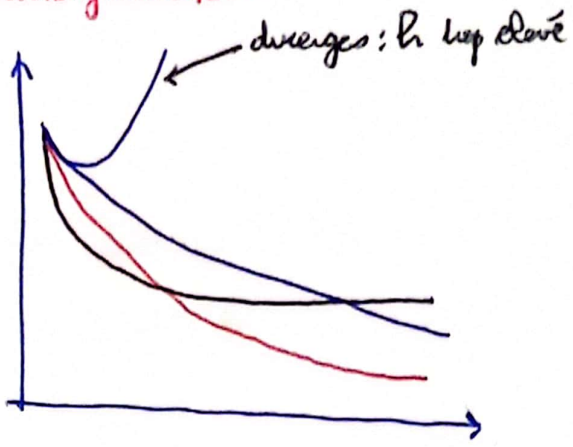
advanced optimization algorithms

① gradient descent

momentum = 0.9

learning rate : fixed or learning RateScheduler

$$w_j = w_j - \underbrace{\alpha}_{\text{learning rate}} \cdot \frac{\partial}{\partial w_j} J(w,b)$$



Ordre croissant du lr :

bleue → rouge → verte

② ADAM algorithm

adaptive moment estimation

not just one α

$$w_1 = w_1 - \alpha_1 \frac{\partial}{\partial w_1} J(w,b)$$

$$\vdots$$

$$w_{10} = w_{10} - \alpha_{10} \frac{\partial}{\partial w_j} J(w,b)$$

$$b = b - \alpha_{11} \frac{\partial}{\partial w_j} J(w,b)$$

⇒ ① if w_j keeps moving in same direction

⇒ increase α_j ⇒ go faster in that direction

② if w_j keeps oscillating =

⇒ reduce α_j

Optimization algorithms

Batch VS mini-batch gradient descent:

Vectorization allows you to efficiently compute on m examples

$$X = \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix}$$

$$Y = \begin{bmatrix} y^{(1)}, \dots, y^{(m)} \end{bmatrix}$$

if we have $m = 5{,}000{,}000$?

the algo will be slow

⇓ solution

split the entire dataset into mini-batches of 1,000 examples each

$X^{\{1\}}$: first mini batch | $Y^{\{1\}}$

$X^{\{2\}}$: second mini batch | $Y^{\{2\}}$

⋮

$X^{\{5000\}}$: 5000-mini batch | $Y^{\{5000\}}$

⇓

for $t = 1$ to 5000:

- forward prop on $X^{\{t\}}$

$$Z^{[1]} = W^{[1]} X^{\{t\}} + b^{[1]}$$
$$A^{[1]} = g^{[1]}(Z^{[1]})$$
$$\vdots$$
$$A^{[L]} = g^{[L]}(Z^{[L]})$$

- compute cost: $J = \frac{1}{1000} \sum_{i=1}^{1000} \mathcal{L}(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2 \cdot 1000} \sum_{l} \| W^{[l]} \|_F^2$

1000 = size of the mini batch

$\hat{y}^{(i)}, y^{(i)}$ from $X^{\{t\}}$ and $Y^{\{t\}}$

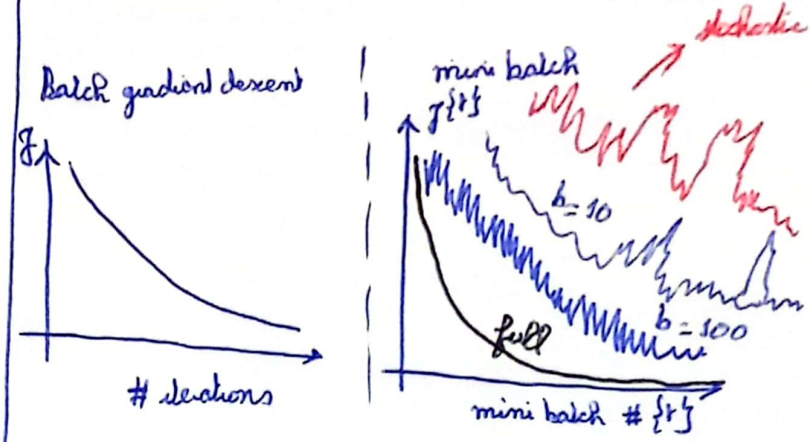
- Backpropagation

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

⋮

}

This step is called 1 epoch: one pass through the training set (one iteration of gradient descent)



