WEEK 2.

→ you need fast algo's cz its overall a big prob

☆ OPTIMIZATION ALGORITHMS

Mini Batch Gradient

vectorizat^n allow to efficiently calculate m examples

what if m is big?

→ calculate calculate gradient of m examples

→ do gradient descent on m examples

→ calculate again for m.

do gradient descent for m examples

if m is large, this is very slow

Batch vs mini-batch Gradient descent.

↓

you divide your training set (m) into multiple ~~examples~~ batches.

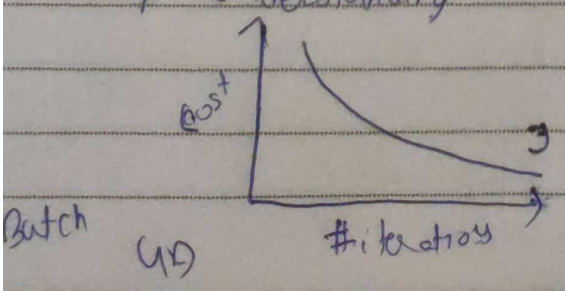
for m = 5M, you divide it into 5000 batches of 1000 each.

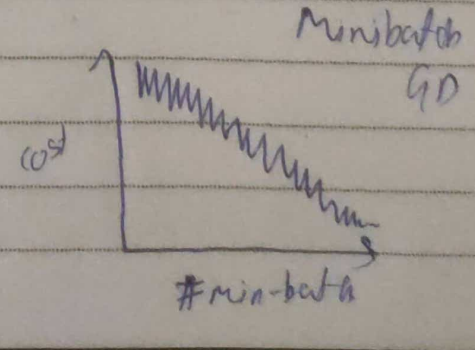
epoch { Now you run gradient descent on batches of 1000

So you'll need a total of 5000 epochs.

☆ minibatch works faster than batch Gradient descent for big m.

understanding Mini batch.





hyperparameter

→ Choosing your mini-batch size.

mini-batch size = m : Batch grad Descent.

mini-batch size = 1 : stochastic GD

big steps
large steps
straight up goes to minima.
but slow

small steps
very noisy
takes zig-zag turn to minima

So usually you'll have something in between 1 and m.

if small training set : use batch gradient descent.

otherwise,

Typical mini-batch size : → 64, 128, 256, 512

Apparently for powers of 2 it is faster

Make sure mini-batch size fits in CPU/GPU memory

Exponentially weighted average.

is used in optimizat^n algo which works faster than GD

Say you want to approximate a very noisy data

eg:-

predict values

$v_0 = 0$

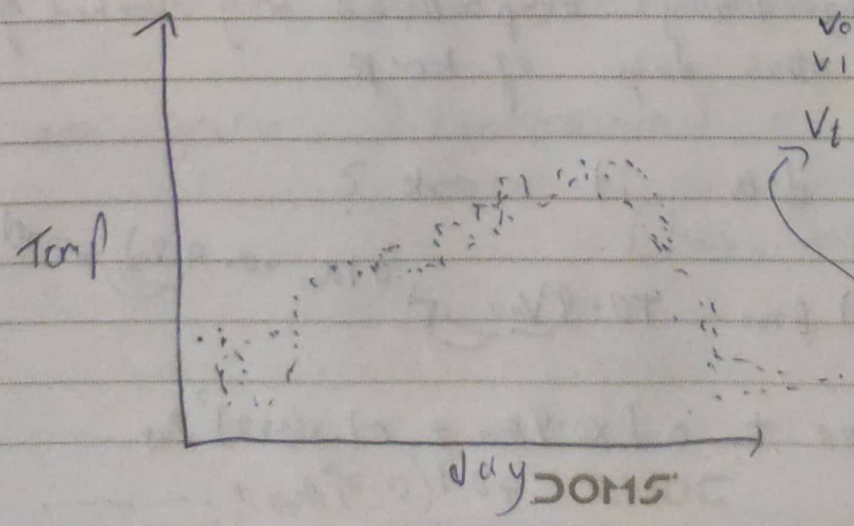
$v_1 = 0.9v_0 + 0.1\theta_1$ ← temp on day 1

$v_t = 0.9v_{t-1} + 0.1\theta_t$

$v \approx$ our average approximation

$\beta = 0.9$

we can chose diff values of β



$V \rightarrow$ exponentially weighted avg.

