



deeplearning.ai

Optimization Algorithms

Mini-batch
gradient descent

Batch vs. mini-batch gradient descent

x, y

$x^{\{t\}}, y^{\{t\}}$

Vectorization allows you to efficiently compute on m examples.

$$X = \begin{bmatrix} x^{(1)} & x^{(2)} & x^{(3)} & \dots & x^{(1000)} & | & x^{(1001)} & \dots & x^{(2000)} & | & \dots & | & \dots & x^{(m)} \end{bmatrix}$$

(n_x, m) $X^{\{1\}} (n_x, 1000)$ $X^{\{2\}} (n_x, 1000)$ $X^{\{5,000\}} (n_x, 1000)$

$$Y = \begin{bmatrix} y^{(1)} & y^{(2)} & y^{(3)} & \dots & y^{(1000)} & | & y^{(1001)} & \dots & y^{(2000)} & | & \dots & | & \dots & y^{(m)} \end{bmatrix}$$

$(1, m)$ $Y^{\{1\}} (1, 1000)$ $Y^{\{2\}} (1, 1000)$ $Y^{\{5,000\}} (1, 1000)$

What if $m = \underline{5,000,000}$?

5,000 mini-batches of 1,000 each

Mini-batch t : $x^{\{t\}}, y^{\{t\}}$

$x^{(i)}$

$z^{[l]}$

$x^{\{t\}}, y^{\{t\}}$

Mini-batch gradient descent

repeat {
for $t = 1, \dots, 5000$ {

Forward prop on $X^{\{t\}}$.

$$Z^{(1)} = W^{(1)} X^{\{t\}} + b^{(1)}$$

$$A^{(1)} = g^{(1)}(Z^{(1)})$$

$$\vdots$$

$$A^{(L)} = g^{(L)}(Z^{(L)})$$

Vectorized implementation
(1000 examples)

Compute cost $J^{\{t\}} = \frac{1}{1000} \sum_{i=1}^L \ell(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2 \cdot 1000} \sum_{\ell} \|W^{(\ell)}\|_F^2$.

↙ ↘ from $X^{\{t\}}, Y^{\{t\}}$

Backprop to compute gradients w.r.t $J^{\{t\}}$ (using $(X^{\{t\}}, Y^{\{t\}})$)

$$W^{(1)} := W^{(1)} - \alpha dW^{(1)}, \quad b^{(1)} := b^{(1)} - \alpha db^{(1)}$$

"1 epoch"

pass through training set.

1 step of grad desc
using $X^{\{t+1\}}, Y^{\{t+1\}}$.
(as if $m=1000$)

X, Y