When gradient is small ... (optimization)

① Training Fails Because

- local minima — No way to go
- saddle point — escape

Which one ?

Taylor series Approximation

$L(\theta)$ around $\theta=\theta'$ can be approximated below → telling the properties of critical points

$$L(\theta) \approx L(\theta') + (\theta-\theta')^T g + \frac{1}{2}(\theta-\theta')^T H (\theta-\theta')$$

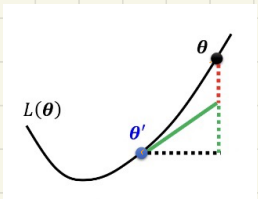
($g \to 0$ at critical point)

Gradient g is a vector: $g = \nabla L(\theta')$, $g_i = \frac{\partial L(\theta')}{\partial \theta_i}$

Hessian H is a matrix: $H_{ij} = \frac{\partial^2 L(\theta')}{\partial \theta_i \partial \theta_j}$

$$\Rightarrow L(\theta) \approx L(\theta') + \frac{1}{2}(\theta-\theta')^T H (\theta-\theta')$$

($v^T H v$)



a. $v^T H v > 0$ $\Rightarrow$ around θ': $L(\theta) > L(\theta')$ $\Rightarrow$ local minima

$\Rightarrow$ H is positive definite (正定矩阵) $\Rightarrow$ All eigen values are positive

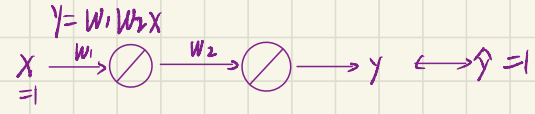
b. $v^T H v < 0$ $\Rightarrow$ Around θ': $L(\theta) < L(\theta')$ $\Rightarrow$ local maxima

$\Rightarrow$ H is negative definite $\Rightarrow$ All eigen values are negative

c. Sometimes $v^T H v > 0$, sometimes $v^T H v < 0$ $\Rightarrow$ saddle point

$\Rightarrow$ some eigen values are positive, some are negative

example:



$$L = (\hat{y} - w_1 w_2 x)^2 = (1 - w_1 w_2)^2$$

$$\frac{\partial L}{\partial w_1} = 2(1-w_1w_2)(-w_2) = 0$$

$$\frac{\partial L}{\partial w_2} = 2(1-w_1w_2)(-w_1) = 0$$

Critical point: $w_1 = 0, w_2 = 0$

$$H = \begin{bmatrix} 0 & -2 \\ -2 & 0 \end{bmatrix} \quad \lambda_1 = 2, \lambda_2 = -2$$

Saddle point

$$\frac{\partial^2 L}{\partial w_1^2} = 2(-w_2)(-w_2) = 0 \qquad \frac{\partial^2 L}{\partial w_1 \partial w_2} = -2 + 4w_1w_2 = -2$$

$$\frac{\partial^2 L}{\partial w_2 \partial w_1} = -2 + 4w_1w_2 = -2 \qquad \frac{\partial^2 L}{\partial w_2^2} = 2(-w_1)(-w_1) = 0$$

↓

我们需要 update θ !

$$L(\theta) \approx L(\theta') + \frac{1}{2}\underbrace{(\theta-\theta')^T}_{u^T} H \underbrace{(\theta-\theta')}_{u}$$

$\because$ u is an eigen vector of H

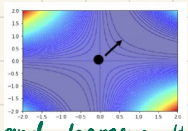
λ is the eigen value of u

$\Rightarrow u^T H u = u^T(\lambda u) = \lambda \|u\|^2$

当 $\lambda < 0$ 时 $L(\theta) < L(\theta')$ $\theta - \theta' = u$ 即 $\theta = \theta' + u$ Decrease L

example:

$\lambda_2 = -2$ $u = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ 任选一个 u

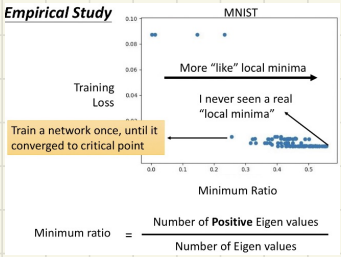


You can escape the saddle point and decrease the loss.

