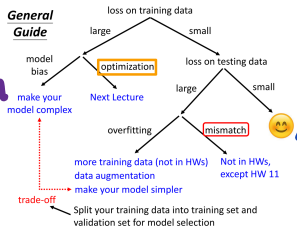
OPTIMIZATION

- Critical Point → Batch & Momentum
- Learning Rate → Optimizer



$L(\theta)$ around $\theta = \theta'$ can be approximated below

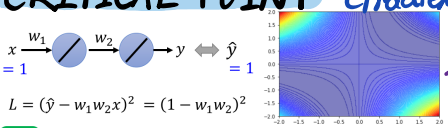
$$L(\theta) \approx L(\theta') + (\theta - \theta')^T g + \frac{1}{2}(\theta - \theta')^T H (\theta - \theta')$$

Let $v = \theta - \theta'$

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

CRITICAL POINT

Gradient = 0



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

$$g = \begin{bmatrix} \frac{\partial L}{\partial w_1} \\ \frac{\partial L}{\partial w_2} \end{bmatrix} = \begin{bmatrix} 2(1 - w_1 w_2)(-w_2) \\ 2(1 - w_1 w_2)(-w_1) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

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

$$H = \begin{bmatrix} \frac{\partial^2 L}{\partial w_1^2} & \frac{\partial^2 L}{\partial w_1 \partial w_2} \\ \frac{\partial^2 L}{\partial w_2 \partial w_1} & \frac{\partial^2 L}{\partial w_2^2} \end{bmatrix} = \begin{bmatrix} 2(-w_2)(-w_2) & -2 + 4 w_1 w_2 \\ -2 + 4 w_1 w_2 & 2(-w_1)(-w_1) \end{bmatrix} = \begin{bmatrix} 0 & -2 \\ -2 & 0 \end{bmatrix}$$

H may tell us parameter update direction!

Local Minima 局部最小值

Saddle Point 鞍点

↗ Hessian 用于判别是哪一种

$$H_{ij} = \frac{\partial^2}{\partial \theta_i \partial \theta_j} L(\theta')$$

Telling Properties of Critical Points

For all v

$v^T H v > 0$ ➡ Around θ': $L(\theta) > L(\theta')$ ➡ **Local minima**

$=$ H is positive definite $=$ All eigen values are positive.

For all v

$v^T H v < 0$ ➡ Around θ': $L(\theta) < L(\theta')$ ➡ **Local maxima**

$=$ H is negative definite $=$ All eigen values are negative.

Sometimes $v^T H v > 0$, sometimes $v^T H v < 0$ ➡ **Saddle point**

Some eigen values are positive, and some are negative.

通过计算 Hessian 矩阵 H 判断是 SaddlePoint or Local Minima/Maxima

(Eigen Vector & Value)

u λ

u is an eigen vector of H

λ is the eigen value of u

$\lambda < 0$

$$u^T H u = u^T (\lambda u) = \lambda \|u\|^2 < 0$$

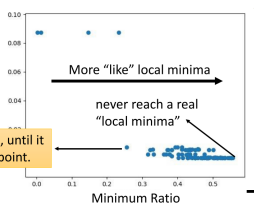
计算更新方向逃离鞍点。

低维的 Local Minima 在高维可能会变成 Saddle Point

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

$\theta - \theta' = u$ $\quad \theta = \theta' + u$ Decrease L

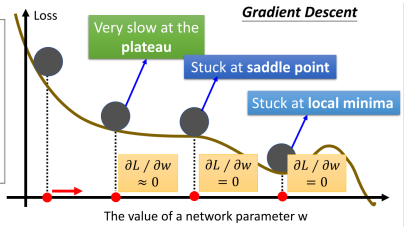
Empirical Study



Train a network once, until it converges to critical point.

