#——

training stuck

#——

基于李宏毅's lecture, there are several scenarios training stuck:

1. Get stuck on

1) local min or max

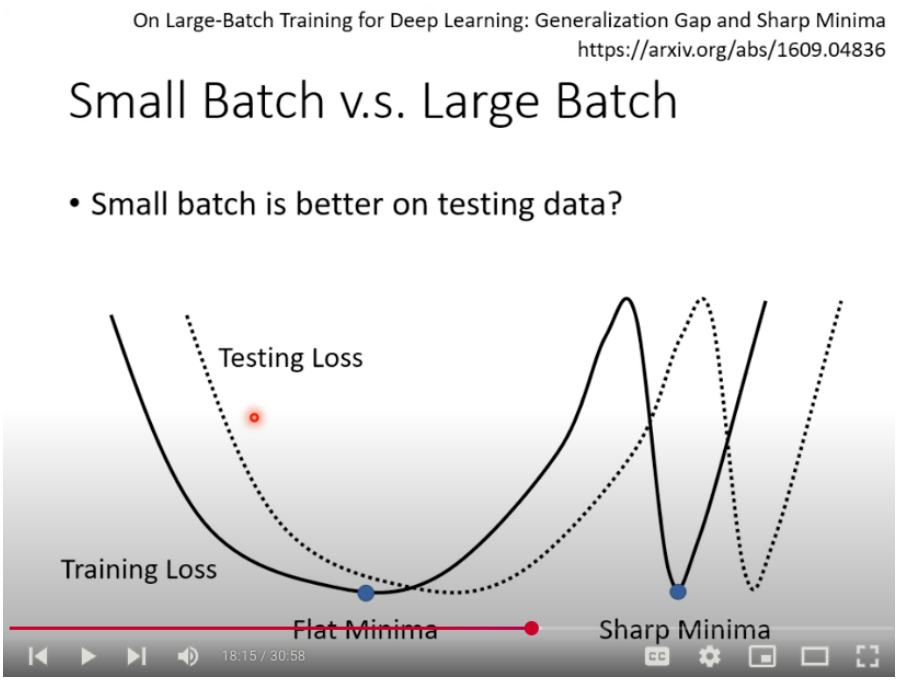
2) Saddle point

The above could use a) gradient, 2) hessian to detect

2. When get stuck on, deep valley local min can not jump out, there are 2 methods:

1) small batch size, to add noise. So it could jump out.

Also small batch size could introduce better test accuracy, which means less overfitting



On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima
https://arxiv.org/abs/1609.04836

Small Batch v.s. Large Batch

- Small batch is better on testing data?

SB = 256

LB = 0.1 x data set

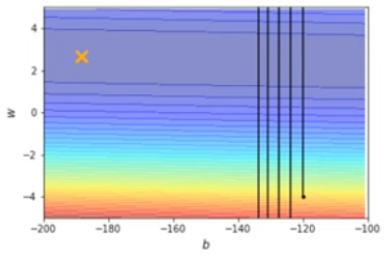
Name	Network Type	Data set
F_1	Fully Connected	MNIST (LeCun et al., 1998a)
F_2	Fully Connected	TIMIT (Garofolo et al., 1993)
C_1	(Shallow) Convolutional	CIFAR-10 (Krizhevsky & Hinton, 2009)
C_2	(Deep) Convolutional	CIFAR-10
C_3	(Shallow) Convolutional	CIFAR-100 (Krizhevsky & Hinton, 2009)
C_4	(Deep) Convolutional	CIFAR-100

