deeplearning.ai

Optimization Algorithms

Mini-batch gradient descent

Batch vs. mini-batch gradient descent

X, Y $\quad$ $X^{\{t\}}, Y^{\{t\}}$.

Vectorization allows you to efficiently compute on m examples.

$X = [\, x^{(1)} \; x^{(2)} \; x^{(3)} \; \cdots \; x^{(1000)} \mid x^{(1001)} \; \cdots \; x^{(2000)} \mid \cdots \mid \cdots \; x^{(m)} \,]$

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

$Y = [\, y^{(1)} \; y^{(2)} \; y^{(3)} \; \cdots \; y^{(1000)} \mid y^{(1001)} \; \cdots \; y^{(2000)} \mid \cdots \mid \cdots \; y^{(m)} \,]$

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

What if $m = 5,000,000$?

5,000 mini-batches of 1,000 each

Mini-batch t: $X^{\{t\}}, Y^{\{t\}}$.

$x^{(i)}$ i example

$z^{[l]}$ l layer

$X^{\{t\}}, Y^{\{t\}}$. t mini-batch

Andrew Ng

Mini-batch gradient descent

Repeat {
  for $t = 1, \ldots, 5000$ {

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

X, Y

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 $J^{\{t\}} = \frac{1}{1000} \sum_{i=1}^{l} \mathcal{L}(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2 \cdot 1000} \sum_{l} \|W^{[l]}\|_F^2$.
(from $X^{\{t\}}, Y^{\{t\}}$)

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

$W^{[l]} := W^{[l]} - \alpha dW^{[l]}$, $b^{[l]} := b^{[l]} - \alpha db^{[l]}$

  }
}

"1 epoch" = pass through training set.