Minimum ratio $=$ $\frac{\text{Number of Positive Eigen values}}{\text{Number of Eigen values}}$

Gradient Descent



→ 大部分都是 Saddle Point

Batch Small Batch Size to Escape

Consider 20 examples (N=20)

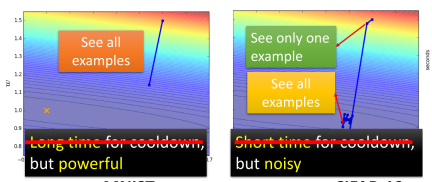
Batch size = N (Full Batch)

Update after seeing all the 20 examples

Batch size = 1

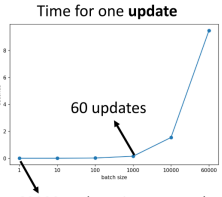
Update for each example

Update 20 times in an epoch

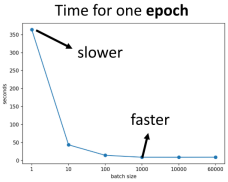


- Smaller batch requires longer time for one epoch (longer time for seeing all data once)

Time for one **update**



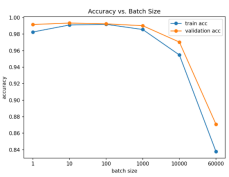
Time for one **epoch**



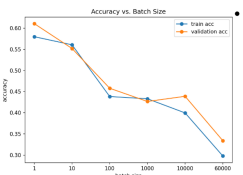
60000 updates in one epoch

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

MNIST



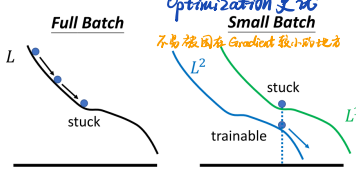
CIFAR-10



➢ Smaller batch size has better performance

➢ What's wrong with large batch size? Optimization Fails

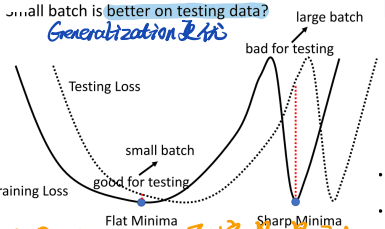
Optimization 更优



不易被困在 Gradient 很小的地方

Small batch is better on testing data?

Generalization 更优



大 Batch Size 更容易遇到 Sharp Minima 且小的更容易逃出 (Small有 Noise)

- 小的 Batch 不一定比大的快
  → 在一定范围内，Batch size 越大越快
- 所有 Batch 遍历一次 → 1 Epoch
- 每个 Epoch 结束后重新分配 Batch → Shuffle
- Small Batch size 准确率更高

	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 (new)
Gradient	Noisy	Stable
Optimization	Better (new)	Worse
Generalization	Better (new)	Worse

- Large Batch Optimization for Deep Learning: Training BERT in 76 minutes (https://arxiv.org/abs/1904.00962)
- Extremely Large Minibatch SGD: Training ResNet-50 on ImageNet in 15 Minutes (https://arxiv.org/abs/1711.04325)
- Stochastic Weight Averaging in Parallel: Large-Batch Training That Generalizes Well (https://arxiv.org/abs/2001.02312)
- Large Batch Training of Convolutional Networks (https://arxiv.org/abs/1708.03888)
- Accurate, large minibatch sgd: Training imagenet in 1 hour (https://arxiv.org/abs/1706.02677)

Momentum 惯性

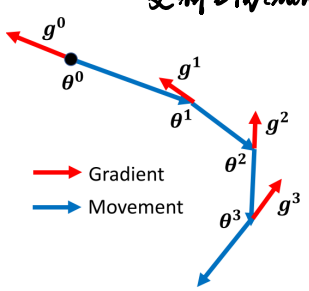
Movement = Negative of $\partial L / \partial w$

Movement = Negative of $\partial L / \partial w$ + Last Movement

(Vanilla) Gradient Descent

普通的

更新方向 (Movement) 和 Gradient 相反



Starting at θ^0

Compute gradient g^0

Move to $\theta^1 = \theta^0 - \eta g^0$

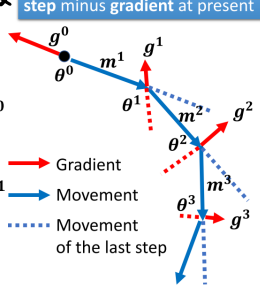
Compute gradient g^1

Move to $\theta^2 = \theta^1 - \eta g^1$

⋮

Gradient Descent + Momentum

Movement: **movement of last step** minus **gradient** at present



Starting at θ^0

Movement $m^0 = 0$

Compute gradient g^0

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

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

Compute gradient g^1

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

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

Movement not just based on gradient, but previous movement.

