Dismiss

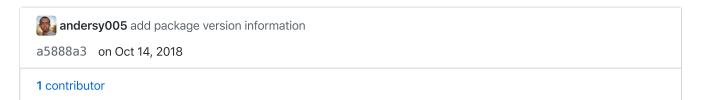
Join GitHub today

GitHub is home to over 40 million developers working together to host and review code, manage projects, and build software together.

Sign up

Branch: master ▼ Find file Copy path

deep-learning-specialization-coursera / 01-Neural-Networks-and-Deep-Learning / week2 /
02-vectorization.ipynb



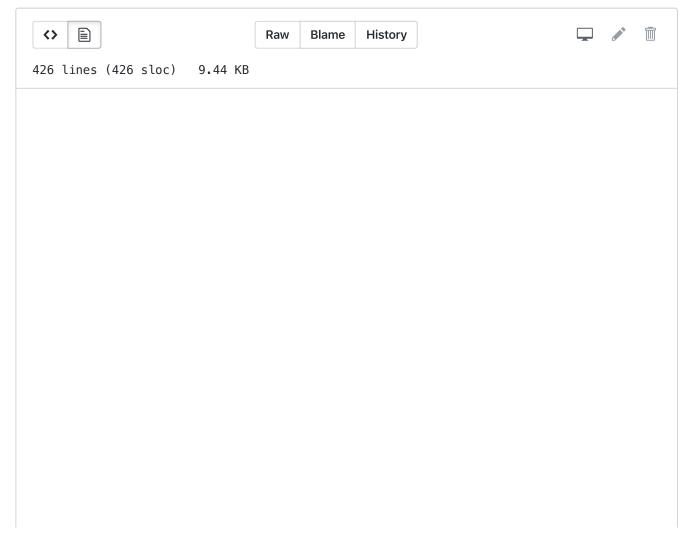
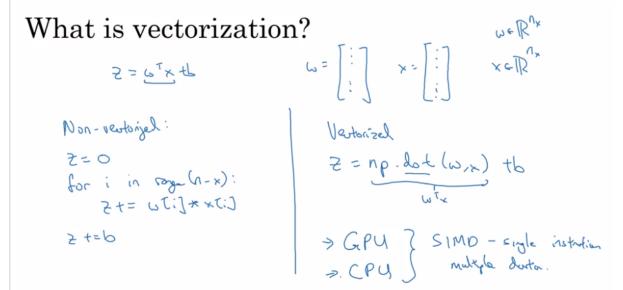


Table of Contents

- 1 Vectorization
- 2 Vectorizing Logistic Regression
- 3 Vectorizing Logistic Regression's Gradient Output
- 4 A note on python/numpy vectors

Vectorization



Neural network programming guideline

Whenever possible, avoid explicit for-loops.

$$U = AV$$

$$U_{i} = \sum_{j} \sum_{i} A_{ij} V_{j}$$

$$U = np \cdot dot (A, v)$$

$$dor i \dots \qquad \leftarrow$$

$$dor i \dots \leftarrow$$

$$u \in Ai : Ti = Ai : Ti = V$$

Vectors and matrix valued functions

Say you need to apply the exponential operation on every element of a matrix/vector.

