Courses - 2
Week - 2

OPTIMIZATION ALGORITHMS

From prev. Discussion
↓
Vectorization allows us to efficiently compute on 'm' examples.

$$X = [x^{(1)} \quad x^{(2)} \quad \dots \quad x^{(m)}]_{(n_x, m)}$$

$$Y = [y^{(1)} \quad y^{(2)} \quad \dots \quad y^{(m)}]_{1 \times m}$$

If m = 50,00,000 examples

Usually forward

X to be computed (if considering all m examples) and then compute loss → gradient descent

lot of time consuming

↓ sol

Mini-Batch Gradient descent

Scanned with CamScanner

50 lakh

mini batch gradient descent

Let $m = 50{,}00{,}000$

$$X = \Big[\underbrace{x^{(1)}\ x^{(2)}\ \cdots\ x^{(1000)}}_{X^{\{1\}}} \Big|,\ \underbrace{x^{(1001)}\ \cdots\ x^{(2000)}}_{X^{\{2\}}},\ \cdots \Big|\ \underbrace{x^{(m)}}_{X^{\{5000\}}}\Big]$$

$n \times m$

$$Y = \Big[\underbrace{y^{(1)}\ y^{(2)}\ \cdots\ y^{(1000)}}_{Y^{\{1\}}},\ \underbrace{y^{(1001)}\ \cdots\ y^{(2000)}}_{Y^{\{2\}}},\ \cdots \Big|\ \underbrace{y^{(m)}}_{Y^{\{5000\}}}\Big]$$

$1 \times m$

If $m = 50{,}00{,}000$

-> we are not do all at once as a single batch

-> divide the set of m examples into mini-batches (say each batch of size 1000)

$\therefore$ # mini-batches = 5000

Notation: $X^{\{t\}} \rightarrow$ mini batch (t^{th} mini)

$$\boxed{\text{mini batch } t : \left(X^{\{t\}}, Y^{\{t\}}\right)}$$

dimension of $X^{\{t\}} = n \times 1000$

$Y^{\{t\}} = 1 \times 1000$

Scanned with CamScanner

mini-batch GD Algorithm

1 step of GD using $X^{\{t\}}, Y^{\{t\}}$

for $t = 1, \ldots, 5000$

forward prob 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 example

Compute cost $= J^{\{t\}}$ from minibatch $X^{\{t\}}, Y^{\{t\}}$

$$= \frac{1}{1000} \sum_{i=1}^{l} \mathcal{L}(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2 \times 1000} \sum_{l} \|W^{[l]}\|_F^2$$

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

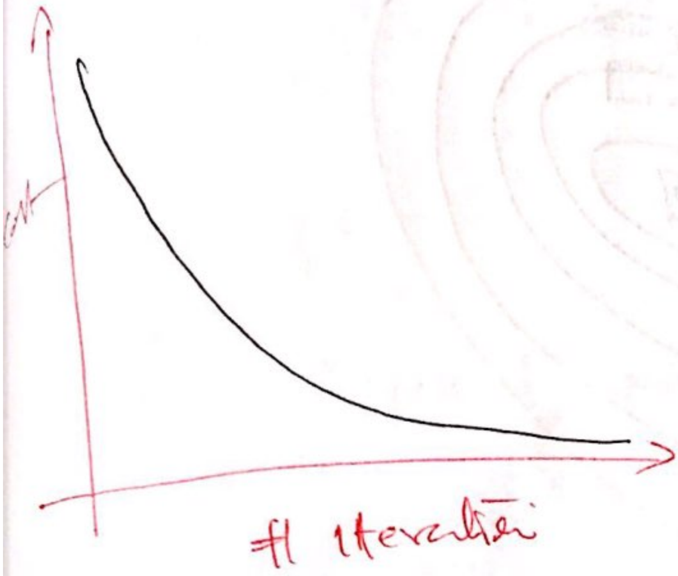
$$W^{[l]} := W^{[l]} - \alpha\, dW^{[l]}, \quad b^{[l]} = b^{[l]} - \alpha\, db^{[l]}$$

