Week-2 Mini-batch gradient descent

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

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

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

} Vectorized implementation

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

Back prop to compute gradients cost

$W^{[l]} = W^{[l]} - \alpha\, dW^{[l]}$, Similarly for b.

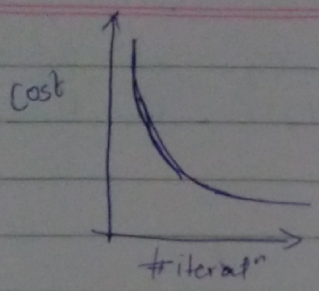
$\}$

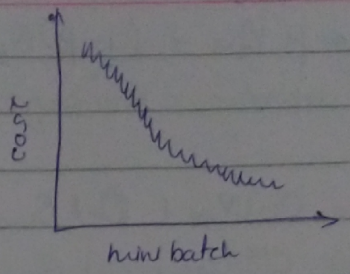
Here, 1 epoch will indicate 1 gradient descent step.

So, instead of waiting for 5 million training examples, we

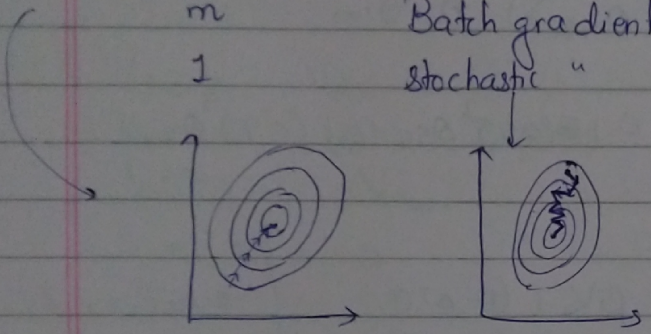
So, 1 epoch of training set = 1 gradient descent step

1 " " "mini" " 5 = 5000 " " " s





minibatch size	Name	
m	Batch gradient descent	$(x^{\{1\}}, y^{\{1\}}) = (X, Y)$
1	stochastic " "	$(x^{\{1\}}, y^{\{1\}}) = (x^{(1)}, y^{(1)}), \ldots$ every e.g. is a minibatch



In practice, stochastic gd will take more time & as each entry will need to be calculated. Thus, not keeping ~~obtaining~~ Vectorizat^n meaning.

minibatch size ≥ 1 & $< m$ $\Rightarrow$ fastest learning.

- How to choose mini-batch size
  - Small training set: Use batch gradient descent
    $m \leq 2000$
- Typical minibatch size: $2^6, 2^7, 2^8 \ldots$
- Make sure mini batch fits the CPU/GPU memory.

In practice, mini batch size is one hyper parameter to adjust for better performance

Exponentially weighted Average.

→ For cases when the data is moving. temp. v/s days. moving Average (trends) sh needs to be calculated.

→ to build more sophasticated optimizat^n algorithms

We take weighted avg.

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

approx. average over $\frac{1}{1-\beta}$ days

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

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

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

$V_{100} = 0.9 (0.9 V_{98} + (0.9 V_{97} + 0.1 \theta_{98})) + 0.1 \theta_{100}$

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

$V_0 = 0$

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

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

$V_3 = \beta V_2 + (1-\beta)\theta_3$

$V_0 := 0$

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

$V_\theta := \beta V_\theta + (1-\beta)\theta_2$

} vectorized

Bias Correction

To compute ~~the~~ weighted average efficiently in Instead of V_t, if we In the previous formula, the initial stage the graph will start ~~bit~~ with a bit smaller value

