

Week 2-2 Vectorization

笔记本: DL 1 - NN and DL

创建时间: 2021/1/5 11:15

更新时间: 2021/1/7 01:04

[1] ▶ ML

```
import numpy as np

a = np.array([1, 2, 3, 4])
print(a)
```

[1 2 3 4]

[20] ▶ ML

```
import time

a = np.random.rand(1000000)
b = np.random.rand(1000000)

tic = time.time()
c = np.dot(a,b)
toc = time.time()
print("Vectorization Version: " + str(1000*(toc-tic)) + " ms")

c = 0
tic = time.time()
for i in range(1000000):
    c += a[i]*b[i]
toc = time.time()
print("For Loop Version: " + str(1000*(toc-tic)) + " ms")
```

Vectorization Version: 0.9870529174804688 ms

For Loop Version: 526.5719890594482 ms

[21] ▶ ML

```
tic = time.time()
d = np.exp(a)
toc = time.time()
print("Vectorization Version: " + str(1000*(toc-tic)) + " ms")

import math
c = 0
tic = time.time()
for i in range(1000000):
    d[i] = math.exp(a[i])
toc = time.time()
print("For Loop Version: " + str(1000*(toc-tic)) + " ms")
```

Vectorization Version: 58.83979797363281 ms

For Loop Version: 351.0608673095703 ms

Vectorizing Logistic Regression

Vectorizing Logistic Regression

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

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

$$\underline{X} = \begin{bmatrix} x^{(1)} & x^{(2)} & \dots & x^{(m)} \\ \uparrow & \uparrow & & \uparrow \\ \mathbb{R}^{n_x \times m} & & & \end{bmatrix} \quad \begin{matrix} (n_x, m) \\ \mathbb{R}^{n_x \times m} \end{matrix} \quad \begin{matrix} \xrightarrow{\quad} \\ \omega^T \end{matrix} \begin{bmatrix} x^{(1)} & x^{(2)} & \dots & x^{(m)} \\ \vdots & \vdots & & \vdots \end{bmatrix}$$

$$\underline{z} = \begin{bmatrix} z^{(1)} & z^{(2)} & \dots & z^{(m)} \end{bmatrix} = \omega^T \underline{X} + \begin{bmatrix} b & b & \dots & b \end{bmatrix}_{1 \times m} = \begin{bmatrix} \omega^T x^{(1)} + b & \omega^T x^{(2)} + b & \dots & \omega^T x^{(m)} + b \end{bmatrix}_{1 \times m}$$

$$\rightarrow \underline{z} = \text{np.dot}(\omega.T, X) + \frac{b}{\mathbb{1}} \quad (1,1) \quad \mathbb{R} \quad \text{"Broadcasting"}$$

$$\underline{A} = \begin{bmatrix} a^{(1)} & a^{(2)} & \dots & a^{(m)} \end{bmatrix} = \sigma(\underline{z})$$

X here $n_x * m$ (compared with one in ML)

Vectorizing Logistic Regression

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

$$\underline{dz} = \begin{bmatrix} dz^{(1)} & dz^{(2)} & \dots & dz^{(m)} \end{bmatrix}_{1 \times m} \quad \leftarrow$$

$$A = \begin{bmatrix} a^{(1)} & \dots & a^{(m)} \end{bmatrix} \quad Y = \begin{bmatrix} y^{(1)} & \dots & y^{(m)} \end{bmatrix}$$

$$\rightarrow \underline{dz} = A - Y = \begin{bmatrix} a^{(1)} - y^{(1)} & a^{(2)} - y^{(2)} & \dots \end{bmatrix}$$

$$\begin{cases} \rightarrow \underline{dw} = 0 \\ \underline{dw} += \underline{x}^{(1)} dz^{(1)} \\ \underline{dw} += \underline{x}^{(2)} dz^{(2)} \\ \vdots \\ \underline{dw} /= m \end{cases} \quad \begin{cases} \underline{db} = 0 \\ \underline{db} += dz^{(1)} \\ \underline{db} += dz^{(2)} \\ \vdots \\ \underline{db} += dz^{(m)} \\ \underline{db} /= m \end{cases}$$

$$\begin{cases} \underline{db} = \frac{1}{m} \sum_{i=1}^m dz^{(i)} \\ = \frac{1}{m} \text{np.sum}(\underline{dz}) \\ \underline{dw} = \frac{1}{m} X \underline{dz}^T \\ = \frac{1}{m} \begin{bmatrix} x^{(1)} & \dots & x^{(m)} \\ \vdots & & \vdots \\ 1 & & 1 \end{bmatrix} \begin{bmatrix} dz^{(1)} \\ \vdots \\ dz^{(m)} \end{bmatrix} \\ = \frac{1}{m} \begin{bmatrix} x^{(1)} dz^{(1)} + \dots + x^{(m)} dz^{(m)} \end{bmatrix}_{n \times 1} \end{cases}$$

Implementing Logistic Regression

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

for $i = 1$ to m :

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

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

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

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

$$\begin{cases} dw_1 += x_1^{(i)} dz^{(i)} \\ dw_2 += x_2^{(i)} dz^{(i)} \\ db += dz^{(i)} \end{cases} \quad dw += x^{(i)} * dz^{(i)}$$

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

$db = db/m$

$$z = w^T X + b$$

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

$$A = \sigma(z)$$

$$dz = A - Y$$

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

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

$$w := w - \alpha dw$$

$$b := b - \alpha db$$

Broadcasting

[22] ▶ M4

Broadcasting

```
A = np.array([[56.0, 6.0, 4.4, 68.0],
               [1.2, 104.0, 52.0, 8.0],
               [1.8, 135.0, 99.0, 0.9]])
```

```
s = np.sum(A, axis=0)
percentage = 100*A/s
print(percentage)
```

```
[[94.91525424  2.44897959  2.83140283 88.42652796]
 [ 2.03389831 42.44897959 33.46203346 10.40312094]
 [ 3.05084746 55.10204082 63.70656371  1.17035111]]
```

```
op1 = np.array([i for i in range(9)]).reshape(3, 3)
op2 = np.array([[1, 2, 3]])
op3 = np.array([1, 2, 3])
```

```
pp.pprint(op1)
pp.pprint(op2)
```

```
# Notice that the result here is DIFFERENT!
print(op2.shape)
pp.pprint(op1 + op2)
pp.pprint(op1 + op2.T)
```

```
# Notice that the result here are THE SAME! - Always use (3,1) vector rather
than (3,) vector
print(op3.shape)
pp.pprint(op1 + op3)
pp.pprint(op1 + op3.T)
```

