Week 2b - Python and vectorization

Saturday, August 1, 2020 12:14 PM

① Vectorization

→ $z = w^T x + b$

$w = \begin{bmatrix} \vdots \\ \vdots \end{bmatrix}$ $x = \begin{bmatrix} \vdots \\ \vdots \end{bmatrix}$ $w \in \mathbb{R}^{n_x}$ $x \in \mathbb{R}^{n_x}$

→ Non-vectorized

```
z = 0
For i in range(n_x):
    z += w[i] * x[i]
z += b
```

→ Vectorized

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

($w^T x$)

→ Neural Network programming guideline

- Whenever possible, avoid explicit for loops

$u = Av$

$u_i = \sum_j A_{ij} v_j$

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

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

- vector and matrix valued functions

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

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

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

→ Algorithm

→ Initialise $J = 0$; ~~$dw_1 = 0$; $dw_2 = 0$~~; $db = 0$

$dw = np.zeros((n_x, 1))$

accumulated $dw_1^{(i)}$ & $dw_2^{(i)}$

→ for i=1 to m

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

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

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

for j=1 to n_x

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

$dw_1 += x_1^{(i)} dz^{(i)}$

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

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

↑ assuming $n = 2$ for this eg. else has to be looped n times

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

→ J/m ;
[redacted] dw/=m
db/m ;

→ $w_1 := w_1 - \alpha dw_1$
$w_2 := w_2 - \alpha dw_2$
$b := b - \alpha db$

② Vectorizing Logistic Regression

Vectorizing Logistic Regression

$z^{(1)} = w^T x^{(1)} + b$ $\quad z^{(2)} = w^T x^{(2)} + b$ $\quad z^{(3)} = w^T x^{(3)} + b$

$a^{(1)} = \sigma(z^{(1)})$ $\quad a^{(2)} = \sigma(z^{(2)})$ $\quad a^{(3)} = \sigma(z^{(3)})$

$X = \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix}$ $\quad (n_x, m)$ $\quad \mathbb{R}^{n_x \times m}$

$w^T \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix}$

$Z = \begin{bmatrix} z^{(1)} & z^{(2)} & \cdots & z^{(m)} \end{bmatrix} = w^T X + \underbrace{[b \; b \cdots b]}_{1 \times m} = \begin{bmatrix} w^T x^{(1)} + b & w^T x^{(2)} + b & \cdots & w^T x^{(m)} + b \end{bmatrix}$

→ `Z = np.dot(w.T, X) + b` $\quad (1,1)$ $\quad \mathbb{R}$ $\quad$ "Broadcasting"

$A = \begin{bmatrix} a^{(1)} & a^{(2)} & \cdots & a^{(m)} \end{bmatrix} = \sigma(Z)$

Andrew Ng

③ Vectorizing Logistic Regression's Gradient Output

Vectorizing Logistic Regression

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

$dZ = \underbrace{[dz^{(1)} \; dz^{(2)} \cdots dz^{(m)}]}_{1 \times m}$ ←

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

→ $dZ = A - Y = [a^{(1)} - y^{(1)} \quad a^{(2)} - y^{(2)} \quad \cdots]$

→ $dw = 0$
$dw \mathrel{+}= x^{(1)} dz^{(1)}$
$dw \mathrel{+}= x^{(2)} dz^{(2)}$
$\vdots$
$dw /= m$

$db = 0$
$db \mathrel{+}= dz^{(1)}$
$db \mathrel{+}= dz^{(2)}$
$\vdots$
$db \mathrel{+}= dz^{(m)}$
$db /= m.$

$db = \frac{1}{m} \sum_{i=1}^{m} dz^{(i)}$
$= \frac{1}{m}$ `np.sum(dZ)`

$dw = \frac{1}{m} X dz^T$
$= \frac{1}{m} \begin{bmatrix} | & & | \\ x^{(1)} & \cdots & x^{(m)} \\ | & & | \end{bmatrix} \begin{bmatrix} dz^{(1)} \\ \vdots \\ dz^{(m)} \end{bmatrix}$
$= \frac{1}{m} \left[x^{(1)} dz^{(1)} + \cdots + x^{(m)} dz^{(m)} \right]$
$n_x \times 1$

Andrew Ng

Implementing Logistic Regression

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

for i = 1 to m:

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

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

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

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

$dw_1\ += x_1^{(i)} dz^{(i)}$

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

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

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

$db = db/m$

for iter in range(1000):

$Z = w^T X + b$

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

$A = \sigma(Z)$

$dZ = A - Y$

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

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

$w := w - \alpha dw$

$b := b - \alpha db$

Andrew Ng

④ Broadcasting in Python

General Principle

(m, n) matrix $\pm$ $*$ $/$ $(1, n)$ $\rightsquigarrow$ (m, n)

(m, n) matrix $\pm$ $*$ $/$ $(m, 1)$ $\rightsquigarrow$ (m, n)

$(m, 1) + \mathbb{R}$

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

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

Matlab/Octave: bsxfun

⑤ Note about Python/NumPy

Python/numpy vectors

```
a = np.random.randn(5)
```
a.shape = (5,)
"rank 1 array"

} Don't use

```
a = np.random.randn(5,1)   → a.shape = (5,1)   column vector ✓
a = np.random.randn(1,5)   → a.shape = (1,5)   row vector ✓
assert(a.shape == (5,1))   ← → to double-check
a = a.reshape((5,1))
```

Andrew Ng

⑥ Explanation of logistic regression cost fn

Logistic regression cost function

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

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

} $p(y|x)$

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

If $y=1$: $p(y|x) = \hat{y}^{1} (1-\hat{y})^{0} = \hat{y} \times 1 = \hat{y}$

If $y=0$: $p(y|x) = \hat{y}^{0} (1-\hat{y})^{(1-0)} = 1 \times (1-\hat{y}) = 1-\hat{y}$

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

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

Andrew Ng