So, to correct it, we can use $\frac{V_t}{1-\beta^t}$ instead of V_t

for $t=2$,

$1-\beta^t = 1-(0.98)^2 = 0.0398$

$V_2 = \beta \left(\frac{V_t}{1-\beta^t}\right) + (1-\beta)\theta_2$

$V_1 = 0.98 V_0 + 0.02\theta_1$

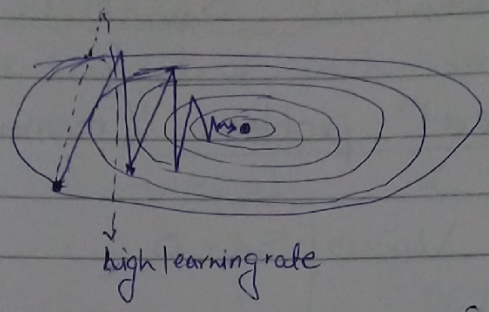
$= 0.0196\theta_1 + 0.02\theta_2$

$= 0.0396$

Gradient Descent with Momentum

As shown in figure, the gradient descent oscillates on the contour in order to find minima

Also, this oscillation prevents from using larger α, as the the gradient descent will go out of the contour



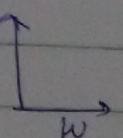
what we want:
↕ slow learning vertically
↔ fast " horizontally

$V_{dw} = \beta V_{dw} + (1-\beta) dw$, from $(V_\theta = \beta V_\theta + (1-\beta)\theta_t)$
$V_{db} = \beta V_{db} + (1-\beta) db$
$w := w - \alpha V_{dw}$ instead of dw, we take weighted average.
$b := b - \alpha V_{db}$ Thus, the oscillations will become more smooth

How, we have two hyperparameters α, β
Generally $\beta = 0.9$, nobody changes it, it is a robust value.
In practice, many people do not use $(1-\beta)$ term, but Andrew prefers to use it. It may happen that, formula with and w/o $1-\beta$ terms can need different α

RMS prop

Like momentum, it is also used to speed up gradient descent
for iteration t,
Compute dw, db on current mini-batch
$S_{dw} = \beta S_{dw} + (1-\beta) dw^2$
$S_{db} = \beta S_{db} + (1-\beta) db^2$
$w := w - \alpha \frac{dw}{\sqrt{S_{dw}}}$ & $b := b - \alpha \frac{db}{\sqrt{S_{db}}}$



Assume that S_{dw} is very small, that is change in w direction is small, thus, w can't update much. If S_{db} is very large, in that case $\sqrt{S_{db}}$ will be large, and hence more change can be expected.
Instead of w & b, their might be high dimensional vectors, that can be modified, based on avg change
Thus, Root mean square prop!

Hence, now we can have large learning rate to speed up gradient descent

Disadvantage, what if $S_{dw} = 0$??

Adam Optimizat^n problem (momentum + RMSprop)

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

On iteration t:

Compute dw, db using correct ~~implemented~~ mini-batch

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

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

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

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

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

choice of hyperparameters' values:

α : needs to be tuned

β_1 : 0.9

β_2 : 0.999

ε : 10^{-8}

Adam = Adaptive Moment Estimation

Learning rate Decay

For a fixed value of alpha, the convergence to minima wont take place.

Incase, if the value of alpha is gradually decreasing, the convergen won't be on minima still, but it will oscillate to a more tighter space

So, by reducing alpha we can take big steps initially and slowly take small steps

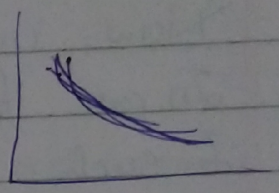
1 epoch = 1 pass through decay rate

$$\alpha = \frac{1}{1 + \text{decayrate} * \text{epoch-num}} \alpha_0$$

$\alpha = 0.2$

decay rate = 1

epoch	α
1	0.1
2	0.067
3	0.05
4	0.04



here, decay rate is another hyper-parameter

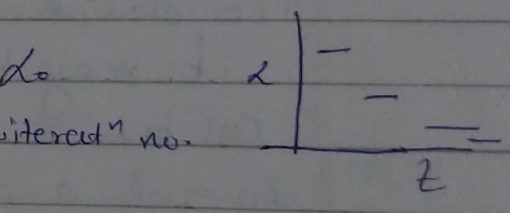
Exponential decay

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

discrete staircase decay

$$\alpha = \frac{k}{\sqrt{\text{epoch-num}}} * \alpha_0 \quad \text{or} \quad \frac{k}{\sqrt{t}} \cdot \alpha_0$$

$t \leftarrow$ iteration no.



manually decay after observation