Vectorization

- We have to calculate $z = w^T x + b$

$$w = \begin{bmatrix} \vdots \\ \vdots \end{bmatrix}, w \in \mathbb{R}^{\mathrm{n_x}} \tag{1}$$
$$x = \begin{bmatrix} \vdots \\ \vdots \end{bmatrix}, x \in \mathbb{R}^{\mathrm{n_x}}$$

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

Vectors and matrix valued functions

- avoid for loop as possible by using np operation instead
- `u=np.exp(v)`
- `np.log(v)`
- `np.abs(v)`
- `np.maximum(v, 0)`
- `v**2`
- `1/v`

Vectorization Logistic Regression

$$X = \begin{bmatrix} \vdots & \vdots & & \vdots \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ \vdots & \vdots & & \vdots \end{bmatrix}, X \in \mathbb{R}^{\mathrm{n_x}} \tag{2}$$

$$Z = \begin{bmatrix} z^{(1)} & z^{(2)} & \cdots & z^{(m)} \end{bmatrix} = w^T X + \begin{bmatrix} b & b & \cdots & b \end{bmatrix} = \begin{bmatrix} w^T x^{(1)} + b & w^T x^{(2)} + b & \cdots & w^T x^{(m)} + b \end{bmatrix} \tag{3}$$

- `Z=np.dot(w.T, X)+b`
- "Broadcasting"

$$dZ = \begin{bmatrix} dz^{(1)} & dz^{(2)} & \cdots & dz^{(m)} \end{bmatrix}, \tag{4}$$
$$dz^{(i)} = a^{(i)} - y^{(i)}$$

$$A = \begin{bmatrix} a^{(1)} & a^{(2)} & \cdots & a^{(m)} \end{bmatrix} \tag{5}$$
$$Y = \begin{bmatrix} y^{(1)} & y^{(2)} & \cdots & y^{(m)} \end{bmatrix}$$

$$dZ = A - Y = \begin{bmatrix} a^{(1)} - y^{(1)} & a^{(2)} - y^{(2)} & \cdots & a^{(m)} - y^{(m)} \end{bmatrix} \tag{6}$$

- `db=np.sum(dZ)/m`

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

- `dw=np.prod(X, dZ.T)/m`

$$dw = \frac{1}{m}\begin{bmatrix} \vdots & \vdots & & \vdots \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ \vdots & \vdots & & \vdots \end{bmatrix}\begin{bmatrix} dz^{(1)} \\ dz^{(2)} \\ \vdots \\ dz^{(m)} \end{bmatrix} \tag{7}$$

Implementing Logistic Regression

```
for iter in range(epoch):
    Z = np.dot(w.T, X) + b
    A = sigmoid(Z)
    dZ = A - Y
    dw = np.dot(X, dZ.T)/m
    db = np.sum(dZ)/m
    w = w - alpha*dw
    b = b - alpha*db
```

Broadcasting in Python

- General principle (element-wise operation)

$$\begin{aligned} (m,n) +/-/\times/\div \underbrace{(m,1)}_{converted\ to\ (m,n)} &\rightarrow (m,n) \\ (m,n) +/-/\times/\div \underbrace{(1,n)}_{converted\ to\ (m,n)} &\rightarrow (m,n) \\ (m,1)/(1,n) +/-/\times/\div const. &= (m,1)/(1,n) \end{aligned} \tag{8}$$

Don't use rank 1 array in numpy

Example:

- rank 1 array

> `a = np.random.randn(5)`
>
> `a.shape`
>
> > (5,)

- vector

> `a.shape`
>
> > (5,1)/(1,5)

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

- `reshape(5,1)`

Logistic Regression cost function

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

$$p(y|x) = \hat{y}^y (1-\hat{y})^{(1-y)} \tag{9}$$
$$log[\ p(y|x)] = \hat{y}log(y) + (1-\hat{y})log[(1-y)] = -\mathscr{L}(\hat{y}^{(i)}, y^{(i)})$$

Cost on m examples (assumed independent identity distribution)

- Maximum Likelihood Estimation

$$log\, p(labels\ in\ training\ set) = log\ \prod_{i=1}^{m} p(y^{(i)}|x^{(i)}) \tag{10}$$
$$= \sum_{i=1}^{m} \underbrace{log\, p(y^{(i)}|x^{(i)})}_{-\mathscr{L}(\hat{y}^{(i)}, y^{(i)})}$$

- Cost $\mathscr{J}(w, b)$: to minimize the cost but not maximize the estimation,

$$\frac{1}{m} \sum_{i=1}^{m} \mathscr{L}(\hat{y}^{(i)}, y^{(i)}) \tag{11}$$