}

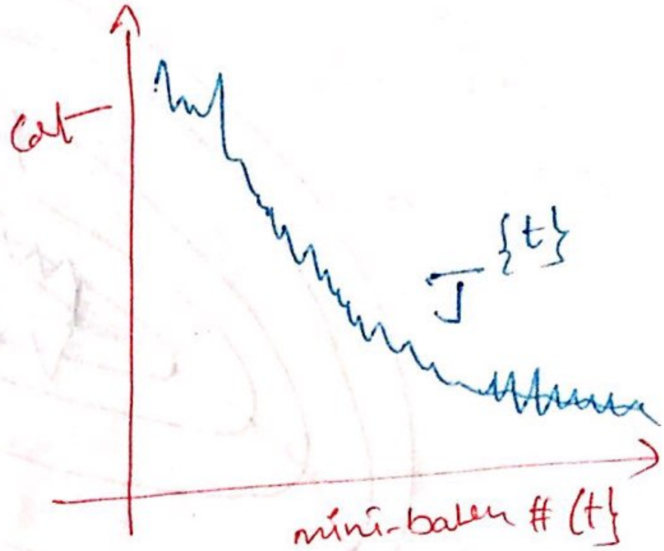
end of 1 epoch
↓
one pass through training set

Scanned with CamScanner

Batch Gradient



mini-batch



Q.? minibatch size

① If (each) minibatch size = m (equal to # examples)

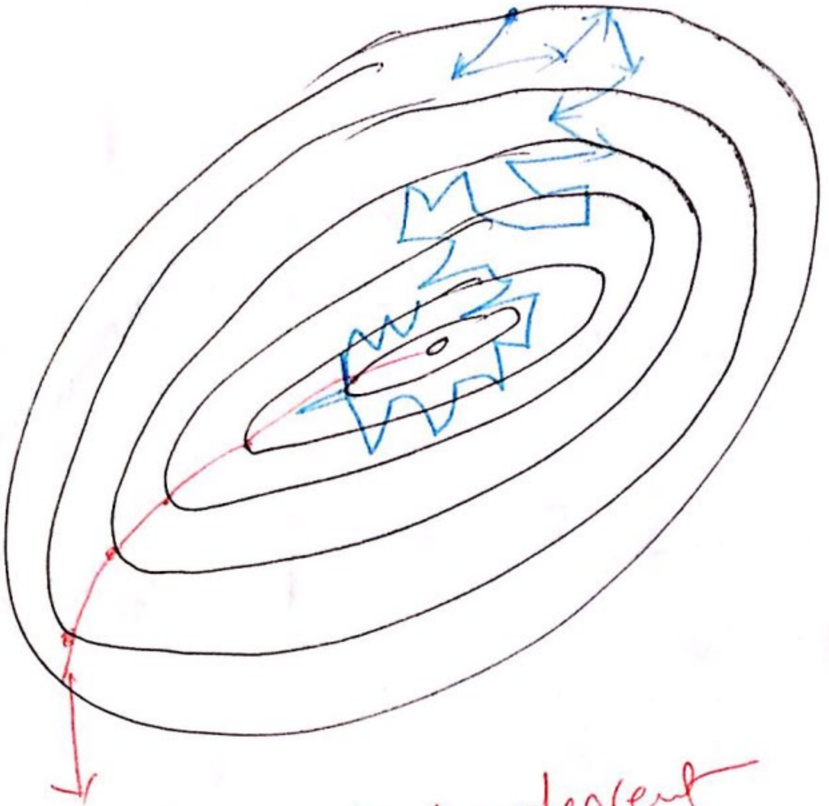
↳ It is same as Batch Gradient

② If minibatch size = 1 ⇒ Stochastic Gradient Descent

every single example become mini-batch

⇒ # mini batchs = m

⇒ $(x^{\{i\}}, y^{\{i\}}) = (x^{(i)}, y^{(i)})$



batch gradient descent

Stochastic gradient descent

batch GD → too long per iteration

Stochastic GD → very slow in convergence

Summary

mini-batch size should not be TOO LARGE & TOO SMALL

($\neq m$) ($\neq 1$)

↓ faster learning

Scanned with CamScanner

Note

Set - ~~...~~ training set is small

① If $m < 2000 \rightarrow$ Go for batch GD

② typical mini-batch size

2^{x} (eg 64, 128, 256 (2^8), 512, ...)