the cost J must decrease in every iteration

1 iteration per epoch

~~mini batch~~ the cost might not decrease in every iteration

but if we plot $J^{\{t\}}$ | $\frac{n}{m}$ iterations per epoch

parameter to choose: mini batch size

if mini batch size = m : batch gradient descent

$$\{X^{\{1\}}, Y^{\{1\}}\} = \{X, Y\}$$

if mini batch size = 1 : stochastic gradient descent : n iterations per epoch

every example is its own mini batch

(doesn't converge to the min : oscillates around him)

see opti course

Loose speed from vectorization

In practice: mini batch should be between 1 and m

↓

un mini batch trop petit entraine un bruit trop grand sur l'estimation du gradient

⇒ empêcher la convergence

⇒ best choice: 562 mini batch

if we have small training set : use batch gradient descent

$(m \leq 2000)$

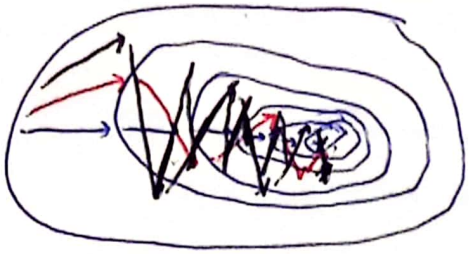
else :

typical mini batch size : 64, 128, 256, 512, 1024

(usually a power of 2)

⚠ make sure that mini batch fit in CPU/GPU memory

descente de gradient batch



descente de gradient mini batch

stochastique : $\theta_j^{\{k+1\}} = \theta_j^{\{k\}} - \eta_k \frac{\partial \ell_i(\theta^{(k)})}{\partial \theta_j}$

convergence du stochastic gradient descent

1/ $\sum_k \eta_k < +\infty$ et $\sum_{k=1} \eta_k^2 = +\infty$

2/ des hypotheses de regularité sur J

Alors la suite $\theta^{(k)}$ converge vers un min local de J

Remarque

momentum : pour réduire la variance et limiter les oscillations + aider de rester bloqué dans un min local

exponentially weighted averages

example : Temperature in London

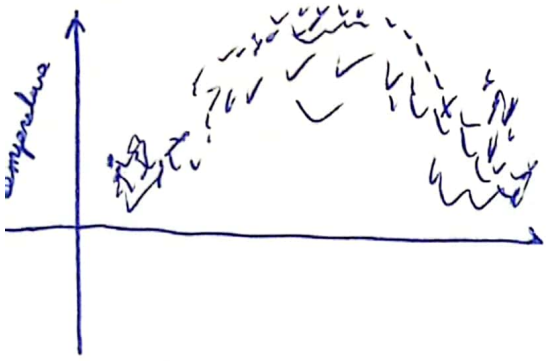
$\theta_1 : 40°F \quad 4°C$

$\theta_2 : 49°F \quad 9°C$

$\vdots$

$\theta_{180} = 60°F \quad 15°C$

$\theta_{181} = 56°F$



$V_0 = 0$

$V_1 = 0.9\, V_0 + 0.1\, \theta_1$

$V_2 = 0.9\, V_1 + 0.1\, \theta_2$

$\vdots$

$V_t = 0.9\, V_{t-1} + 0.1\, \theta_t$

if we plot V_t : we obtain exponentially weighted averages

$V_t = \beta \cdot V_{t-1} + (1-\beta)\, \theta_t$

$\simeq \frac{1}{1-\beta}$ days temperature

$\beta = 0.9$: $\simeq$ averaging over $\frac{1}{1-0.9} = 10$ days

$\beta = 0.98$ $\simeq$ averaging over $\frac{1}{1-0.98} \simeq 50$ days

$V_{100} = 0.1\, \theta_{100} + 0.1 \times 0.9 \times \theta_{99} + 0.1 \times (0.9)^2\, \theta_{98}$