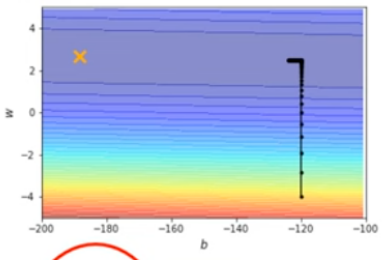
	Training Accuracy		Testing Accuracy	
Name	SB	LB	SB	LB
F_1	$99.66\% \pm 0.05\%$	$99.92\% \pm 0.01\%$	$98.03\% \pm 0.07\%$	$97.81\% \pm 0.07\%$
F_2	$99.99\% \pm 0.03\%$	$98.35\% \pm 2.08\%$	$64.02\% \pm 0.2\%$	$59.45\% \pm 1.05\%$
C_1	$99.89\% \pm 0.02\%$	$99.66\% \pm 0.2\%$	$80.04\% \pm 0.12\%$	$77.26\% \pm 0.42\%$
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	$99.10\% \pm 1.23\%$	$99.57\% \pm 1.84\%$	$63.08\% \pm 0.5\%$	$57.81\% \pm 0.17\%$

15:30 / 30:58

2) Adaptive learning rate

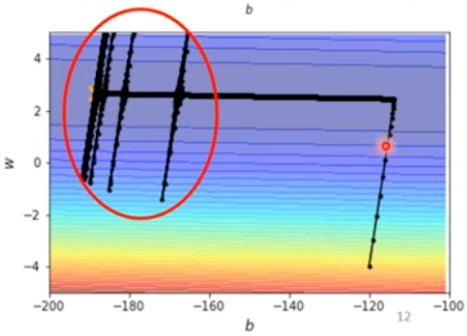
Without Adaptive Learning Rate





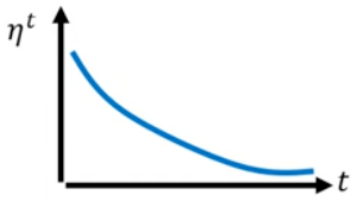
$$\theta_i^{t+1} \leftarrow \theta_i^t - \frac{\eta}{\sigma_i^t} g_i^t$$

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



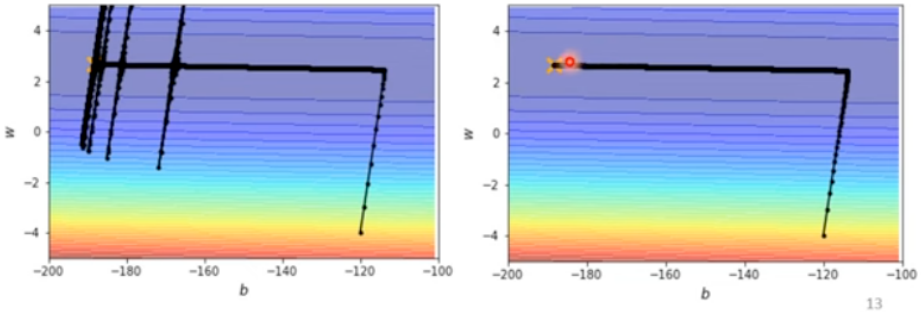
Learning Rate Scheduling

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



Learning Rate Decay

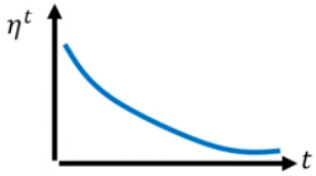
As the training goes, we are closer to the destination, so we reduce the learning rate.



13

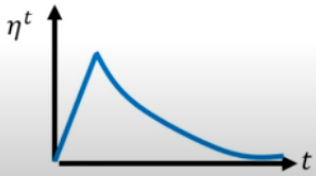
Learning Rate Scheduling

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



Learning Rate Decay

After the training goes, we are close to the destination, so we reduce the learning rate.