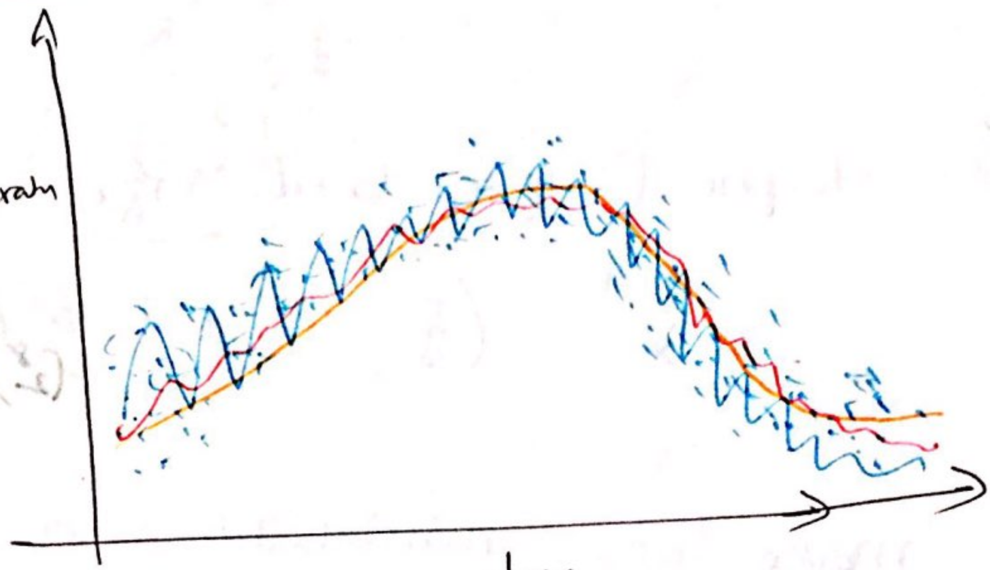
③ make sure mini-batch

$x^{\{t\}}$ $y^{\{t\}}$ fits in CPU/GPU (memory)

Scanned with CamScanner

Algorithm faster than GD

$\boxed{\text{Exponentially weighted moving averages}}$



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

for $\beta = 0.9$ $\Rightarrow$ V_t is approximately averaging over $\frac{1}{1-\beta}$ days temperature

i.e. for $\beta = 0.9$ $\Rightarrow$ averaging $\approx$ 10 days temperature

for $\beta = 0.98$ $\Rightarrow$ averaging $\approx$ 50 days temperature

$\beta = 0.5$ $\Rightarrow$ averaging $\approx$ 2 days temperature

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

exponentially weighted averaged formula

let us compute $\beta = 0.9$

$\boxed{\theta_i = i^{th} \text{ day temperature}}$

$V_{100} = 0.9\, V_{99} + 0.1\, \theta_{100}$

$V_{99} = 0.9\, V_{98} + 0.1\, \theta_{99}$

$V_{98} = 0.9\, V_{97} + 0.1\, \theta_{98}$

$\vdots$

Notation

$V_0 = 0.$

$V_1 = \beta V_0 + (1-\beta)\,\theta_1$

$V_2 = \beta(V_1) + (1-\beta)\,\theta_2$

$\vdots$

$V_\theta = 0$

$V_\theta = \beta V + (1-\beta)\,\theta_1$

$V_\theta =$

$\rightarrow V_\theta = 0$

Repeat {

get next θ_t.

$V_\theta := \beta V_\theta + (1-\beta)\,\theta_t$

}.

