 & 200 \end{bmatrix}$$

$\therefore$ (m, n) matrix $\begin{pmatrix} + \\ - \\ * \\ \div \end{pmatrix}$ $(1,n) \rightsquigarrow (m,n)$ / $(m,1) \rightsquigarrow (m,n)$ 로 확장하여 계산

<Explanation of Logistic Regression Cost Function>

Logistic regression cost function

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

$$\rightarrow \hat{y} = P(y=1 \mid x)$$

if $y=1$: $P(y|x) = \hat{y}$

if $y=0$: $P(y|x) = 1-\hat{y}$

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

if $y=1$: $P(y|x) = \hat{y}^{1}(1-\hat{y})^{0} = \hat{y}$

if $y=0$: $P(y|x) = \hat{y}^{0}(1-\hat{y})^{1} = 1-\hat{y}$

로그함수는 단조증가 함수이므로,

$$\underbrace{\log P(y|x)}_{\text{최대화}} = \log(\hat{y}^{y}(1-\hat{y})^{(1-y)}) = \underbrace{y\log\hat{y} + (1-y)\log(1-\hat{y})}_{= -L(y,\hat{y})}$$

$= -L(y,\hat{y})$: $-$손실함수 (최소화)

$\therefore$ 손실함수 $L(\hat{y}, y) = -\log P(y|x) = -(\hat{y}^{y}(1-\hat{y})^{(1-y)})$

m개의 훈련세트로 확장시키면,

$$P(\text{labels of training set}) = \prod_{i=1}^{m} P(y^{(i)} \mid x^{(i)})$$

$\xrightarrow{\text{로그}} \log P(\text{labes of training set}) = \log \prod_{i=1}^{m} P(y^{(i)} \mid x^{(i)}) = -\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)})$

$\therefore$ 비용함수 $J(w,b) = -\log P(\text{labels of traing set})$

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

Made with Goodnotes