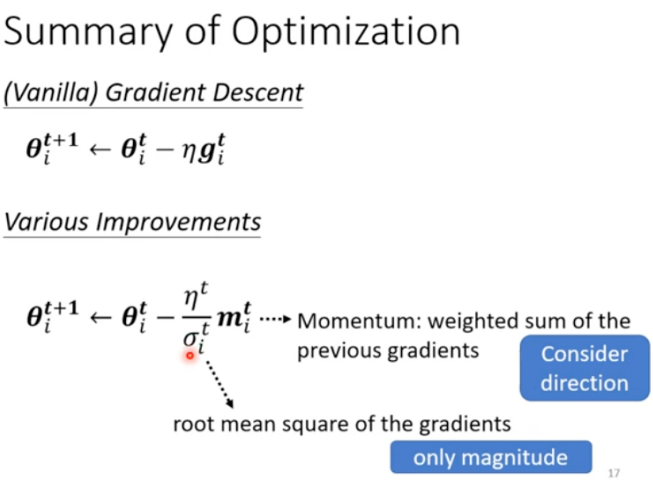


Warm Up

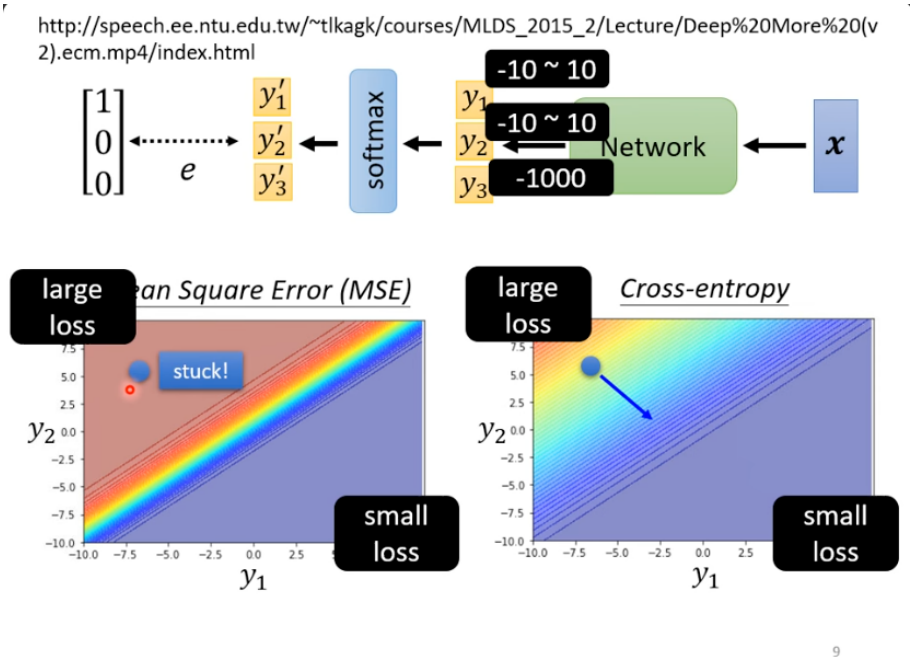
Increase and then decrease?

