Week6
Wednesday, May 13, 2020 8:53 PM

Optimization Algorithms :-

$X = [x^{(1)} \; x^{(2)} \; \dots \; x^{(m)}]$

$n_x \times m \rightarrow$ training examples
↳ features

$Y = [y^{(1)} \; \dots \dots \; y^{(m)}]$

$1 \times m$; if y = SM, then it's slow

$X = [\underbrace{x^{(1)} \; \dots \; x^{(1000)}}_{X^{\{1\}}} \; \underbrace{x^{(1001)} \; \dots \; x^{(2000)}}_{X^{\{2\}}} \; \dots \; \underbrace{x^{(m)}}_{X^{\{5000\}}}]$

Same for Y.

Notations :- $x^{(i)}$, $z^{(l)}$, $X^{\{t\}}, Y^{\{t\}}$
↳ $(n_x \times 1000)$ (1×1000)

for $t = 1$ to 5000
FP 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 implement^n

Compute cost $J^{\{t\}} = \frac{1}{1000} \sum_{i=1}^{l} L(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2 \times 1000} \sum_l \|w^{[l]}\|_F^2$

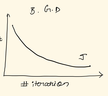
training examples $\{ 5000 \}$

↳ regular

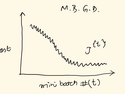
$w^{[l]} := w^{[l]} - \alpha . w^{[l]}$ .. same for b

"1 epoch" → passing through 1 training set.

B. G.D.



M.B. G.D.



* Mini-batch size :-

If mini-batch = m : Batch Gradient Descent $(X^{\{1\}}, Y^{\{1\}}, (X,Y))$

If mini-batch = 1 : Stochastic Gradient Descent → Every example is $(X^{\{1\}}, Y^{\{1\}}) = (x^{(1)}, y^{(1)})$ $(X^{\{2\}}, Y^{\{2\}}) = (x^{(2)}, y^{(2)})$
→ slow converging.

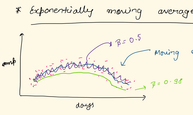
mini batch
→ slow converging.

In practice, somewhere b/w 1 & M → 64, 128, 256, 512 → preferred

→ Gradient descent learning for above three cases.

* Mini batch must fit in GPU/CPU memory.

* Exponentially moving averages :-



Moving Average

$v_0 = 0$
$v_1 = 0.9 v_0 + 0.1 \times \theta_1$ → temp of that day
$v_2 = 0.9 \times v_{(2-1)} + 0.1 \times \theta_2$

$v_t = \beta . v_{t-1} + (1-\beta) \theta_t$

$\beta = 0.9$: ≈ 10 days temperature
$\beta = 0.98$: ≈ 50 days
$\beta = 0.5$: ≈ 2 days.

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

$\frac{1}{1-0.98} = 50$

Bias Correction :-

$v_t = \beta v_{t-1} + (1-\beta) \theta_t$
$v_0 = 0$
$v_1 = \beta v_0 + 0.02 \theta_1$
$v_2 = 0.98 \times (0.02 \theta_1) + 0.02 \theta_2$
$= 0.0196 \theta_1 + 0.02 \theta_2$

Change $v_b = \frac{v_t}{1-\beta^t}$ } independent

at $t = 2$; $1-\beta^t = 1 - (0.98)^2$
$= 0.0396$

Now, $\frac{v_2}{0.0396} = \frac{0.0196 \theta_1 + 0.02 \theta_2}{0.0396}$

weighted average

Gradient Descent with Momentum :-

→ } RMS Prop.

Momentum :-

on iteration t :- → moving averages of derivatives
↳ smooth out the steps, reduces oscillations, speed up converging

Compute dw and db on current mini-batch

$v_{dw} = \beta v_{dw} + (1-\beta) . dw$ ← $v_0 = \beta v_0 + (1-\beta) \theta_t$
$v_{db} = \beta v_{db} + (1-\beta) db$

velocity of dw ↳ velocity ↳ acceleration (accelerate down)

$w = w - \alpha v_{dw}$, $b = b - \alpha v_{db}$

Preventing from blowing up. Hyperparameters → α, β Generally, $\beta = 0.9$ → weighted average value

* Generally bias correction is not used. $v_t = \frac{v_t}{1-\beta^t}$
we can remove $(1-\beta)$ as well from equation.
↳ v_{dw} is scaled by a factor of $\frac{1}{1-\beta}$
↳ α will change by a factor of $\frac{1}{1-\beta}$
} you may do this

RMSProp :-

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

$w = w - \alpha \frac{dw}{\sqrt{S_{dw}} + \varepsilon}$ } large → reduced faster than b
$b = b - \alpha . \frac{db}{\sqrt{S_{db}} + \varepsilon}$ } small → reduced slower
↳ slowing down updates in vertical direction
↳ dampen oscillations
↳ helps converge fast
$\varepsilon = 10^{-8}$ } to prevent blowing up.

↳ increase α,
↳ take big steps, reach minima fast with less oscillations
↳ $w \uparrow$ $b \downarrow$

Adam Optimization Algorithm :

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

On iteration t :

$V_{dw} = \beta_1 . V_{dw} + (1-\beta_1) dw$, $V_{db} = \beta_1 . V_{db} + (1-\beta_1) . db$ Momentum β_1
$S_{dw} = \beta_2 . V_{dw} + (1-\beta_2) dw^2$, $S_{db} = \beta_2 . S_{db} + (1-\beta_1) . db^2$ RMS Prop β_2

Now, bias correction,

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

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

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

Learning Rate decay :-

when converging, as the steps are in final stage, α starts to overshoot the min point and again comes back to minimum point. Hence, we need to reduce α to gain optimal results in minimum steps.

$\alpha = \frac{1}{1 + \text{decay rate} * \text{epoch\#}} * \alpha_0$
↳ initial LR

Epoch	α
1	0.1
2	0.67
3	0.5
4	0.4

Also

$\alpha = 0.95^{\text{epoch}} . \alpha_0$ → exponential decay.

or $\alpha = \frac{k}{\sqrt{\text{epoch\#}}} * \alpha_0$ or $\frac{k}{\sqrt{t}} * \alpha_0$

→ constant
→ mini-batch #