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

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

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

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

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

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

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

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

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

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

Logistic regression derivatives

$$J = 0, \quad dw1 = 0, \quad dw2 = 0, \quad db = 0$$

$$\Rightarrow \text{for } i = 1 \text{ to } n:$$

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

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

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

$$dz^{(i)} = a^{(i)}(1-a^{(i)})$$

$$dw_{1} += x_{2}^{(i)}dz^{(i)}$$

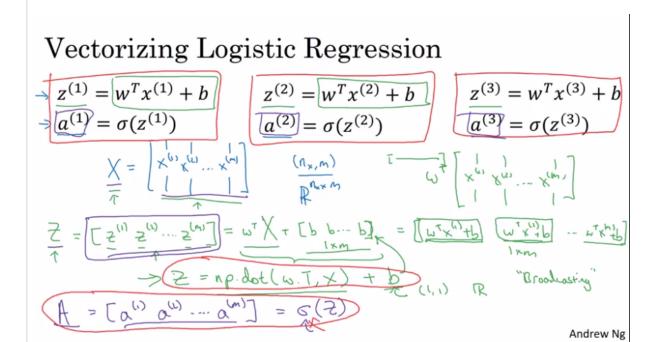
$$dw_{2} += x_{2}^{(i)}dz^{(i)}$$

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

$$J = J/m, \quad dw_{1} = dw_{1}/m, \quad dw_{2} = dw_{2}/m, \quad db = db/m$$

$$d\omega /= m$$

Vectorizina Loaistic Rearession



Vectorizing Logistic Regression's Gradient Output

Implementing Logistic Regression

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

$$for i = 1 to m:$$

$$z^{(i)} = w^T x^{(i)} + b \\
a^{(i)} = \sigma(z^{(i)}) \\
J + = -[y^{(i)} \log a^{(i)} + (1 - y^{(i)}) \log(1 - a^{(i)})]$$

$$dz^{(i)} = a^{(i)} - y^{(i)} \\
dw_1 + = x_1^{(i)} dz^{(i)} \\
dw_2 + = x_2^{(i)} dz^{(i)}$$

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

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

$$db = db/m$$

$$dcoo iter in rowse(1000):

$$= n p \cdot dot (\omega.T, \chi) + b$$

$$A = c(z)$$

$$A = A - C$$

$$A = m \times A = T$$

$$A$$$$

Andrew Ng

A note on python/numpy vectors

```
In [6]: a = np.random.randn(5)
Out[6]: array([-0.91796822, -0.53903443, 1.00289266, 0.22272871, -
        0.35617949])
```

```
In [7]: a.shape
Out[7]: (5,)
 In [8]: a.T
Out[8]: array([-0.91796822, -0.53903443, 1.00289266, 0.22272871, -
         0.356179491)
In [9]: np.dot(a, a.T)
Out[9]: 2.3154893533786054
In [10]: a = np.random.randn(5, 1)
Out[10]: array([[-1.26834861],
                [-0.254855]
                [-1.37786229],
                [ 0.18718574],
                [-1.31341244]])
In [11]: a.shape
Out[11]: (5, 1)
In [12]: a.T
Out[12]: array([[-1.26834861, -0.254855 , -1.37786229, 0.18718574,
         -1.31341244]
In [13]: np.dot(a, a.T)
Out[13]: array([[ 1.60870819, 0.32324498, 1.74760972, -0.23741677,
         1.66586484],
                [ 0.32324498, 0.06495107, 0.35115509, -0.04770522,
         0.33472972],
                [ 1.74760972, 0.35115509, 1.8985045 , -0.25791618,
         1.809701471,
                [-0.23741677, -0.04770522, -0.25791618, 0.0350385]
         -0.245852081,
                [ 1.66586484, 0.33472972, 1.80970147, -0.24585208,
         1.72505223]])
```

Python/numpy vectors

```
a = np.random.randn(5)

a shape = (5,)

bon't we

"conk | array"

a = np.random.randn(5,1) \rightarrow a.shape = (5,1)

vector \checkmark

a = np.random.randn(1,5) \rightarrow a.shape = (1,5)
```

assert (a. shape ==
$$(5,1)$$
) \leftarrow
 $Q = Q \cdot (eshape((5,1)))$

Out[14]:

Software	Version
Python	3.6.6 64bit [GCC 4.2.1 Compatible Apple LLVM 6.1.0 (clang-602.0.53)]
IPython	7.0.1
os	Darwin 17.7.0 x86_64 i386 64bit
numpy	1.15.1
Sun Oct 14 19:41:16 2018 MDT	