Basics of Neural Network Programming

deeplearning.ai

Vectorizing Logistic Regression's Gradient Computation

在反向传播中使用向量化

Vectorizing Logistic Regression

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

$$\boxed{dZ = [dz^{(1)} \; dz^{(2)} \; \cdots \; dz^{(m)}]} \leftarrow$$

$1 \times m$

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

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

$$\rightarrow dw = 0$$

$$dw \mathrel{+}= x^{(1)} dz^{(1)}$$

$$dw \mathrel{+}= x^{(2)} dz^{(2)}$$

$$\vdots$$

$$dw \mathrel{/}= m$$

$$\cancel{dw_1} \quad \cancel{dw_2} \quad \vdots$$

$$db = 0$$

$$db \mathrel{+}= dz^{(1)}$$

$$db \mathrel{+}= dz^{(2)}$$

$$\vdots$$

$$db \mathrel{+}= dz^{(m)}$$

$$db \mathrel{/}= m$$

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

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

$$= \frac{1}{m} \begin{bmatrix} | & & | \\ x^{(1)} & \cdots & x^{(m)} \\ | & & | \end{bmatrix} \begin{bmatrix} dz^{(1)} \\ \vdots \\ dz^{(m)} \end{bmatrix}$$

$$= \frac{1}{m} \left[x^{(1)} dz^{(1)} + \cdots + x^{(m)} dz^{(m)} \right]$$

$n \times 1$

Implementing Logistic Regression

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

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

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

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

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

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

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

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

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

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

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

$db = db/m$

for iter in range(1000):   gd

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

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

$$A = \sigma(Z)$$

$$dZ = A - Y$$

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

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

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

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

Andrew Ng