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

Please refer to **RAdam** https://arxiv.org/abs/1908.03265

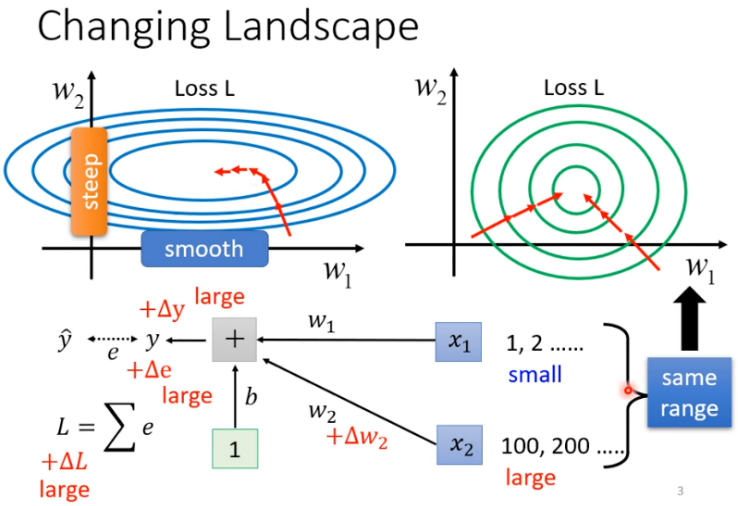


3) loss function



4) activation function, please check activationV0.py in this GitHub repository

5) normalization, e.g., bn

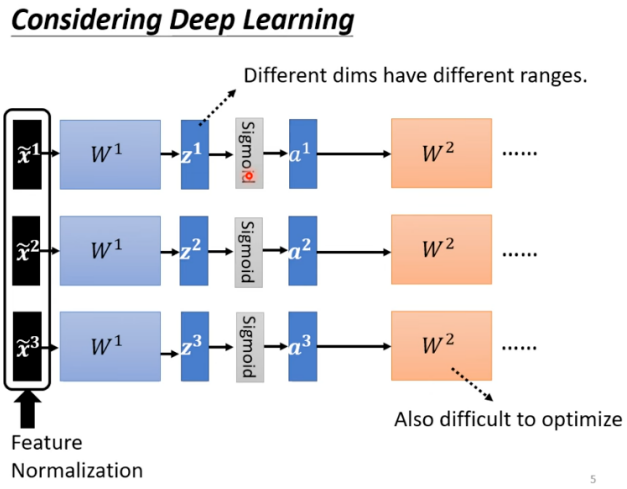


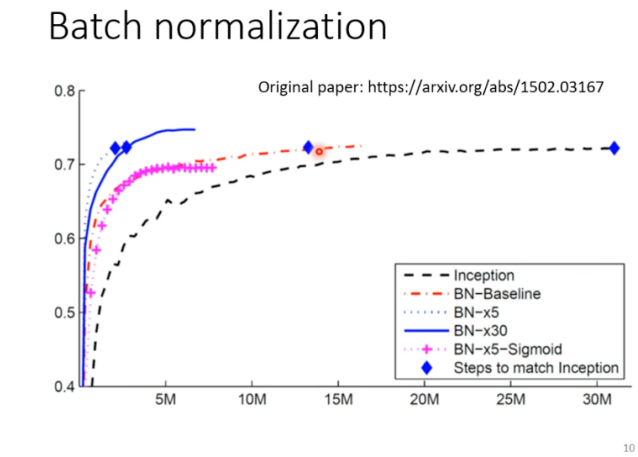
Question:

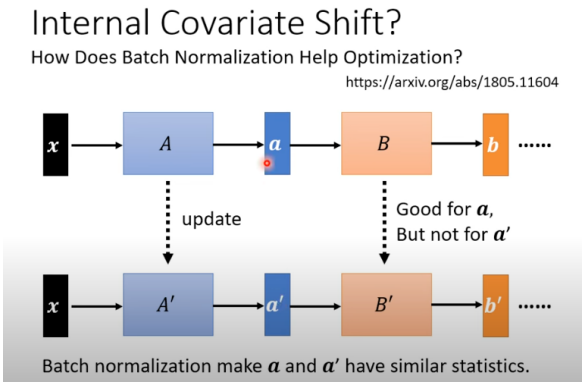
Whether to put normalization before or after activation function?

Ans:

Either way







Why is Internal Covariate Shift a Problem?

1. **Training Instability**: As the input distribution to a layer changes, the layer's parameters (e.g., weights) may no longer be optimal, requiring constant readjustment. This can lead to slower convergence or even training divergence.
2. **Vanishing/Exploding Gradients**: Shifting distributions can exacerbate gradient issues, especially in deep networks, making it harder for gradients to propagate effectively.
3. **Hyperparameter Sensitivity**: Models become more sensitive to learning rates and other hyperparameters, as they must compensate for the shifting distributions.

Details please check internal_covariate_shift0 in this repository.