Saddle Point v.s. Local Minima

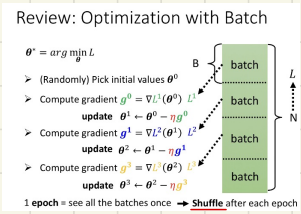
在一维空间中是 local minima，但现实中维数很大，对 Saddle point 会比 local minima 多。

Empirical study 实验研究



② Batch and Momentum

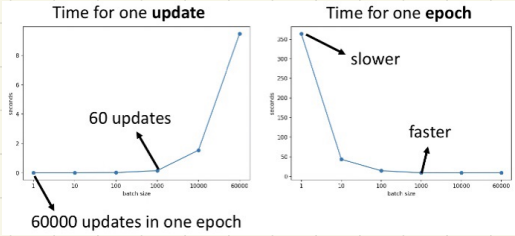
a. Batch



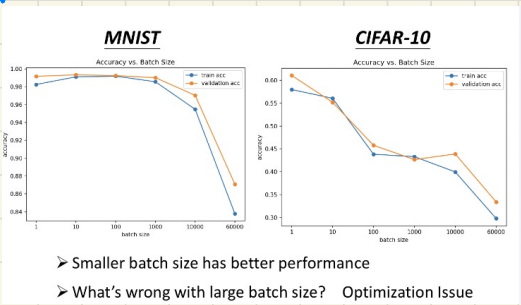
每 shuffle 之后 batch 不一样

Small Batch vs. Large Batch

- large batch size does not require longer time to compute gradient
- smaller batch requires longer time for one epoch

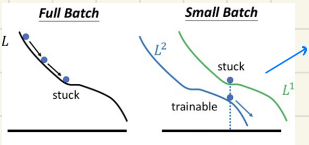


所以大的 batch 在时间上是不吃亏的.



- Smaller batch size has better performance
- What's wrong with large batch size? Optimization Issue

- Small batch size has better performance
- "Noisy" update is better for training



每次使用的 loss function 不一样

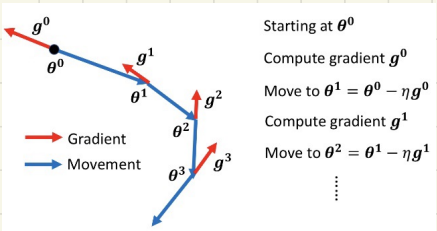
Name	Training Accuracy SB	Training Accuracy LB	Testing Accuracy SB	Testing Accuracy LB
F_1	$99.66\% \pm 0.05\%$	$99.92\% \pm 0.01\%$	$98.03\% \pm 0.07\%$	$97.81\% \pm 0.07\%$
F_2	$99.99\% \pm 0.03\%$	$98.35\% \pm 2.08\%$	$64.02\% \pm 0.2\%$	$59.45\% \pm 1.05\%$
C_1	$99.89\% \pm 0.02\%$	$99.66\% \pm 0.2\%$	$80.04\% \pm 0.12\%$	$77.26\% \pm 0.42\%$
C_2	$99.99\% \pm 0.04\%$	$99.99\% \pm 0.01\%$	$89.24\% \pm 0.12\%$	$87.26\% \pm 0.07\%$
C_3	$99.56\% \pm 0.44\%$	$99.88\% \pm 0.30\%$	$49.58\% \pm 0.39\%$	$46.45\% \pm 0.43\%$
C_4	$99.10\% \pm 1.23\%$	$99.57\% \pm 1.84\%$	$63.08\% \pm 0.5\%$	$57.81\% \pm 0.17\%$

"Noisy" update is better for generalization

	Small	Large
Speed for one update (no parallel)	Faster	Slower
Speed for one update (with parallel)	Same	Same (not too large)
Time for one epoch	Slower	Faster
Gradient	Noisy	Stable
Optimization	Better	Worse
Generalization	Better	Worse

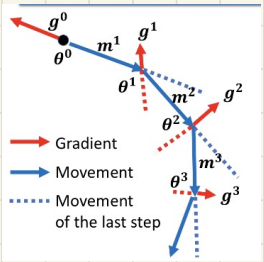
② momentum

Recall (Vanilla) Gradient Descent



Gradient Descent + Momentum

Movement: movement of last step minus gradient at present



① Starting at θ^0

Movement $m^0 = 0$

Compute gradient g^0

Movement $m^1 = \lambda m^0 - \eta g^0$

Move to $\theta^1 = \theta^0 + m^1$

② Compute gradient g^1

movement $m^2 = \lambda m^1 - \eta g^1$

move to $\theta^2 = \theta^1 + m^2$

$m^0 = 0$

$m^1 = -\eta g^0$

$m^2 = -\lambda\eta g^0 - \eta g^1$

Movement not just based on gradient, but previous movement.

图示