β is actually a hyperparameter

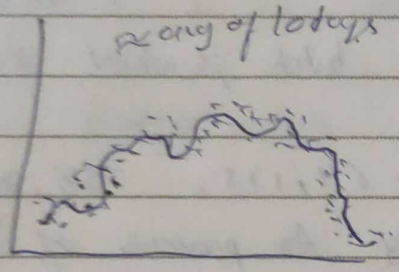
β approximately represents, the no. of days one which we're averaging values to predict today's value

combined with V.

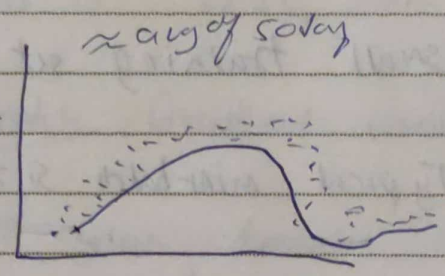
$$V_t = \beta V_{t-1} + (1-\beta)\theta_t$$

↳ $\approx$ avg over $\frac{1}{1-\beta}$ days of temp.

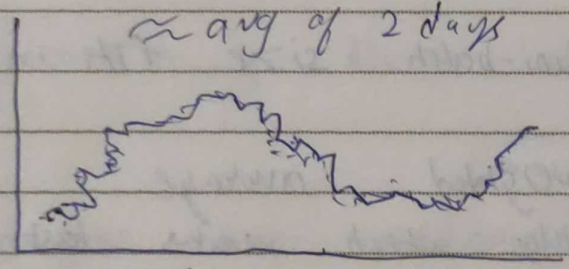
if $\beta = 0.9$ $\approx$ avg of 10 days of temp taken into consideration to predict today's temp



$\beta = 0.9$



$\beta = 0.98$



$\beta = 0.5$

But what are we actually doing here?

↳ approximating today's temp $\approx$ based avg values of some other days of temp

how does it work?

eg:-

$$V_{100} = 0.1\,\theta_{100} + 0.9\,V_{99} \rightarrow (0.1\theta_{99} + 0.9\,V_{98}) \rightarrow \text{and so on.}$$

exponentially decaying fn.

$$= \{0.1\,\theta_{100} + 0.1 \times 0.9\,\theta_{99} + 0.1 \times (0.99)^2\,\theta_{98} + 0.1(0.9)^3\,\theta_{97} + \ldots\ldots$$

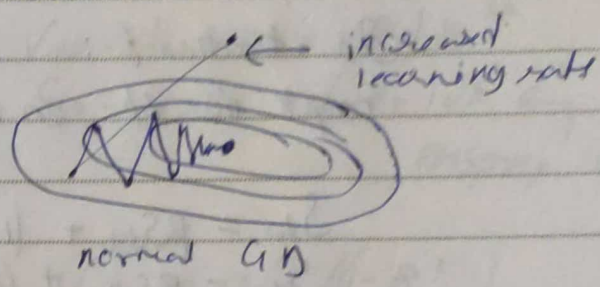
Bias correction in exponentially weighted avg

↳ makes it more accurate

→ divide v_t by $1-\beta^t$ → $V_t = \frac{V_t}{(1-\beta^t)}$

Gradient Descent with momentum

lots of unnecessary computation, bcz it doesn't go in the dirctn of minima ←



☆ → we want to give it the right direction ~~what~~ but if we increase learning rate, it might over shoot

what we want :- ↕ slow learning in vertical dirctn
↔ fast learning in horizontal dirctn.

→ so we introduce momentum.