m^i is the weighted sum of all the previous gradient: $g^0, g^1, \ldots, g^{i-1}$

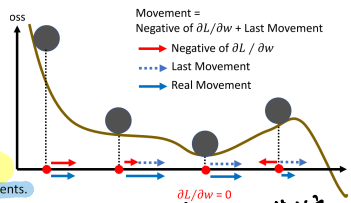
$m^0 = 0$

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

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

⋮

Gradient Descent + Momentum



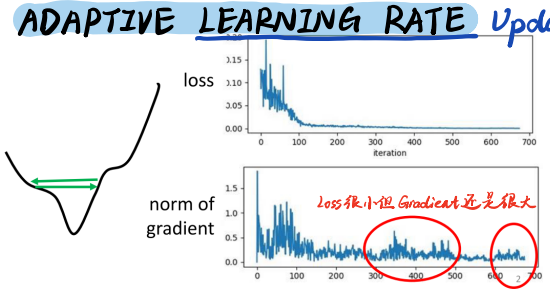
CONCLUSIONS

- Critical points have zero gradients.
- Critical points can be either saddle points or local minima.
  - Can be determined by the Hessian matrix.
  - It is possible to escape saddle points along the direction of eigenvectors of the Hessian matrix.
  - Local minima may be rare.
- Smaller batch size and momentum help escape critical points.

- Hessian 判定
- Small Batch & Momentum 解决

ADAPTIVE LEARNING RATE Update 的步伐

(Small Gradient)

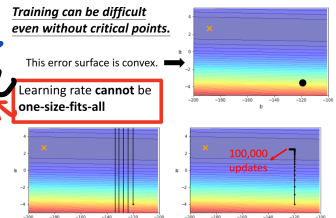


- Training stuck 很少是因为 Critical Point
  大部分在 Gradient 较大时已经下降到很小的 Loss
- 固定的 Learning rate 不能适用所有的训练情况
  Gradient 较小时，小 LR 无法推动训练进行
  Gradient 较大时，大 LR 会越过四处
  → Learning Rate 根据 Gradient 做调整

Training can be difficult even without critical points.

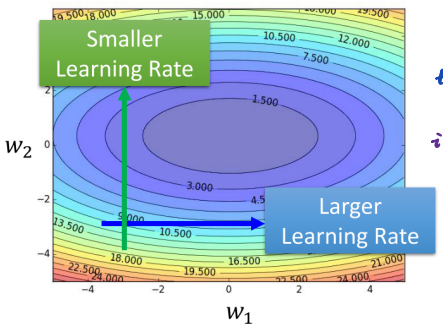
This error surface is convex.

Learning rate **cannot** be one-size-fits-all



→ Learning Rate 根据 Gradient 做调整：

Formulation for **one** parameter:



$$\theta_i^{t+1} \leftarrow \theta_i^t - \eta g_i^t$$

t th Iteration

$$g_i^t = \frac{\partial L}{\partial \theta_i}\Big|_{\theta=\theta^t}$$

i th parameter

$$\theta_i^{t+1} \leftarrow \theta_i^t - \frac{\eta}{\sigma_i^t} g_i^t$$

与 iteration 和 parameter 都有关

Parameter dependent

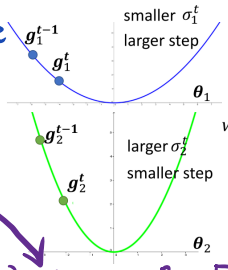
- **Adagrad**

Root Mean Square

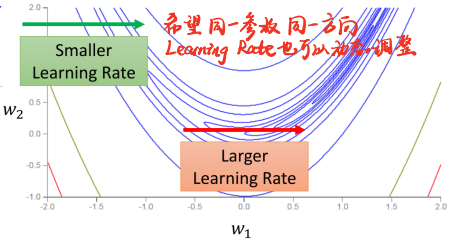
$$\theta_i^{t+1} \leftarrow \theta_i^t - \frac{\eta}{\sigma_i^t} g_i^t$$

$$\sigma_i^t = \sqrt{\frac{1}{t+1}\sum_{i=0}^{t}(g_i^t)^2}$$

Used in **Adagrad**



Error Surface can be very complex.