Scanned with CamScanner

Bias Correction in exponentially weighted average

$\beta = 0.98$

$t = 2$

$1-\beta^t = 1-(0.98)^2$

$= 0.0396$

$\cancel{V_t}$ $\dfrac{V_t}{1-\beta^t} \rightarrow$ bias correction

as $t \rightarrow$ large

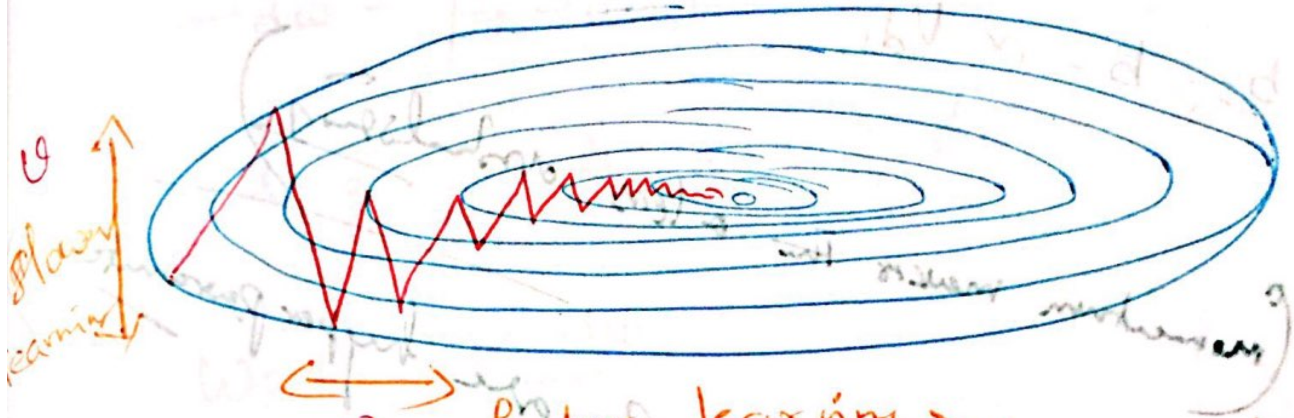
only $V_t = \beta V_{t-1} + (1-\beta)\theta_t$.

need to be followed by

$$\frac{V_t}{1-\beta^t}$$

Gradient descent with moment

key:- Compute exp. weighted average of your gradient and use then to update the new gradients



→ this can be achieved by GD with momentum

momentum

on each iteration t:

Compute dw, db on current mini-batch

$$V_{dw} = \beta \underline{V_{dw}} + (1-\beta) dw$$ → Momentum term (velocity) ; Acceleration

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

$\beta = 0.9$ is high preference

if $\beta = 0$ it is standard Grad. Desc

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

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

earlier (usually)
$w = w - \alpha dw$
$b = b - \alpha db$

often in the literature

$V_{dw} = \beta V_{dw} + (1-\beta)\, dw \rightarrow V_{dw} = \beta V_{dw} + \cancel{(x)}$

$V_{db} = \beta V_{db} + (1-\beta)\, db. \rightarrow V_{db} = \beta V_{db} + \cancel{(x)}$

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

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

(momentum makes ~~the~~ a less oscillation)

(α, β are hyper parameter)

Scanned with CamScanner

RMS prop

on iteration t:

Compute dw, db on current mini-batch

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

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

$$w := w - \alpha \frac{dw}{\sqrt{S_{dw}}}$$ ← small + ε

$$b := b - \alpha \frac{db}{\sqrt{S_{db} + \varepsilon}}$$ ← large

ε is added to avoid '0' in denominator

With this algorithm

slow in (b) direction

faster in (w) direction

Adam optimization algorithm

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

on each iteration t:

Compute dw, db using current mini-batch

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

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

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

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

$V_{dw}^{corrected} = \dfrac{V_{dw}}{(1-\beta_1^t)} \qquad V_{db}^{corrected} = \dfrac{V_{db}}{(1-\beta_1^t)}$

$S_{dw}^{correct} = \dfrac{S_{dw}}{(1-\beta_2^t)} \qquad S_{db}^{correct} = \dfrac{S_{db}}{(1-\beta_2^t)}$

$w := w - \alpha \dfrac{V_{dw}^{corr}}{\sqrt{S_{dw}^{corr}} + \epsilon} \qquad b := b - \alpha \dfrac{V_{db}^{corr}}{\sqrt{S_{db}^{corr}} + \epsilon}$

hyperparameter choice

α : need to be tuned

$\beta_1 = 0.9$

$\beta_2 = 0.999$

$\epsilon = 10^{-8}$

Adam: Adaption moment estimation

Learning Rate decay

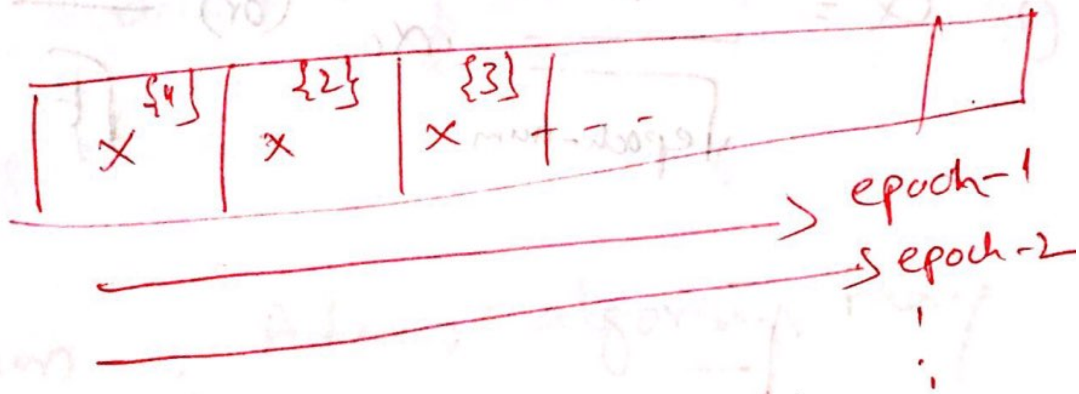
Idea:- Reduce the learning rate towards (Convergence) after initial phases