Algorithm: On iteration t:

compute dW, db on current mini batch

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

$V_{db} = \beta V_{db} + (1-\beta) db$ → provide accl^n in GD

provides direction in GD ← provide velocity in GD

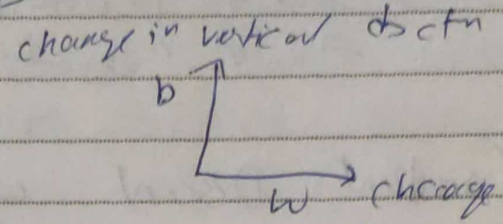
$W = W - \alpha V_{dW}$; $b = b - \alpha V_{db}$

now you'll 2 hyperparameters = α, β → usually 0.9 (good value)

[bias correction isn't usually considered a hyperparameter cz it doesn't matter that much]

RMSprop

↳ speeds up GD



On itration t :
compute dw, db on current mini-batch

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

$$w := w - \frac{\alpha\, dw}{\sqrt{S_{dw}}} \qquad b := b - \frac{\alpha\, db}{\sqrt{S_{db}}}$$

we hope → S_{dw} is relatively small so change in w (horizont is big)

→ S_{db} is v large so change in b (vertical dirctn) is small.

Adam Optimization

→ there have been alot optimization algo's in DL history

→ but most of them don't generalize in all case

→ RMSprop and GD with momentum are among v rare ones which generalize

Adam optimization → combines GD with momentum and RMSprop

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

On iteration t:

compute dw, db using current mini-batch.

momentum, β_1

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

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

RMSprop, β_2

bias correction
$$\begin{cases} V_{dw}^{corrected} = V_{dw}/(1-\beta_1)^t, & V_{db}^{corrected} = V_{db}/(1-\beta_1^t) \\ S_{dw}^{corrected} = S_{dw}/(1-\beta_2^t), & S_b^{corrected} = S_{db}/(1-\beta_2^t) \end{cases}$$

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

hyperparameter choice:

α: needs to be tuned

Suggested by authors of adam optimizat^n paper
$$\begin{cases} \beta_1 = 0.9 & (dw) \\ \beta_2 = 0.999 & (dw^2) \\ \varepsilon = 10^{-8} \end{cases}$$

why is it called adam?
→ Adaptive Moment Estimation.

Learning Rate decay
↳ slowly reduce learning-rate over time

- yK why we need this, you actually proposed it

1 epoch = 1 pass thru the data

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

decay_rate → another hyperparameter

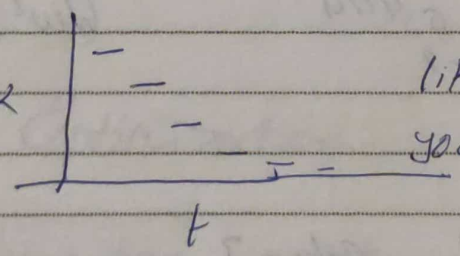
for $(\alpha_0) = 0.2$, decay rate = 7

epoch	α
1	0.1
2	0.067
3	0.05
4	0.04

other ways of calculating α

$\alpha = (0.95)^{\text{epochnum}} \cdot \alpha_0$ — exponential decay

$$\alpha = \frac{k}{\sqrt{\text{epochnum}}} \cdot \alpha_0 \quad \text{or} \quad \frac{k}{\sqrt{t}} \alpha_0$$



like a stair case
you keep dividing α by 2
in each iteration

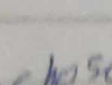
OR you can control α, manually !!

the problem of local optima

our intuitions about local and principal minima's in 2,3 dimensions don't really hold up in higher dimension function

we prefer to chose a convex f^n

but what if we'r a 20000 dimensional f^n?

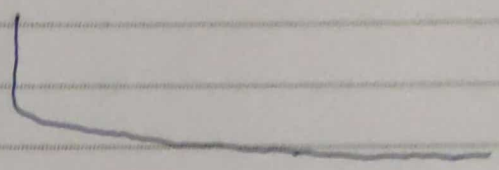
its crazy hard to get a function which looks like  this in 20000 dimensions

you get a lot of saddle points the

Ll point of ~~principal minimum~~ what you get in high dimensions saddle point

☆ another thing which slows GD is "plateau" region

the slope remains close to zero for a long period of time

something like 

→ we're unlikely to get stuck in local optima in higher dimension

→ Plateau's are a problem