- **RMSProp**

RMSProp

初始化：$G_i^0 = \sqrt{(g_i^0)^2} = |g_i^0|$

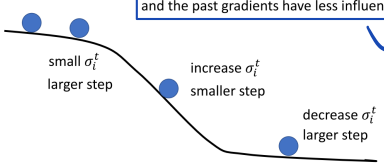
$g_i^1\ g_i^2 \cdots g_i^{t-1}$

$$\theta_i^{t+1} \leftarrow \theta_i^t - \frac{\eta}{\sigma_i^t} g_i^t \qquad \sigma_i^t = \sqrt{\alpha(\sigma_i^{t-1})^2 + (1-\alpha)(g_i^t)^2} \qquad 0 < \alpha < 1$$

The recent gradient has larger influence, and the past gradients have less influence.

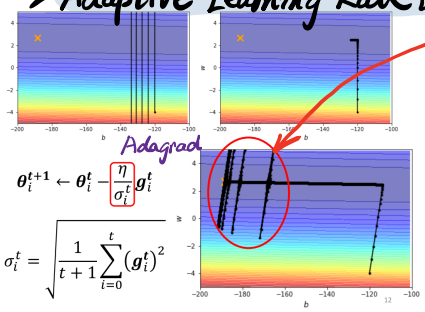
$\alpha \to 0$，g_i^t 相对于之前的 $(t-1)$ Gradient 更重要

$\alpha \to 1$，之前的 gradient 相对更重要



→ 通过调整 α 改变更新速度

→ Adaptive Learning Rate 出现的问题：



累积了很多较小的 g 和 σ（震荡后会回）

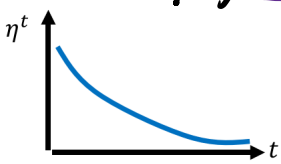
How to avoid?

→ Learning rate scheduling

使 Learning Rate 与时间有关

- **Learning Rate Decay**

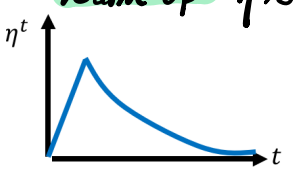
随时间的推移，Learning Rate 会越来越小



Learning Rate Decay 避免震荡

As the training goes, we are closer to the destination, so we reduce the learning rate.

- **Warm Up** η 先增大后减小（RAdam）



Warm Up

Increase and then decrease?

At the beginning, the estimate of σ_i^t has large variance.

σ：某个方向的陡峭 or 平滑程度

学习率小但方差大

→ 探索更好的方向

- **Adam** RMSProp + Momentum

应用最广泛的 Optimizer

Algorithm 1: *Adam*, our proposed algorithm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation. g_t^2 indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise. With β_1^t and β_2^t we denote β_1 and β_2 to the power t.

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
$m_0 \leftarrow 0$ (Initialize 1st moment vector) → for momentum
$v_0 \leftarrow 0$ (Initialize 2nd moment vector) → for RMSprop
$t \leftarrow 0$ (Initialize timestep)
while θ_t not converged **do**
$t \leftarrow t + 1$
$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
$\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
$\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
$\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$ (Update parameters)
end while
return θ_t (Resulting parameters)

11

(Residual Network & Transformer)

(Vanilla) Gradient Descent

$$\theta_i^{t+1} \leftarrow \theta_i^t - \eta g_i^t$$

Various Improvements

$$\theta_i^{t+1} \leftarrow \theta_i^t - \frac{\eta^t}{\sigma_i^t} m_i^t$$

η^t: Learning rate scheduling

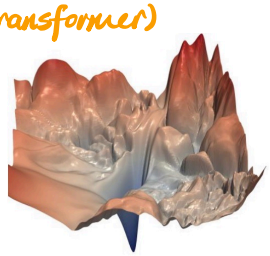
m_i^t: Momentum: weighted sum of the previous gradients → Consider direction

σ_i^t: root mean square of the gradients → only magnitude

17

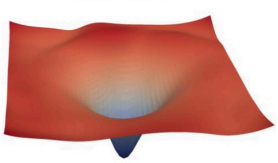
CONCLUSIONS：

- Adaptive Learning Rate
  - Adagrad
  - RMSProp
  - Adam
- 震荡
  - → Learning Rate Scheduling
    - Learning Rate Decay
    - Warm Up



Better optimization strategies: If the mountain won't move, build a road around it.

