Mini-Batch Gradient Descent

(eg. $X = 512{,}000$, mini-batch 512)

for $t = 1, \ldots, 1000$:

Forward Prop on $X^{\{t\}}$

Compute cost: $J^{\{t\}} = \frac{1}{512} \sum_{i=1}^{l} \mathcal{L}(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2 \times 512} \sum \|w^{[l]}\|_F^2$

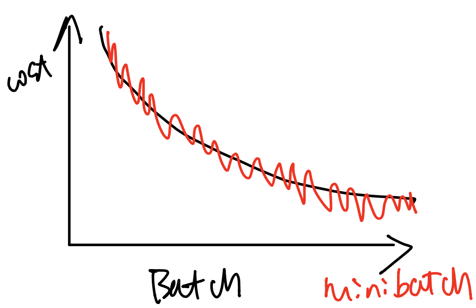
Backprop to compute gradient $J^{\{t\}}$

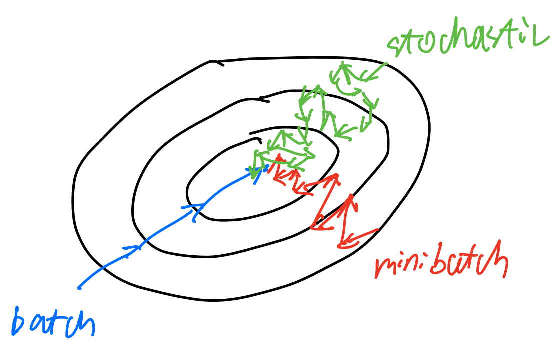
$w^{[l]} := w^{[l]} - \alpha\, dw^{[l]}$, $b^{[l]} := b^{[l]} - \alpha\, db^{[l]}$

Basically same as gradient descent but run on each smaller batch so as to save computation time

Batch : whole dataset $(X^{\{t\}}, y^{\{t\}}) = (X, y)$

Stochastic : 1 training example $(X^{\{1\}}, y^{\{1\}}) = (X^{(1)}, y^{(1)})$





Batch : too slow for each iterations

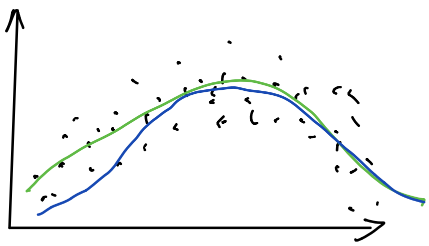
Stochastic : lose speeding from vectorization

mini-batch : Fastest learning, make progress without waiting to process entire dataset.

$n = \{64, 128, 256, 512\}$

$2^6 \quad 2^7 \quad 2^8 \quad 2^9$ → CPU / GPU optimization

Exponentially Weighted Average



$V_t = \beta_t V_{t-1} + (1-\beta_t)\theta_t$

$\beta = 0.9 \approx$ looking at past 10 days.

V_t as approximately $\frac{1}{1-\beta}$

Bias correction

Problem: initially $V_t = 0$, the line looks like the blue one.

solution: Let $V_t = \frac{V_t}{1-(\beta)^t}$ → as t increase, $1-(\beta)^t$ close to 1

Gradient Descent with Momentum

On iteration t:

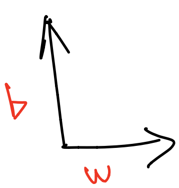
Compute dw, db --- on mini-batch

$V_{dw} = \beta V_{dw} + (1-\beta) dw$

$V_{db} = \beta V_{db} + (1-\beta) db$ → acceleration

$W := W - \alpha V_{dw}$, $b := b - \alpha V_{db}$

friction velocity learning rate



not weight, bias, just features, could also be w_1, w_2 .....

RMSProp

On iteration t:

compute dw, db --- on mini-batch

$S_{dw} = \beta S_{dw} + (1-\beta) dw^2$

$S_{db} = \beta S_{db} + (1-\beta) db^2$

$\}$ slow down vertical and speed up horizontal

$W := W - \alpha \frac{dw}{\sqrt{S_{dw}}}$ $b := b - \alpha \frac{db}{\sqrt{S_{db}}}$

dampen out oscillation (can use larger α)

Adam : a combination of Momentum and RMSProp

$V_{dw}, S_{dw}, V_{db}, S_{db} = 0$

on iteration t:

compute dw, db use mini-batch

$V_{dw} = \beta_1 V_{dw} + (1-\beta_1) dw$, $V_{db} = \beta_1 V_{db} + (1-\beta_1) db$

$S_{dw} = \beta_2 S_{dw} + (1-\beta_2) dw^2$, $S_{db} = \beta_2 S_{db} + (1-\beta_2) db^2$

$V^{corrected}_{dw} = V_{dw} / (1-\beta_1^t)$, $V^{corrected}_{db} = V_{db} / (1-\beta_1^t)$

$S^{corrected}_{dw} = S_{dw} / (1-\beta_2^t)$, $S^{corrected}_{db} = S_{db} / (1-\beta_2^t)$

$$w \mathrel{-}= \alpha \frac{V^{corrected}_{dw}}{\sqrt{S^{corrected}_{dw}} + \varepsilon} \qquad b \mathrel{-}= \alpha \frac{V^{corrected}_{db}}{\sqrt{S^{corrected}_{db}} + \varepsilon}$$

avoid division by zero, typically 10^{-8}

hyperparams: $\alpha \rightarrow$ tune $\quad \beta_1 = 0.9 \quad \beta_2 = 0.999 \quad \varepsilon = 10^{-8}$

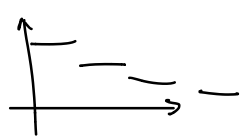
Learning Rate Decay : slowly decrease your learning rate.

$$\alpha = \alpha_0 \frac{1}{1 + \text{decay rate} * \text{epoch number}}$$

exponential decay $\quad 0.95^{\text{epoch-num}} \cdot \alpha_0$

$$\alpha = \frac{k}{\sqrt{\text{epoch-num}}} \cdot \alpha_0$$

Discrete decay



Manual Decay