larger learning — initial

↓

reduce the learning rate as proceed

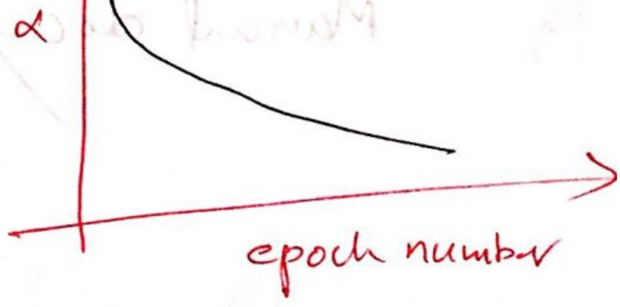
Intuition



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

$\boxed{\text{Say } \alpha_0 = 0.2, \ \text{decay\_rate} = 1}$

Epoch	α
1	0.1
2	0.67
3	0.5
4	0.4
⋮	⋮

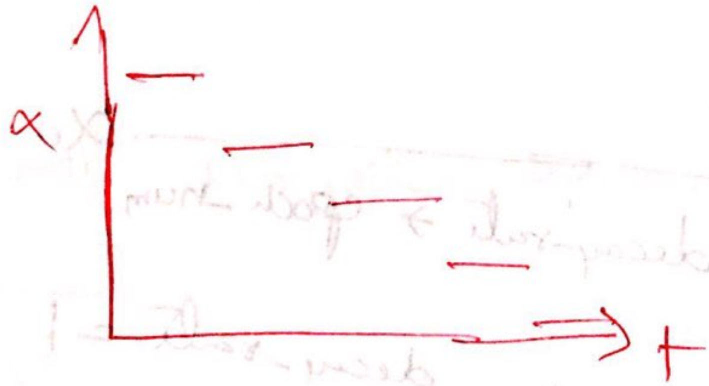


other formulas for decay

① exponential decay

$$\alpha = 0.95^{\text{epoch-num}} \cdot \alpha_0.$$

② $$\alpha = \frac{k}{\sqrt{\text{epoch-num}}} \cdot \alpha_0. \quad \text{(or)} \quad \frac{k}{\sqrt{t}} \cdot \alpha_0$$

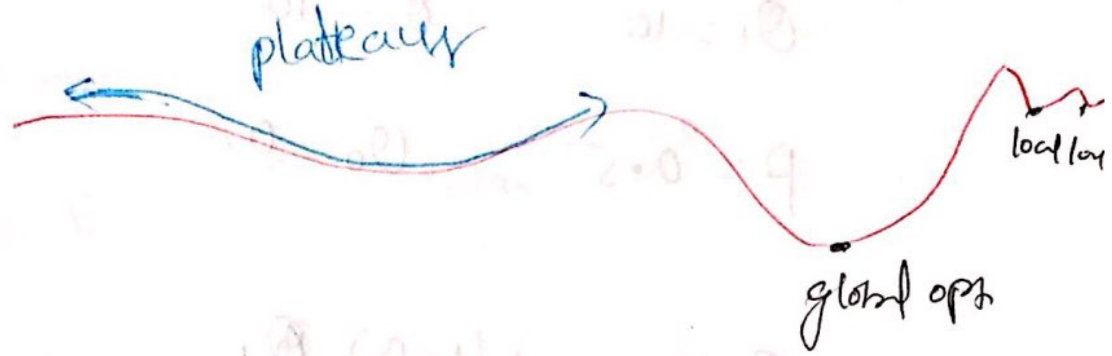
③ 

discrete
Stair case

④ Manual decay

local optima problem

key point



→ plateaus are the regions where the gradient is close to zero for long duration

* momentum, Adam algorithm may overcome "plateau" problems