Next time



Can we change the error surface? Directly move the mountain!

从崎岖的山坡中找最优路径

→ 将崎岖的山坡变平滑

BATCH NORMALIZATION

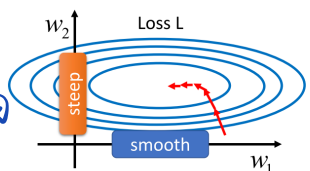
Input Feature $(x_1, x_2 \cdots)$ 每个 Dimension 的值差别很大

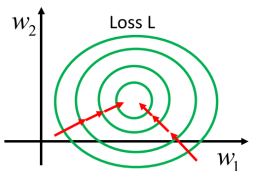
→ 产生椭圆形的 Error Surface 不同方向斜率坡度差异较大

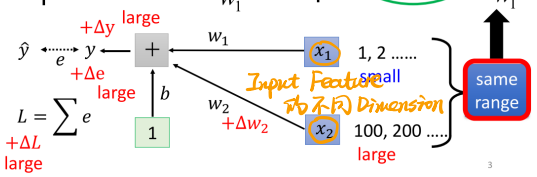
→ Feature Normalization 将不同维度的 Feature 规范到相同范围

→ 能够加速收敛

Changing Landscape 椭圆→圆



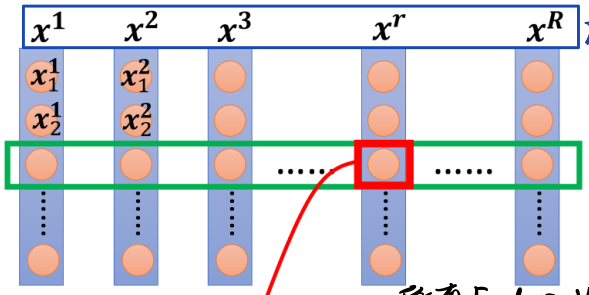




Feature Normalization

(不同的 Feature Vector)

- 取不同训练 Data 的同一 Dimension 的 Mean 和 Variance
- 上标不同 → 不同笔训练 Data（不同样本）
  下标不同 → Feature Vector 的 Dimension 不同



所有训练Data的 Feature

For each Vector dimension i:

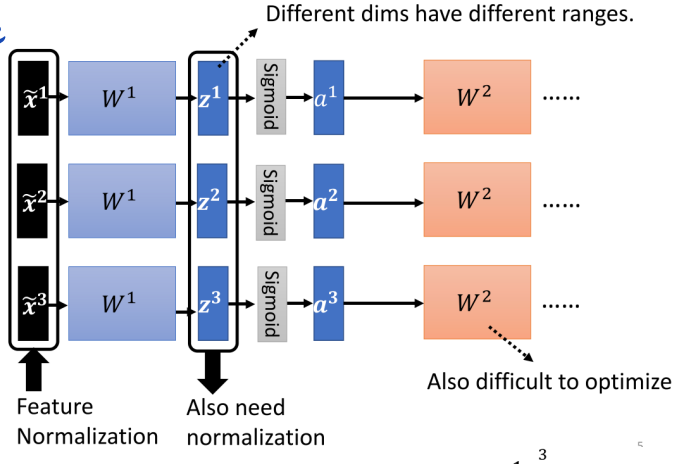
mean: m_i

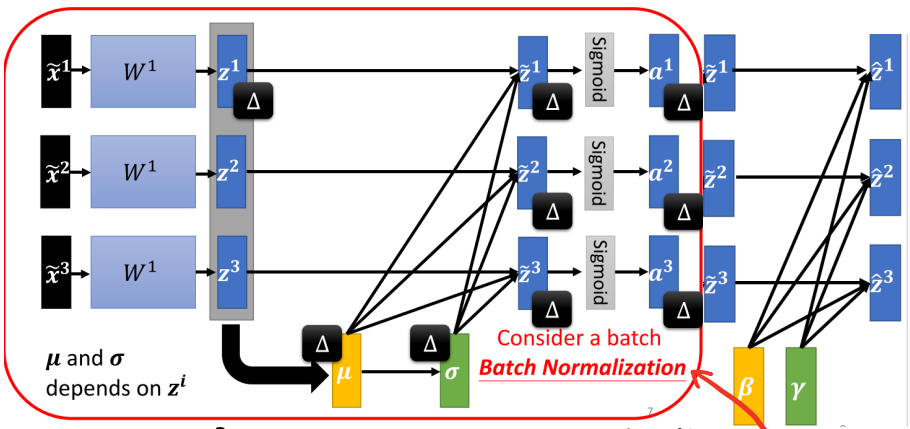
standard deviation: σ_i

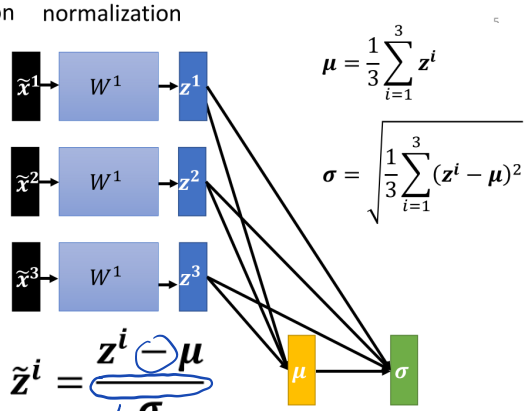
$$\tilde{x}_i^r \leftarrow \frac{x_i^r - m_i}{\sigma_i}$$

所有 Feature Vector 的不同 Dimension 都在 0 上下

The means of all dims are 0, and the variances are all 1







$$\mu = \frac{1}{3}\sum_{i=1}^{3} z^i$$

$$\sigma = \sqrt{\frac{1}{3}\sum_{i=1}^{3}(z^i - \mu)^2}$$

一个样本的变化会对其他样本的结果产生影响

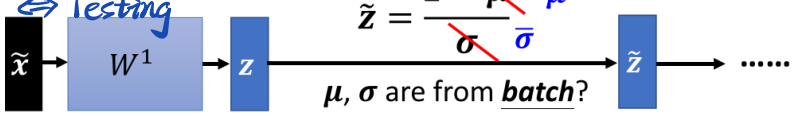
→ A Large Network 一个Network一次处理一个样本

Large Network 同时处理所有样本

只处理一个Batch 容量不够

Inference 测试阶段

⇔ Testing



μ, σ are from ***batch***?

Testing时不能收集到足够的 Batch

We do not always have ***batch*** at testing stage.

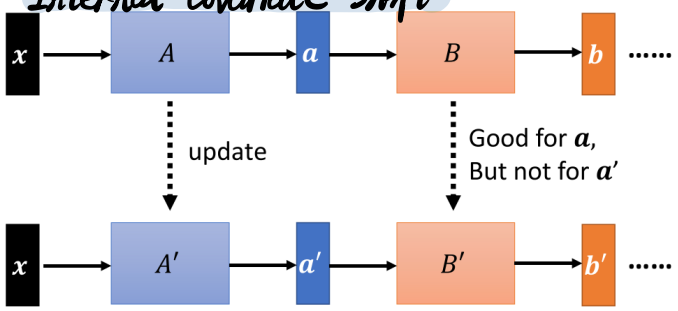
Computing the moving average of μ and σ of the batches during training. 训练时计算 Moving Average ($\bar{\mu}$, $\bar{\sigma}$)

$$\mu^1 \quad \mu^2 \quad \mu^3 \quad \ldots\ldots \quad \mu^t$$

$$\bar{\mu} \leftarrow p\bar{\mu} + (1-p)\mu^t$$

Why Batch Normalization?

- Internal Covariate Shift



Batch normalization make a and a' have similar statistics.

Experimental results do not support the above idea.

How Does Batch Normalization Help Optimization?

https://arxiv.org/abs/1805.11604

- Smoother Error Surface