#——

#llm training, why loss curve中有尖刺？请举例说明

#——

在训练大语言模型（LLM）时，损失曲线（loss curve）中出现尖刺（spikes）通常是由于训练过程中的某些不稳定性或异常情况导致的。这些尖刺表现为损失值在某些训练步骤突然显著增加，随后可能迅速下降或继续波动。以下是一些常见原因及举例说明：

1. 学习率过高

- **原因**：学习率设置过高可能导致模型参数更新幅度过大，跳过损失函数的局部最优解，进入高损失区域。
- **举例**：假设你在训练一个基于Transformer的LLM，使用Adam优化器，初始学习率设为1e-3。如果学习率过高，模型可能在某些batch上更新过大，导致损失突然从0.5激增到5.0，随后可能回落。
- **解决方法**：降低学习率（如从1e-3到1e-4），或使用学习率调度器（如线性衰减或余弦退火）平滑参数更新。

2. 数据质量问题

- **原因**：训练数据中可能存在噪声、错误标签或异常样本（如格式错误的文本、极长的序列等），导致模型在处理这些样本时损失激增。
- **举例**：在训练一个中文对话模型时，数据集中混杂了一段包含乱码或无关语言（如HTML代码）的样本。模型在处理这部分数据时，预测分布与目标分布差异巨大，损失值从0.3突增到10.0。
- **解决方法**：进行数据预处理，过滤掉异常样本或对输入序列长度进行截断。

3. 梯度爆炸

- **原因**：梯度在反向传播时变得过大，导致参数更新异常，进而引起损失尖刺。这在深层网络（如Transformer）中较为常见，尤其是在训练初期。

- **举例**：在训练一个多层Transformer模型时，某一层的梯度范数突然达到1e5量级，参数更新后损失从0.4暴增到20.0。
- **解决方法**：使用梯度裁剪（gradient clipping），将梯度范数限制在某个阈值（如1.0）。此外，梯度累积或混合精度训练也可缓解此问题。

4. 模型架构或初始化问题

- **原因**：模型初始化不当（如权重分布方差过大）或架构设计问题（如残差连接缺失）可能导致训练不稳定，引起损失尖刺。
- **举例**：在训练一个自定义的LLM时，如果未正确初始化注意力层权重，模型可能在某些batch上产生不稳定的输出，导致损失从0.6激增到15.0。
- **解决方法**：采用标准初始化方法（如Xavier或He初始化），或参考已验证的架构（如LLaMA）。

5. 优化器或超参数问题

- **原因**：优化器配置不当（如动量参数设置不合理）或batch size过小可能导致损失波动剧烈。
- **举例**：使用SGD优化器训练LLM时，batch size设为8，样本间方差较大，导致某些batch的损失从0.5突增到8.0。
- **解决方法**：增大batch size（如果硬件允许），或使用更稳定的优化器（如AdamW）。

6. 对抗性样本或分布偏移

- **原因**：训练数据分布在某些batch中发生显著变化（如从短文本切换到长文档），模型难以适应，导致损失尖刺。
- **举例**：在微调一个预训练LLM时，输入数据从短对话切换到长篇文章，模型未适应长序列的注意力机制，损失从0.2激增到7.0。
- **解决方法**：对数据进行分桶（按长度分组），或逐步增加序列长度进行训练。

可视化示例

假设你在训练一个中文LLM，损失曲线如下（假设为1000个训练步骤）：

- **正常情况**：损失从2.0平滑下降到0.3。
- **尖刺情况**：在第200步，损失从0.5突增到10.0，随后回落至0.4。这种尖刺可能由上述任一原因引起。

如果需要可视化损失曲线，可以提供具体数据，我可以为你生成一个Chart.js格式的折线图展示损失尖刺。

总结

损失曲线中的尖刺通常由学习率、数据质量、梯度爆炸、模型初始化、优化器配置或数据分布变化引起。解决方法包括调整超参数、清洗数据、使用梯度裁剪等。建议监控训练过程中的梯度范数、损失分布和样本质量，以快速定位问题。