$+ \dots + 0.1 \times (0.9)$

$(0.9)^{10} \simeq 0.35 = \frac{1}{e}$

$(1-\varepsilon)^{1/\varepsilon} = \frac{1}{e}$

Bias correction in

we start very low due to initialization $V_0 = 0$

$\Rightarrow$ at the beginning

instead of taking V_t we take $\frac{V_t}{1-\beta^t}$

Gradient descent with momentum

we compute an exponentially weighted average on the gradients

$\Rightarrow$ compute faster : tends to smooth the gradient descent : less oscillations

momentum :

on iteration t :

compute dw, db on current mini batch

vélocité v :
$$V_{dw} = \beta \cdot V_{dw} + (1-\beta)\, dw$$
$$V_{db} = \beta \cdot V_{db} + (1-\beta) \cdot db$$

$$w := w - \alpha \cdot V_{dw}$$

$$b := b - \alpha \cdot V_{db}$$

vélocité : direction dans laquelle les paramètres vont être modifiés

Hyperparameters : α, β

if $\beta = 0.9$ $\simeq$ average over last $\simeq$ 10 gradients

RMS prop algorithm

on iteration t:

compute dw, db on current mini batch

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

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

$$w = w - \alpha \cdot \frac{dw}{\sqrt{S_{dw}}}$$

$$b = b - \alpha \cdot \frac{db}{\sqrt{S_{db}}}$$

we hope that S_{dw} ~~is~~ to be small if we want learning be fast in the direction of w

we hope that S_{db} to be large if we want learning be slow in the direction of b

Adam optimization algorithm

$V_{dW} = 0 \quad S_{dW} = 0 \quad V_{db} = 0 \quad V_{dW} = 0$

on iteration t

compute dW, db using current mini batch

$V_{dW} = \beta_1 V_{dW} + (1-\beta) dW$

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

$S_{dW} = \beta_2 S_{dW} + (1-\beta) \underbrace{dW^2}_{\text{element wise squaring}}$

$S_{db} = \beta_2 S_{db} + (1-\beta) \cdot db^2$

$$V_{dW}^{corrected} = \frac{V_{dW}}{(1-\beta_1^t)}$$

$$V_{db}^{corrected} = \frac{V_{db}}{(1-\beta_1^t)}$$

$$S_{dW}^{corrected} = \frac{S_{dW}}{(1-\beta_2^t)}$$

$$S_{db}^{corrected} = \frac{S_{db}}{(1-\beta_2^t)}$$

$$W = W - \alpha \frac{V_{dW}^{corrected}}{\sqrt{S_{dW}^{corrected}} + \varepsilon}$$

$$b = b - \alpha \cdot \frac{V_{db}^{corrected}}{\sqrt{S_{db}^{corrected}} + \varepsilon}$$

Hyperparameters: α : needs to be tuned

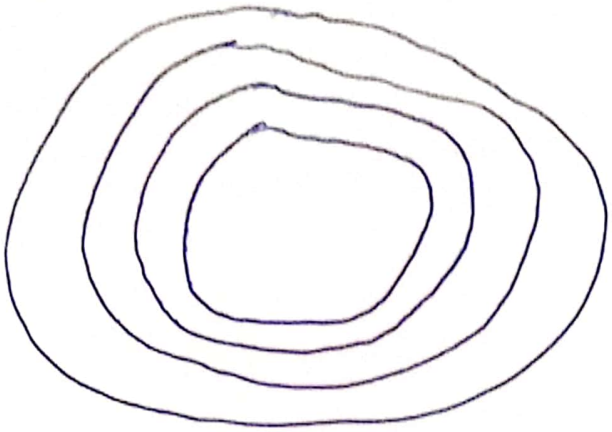
$\beta_1 \simeq 0.9$
$\beta_2 \simeq 0.999$
$\text{epsilon} \simeq 10^{-8}$

} usually default values usually

Adam : adaptative moment estimation

learning rate decay

we slowly decrease alpha while learning



1 epoch = 1 pass through the data

$$\alpha = \frac{1}{1 + \text{decay\_rate} * \text{epoch\_num}} \cdot \alpha_0$$

other methods

exponential $\alpha = 0,95^{\text{epoch\_num}} \cdot \alpha_0$

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

$$\alpha = \frac{k}{\sqrt{t}} \cdot \alpha_0$$