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]

```python
import numpy as np

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

```
[1 2 3 4]
```

[20]

```python
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")
```

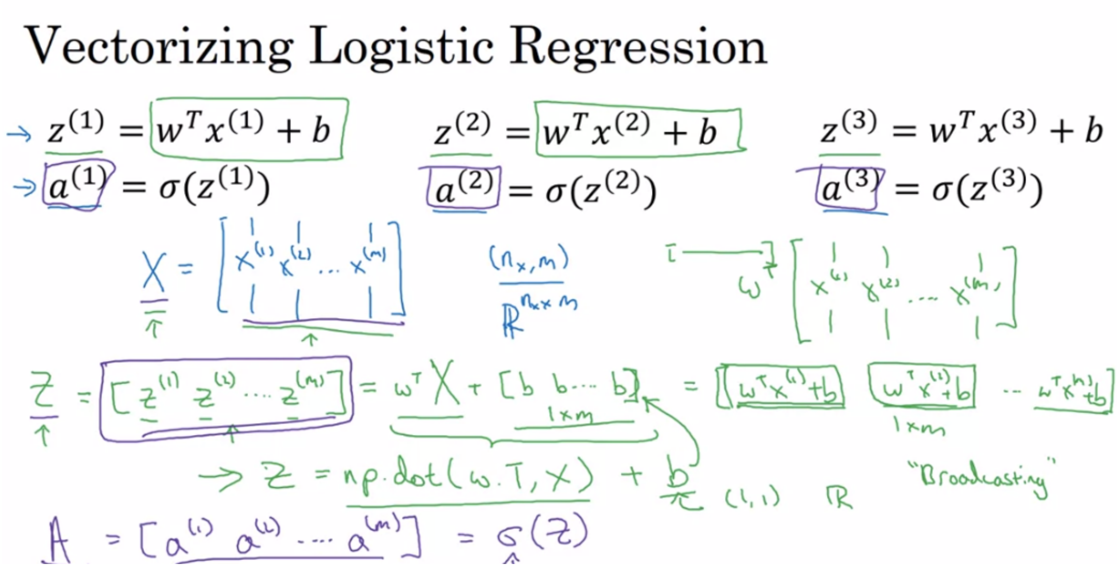
```
Vectorization Version: 0.9870529174804688 ms
For Loop Version: 526.5719890594482 ms
```

[21]

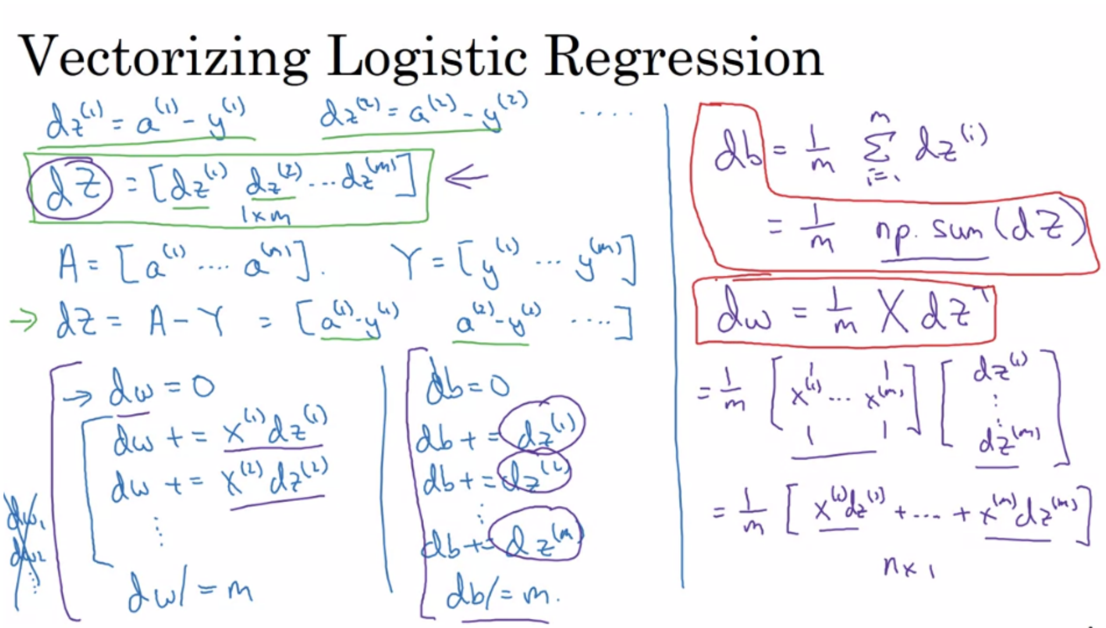
```python
tic = time.time()
d = np.exp(a)
toc = time.time()
print("Vectorization Version: " + str(1000*(toc-tic)) + " ms")

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



X here nx * m (compared with one in ML)



Implementing Logistic Regression

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

$\boxed{\text{for } i = 1 \text{ to } m:}$

$$z^{(i)} = w^T x^{(i)} + b$$
$$a^{(i)} = \sigma(z^{(i)})$$
$$J \mathrel{+}= -[y^{(i)} \log a^{(i)} + (1 - y^{(i)}) \log(1 - a^{(i)})]$$
$$dz^{(i)} = a^{(i)} - y^{(i)}$$
$$\left.\begin{array}{l} dw_1 \mathrel{+}= x_1^{(i)} dz^{(i)} \\ dw_2 \mathrel{+}= x_2^{(i)} dz^{(i)} \end{array}\right\} dw \mathrel{+}= x^{(i)} * dz^{(i)}$$
$$db \mathrel{+}= dz^{(i)}$$

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

$db = db/m$

$$Z = w^T X + b$$
$$= np.dot(w.T, X) + b$$
$$A = \sigma(Z)$$
$$dZ = A - Y$$
$$dw = \frac{1}{m} X dZ^T$$
$$db = \frac{1}{m} np.sum(dZ)$$
$$w := w - \alpha dw$$
$$b := b - \alpha db$$

Broadcasting

[22]

```python
# 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]]
```

```python
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)
```