推理和训练简介

周家豪

架构部，算法组

zhoujiahao@cambricom.com

2020 年 10 月 26 日

目录

机器学习

“对于任务 T 和度量 P，认为机器可以从经验 E 中学习是指，通过经验 E，它在任务 T 上由度量 P 衡量的性能有所提升。” - Mitchell (1997)

任务 T 任务一般为难以直接写出代码，例如：识别一张图片是不是树？(分类)

度量 P 度量一般与任务相关，以二分类为例，分类的准确率是任务完成程度的度量（评价指标）。

经验 E 经验一般指数据集，数据集一般分为训练集（验证集可选）和测试集。

目标通过使用训练集（验证集可选）提升任务在测试集上的度量。

常见机器学习任务：

1. 图像：分类 (ResNet50)，检测 (Faster RCNN)，分割 (Deeplab)，超分 (RCAN) 等
2. 视频：分类 (TSN)，跟踪 (PoseTrack) 等
3. 语音：识别 (DeepSpeech)，生成 (Wave RNN+Tacotron) 等
4. 文本：翻译 (Transformer) 等

评价指标

以分类任务为例，评价指标以 True|False（是否分对）Positive|Negative（正例 | 负例），分为：

1. True Position(TP)，分对的正例，即正分为正;
2. True Negative(TN)，分对的负例，即负分为负;
3. False Positive(FP)，分错的正例，即负分为正;
4. False Negative(FN)，分错的负例，即正分为负。

准确率 $Accuracy(Acc) = \frac{TP+TN}{TP+TN+FP+FN}$(ImageNet)

查准率 $Precision(P) = \frac{TP}{TP+FP}$

查全率 $Recall(R) = \frac{TP}{TP+FN}$

F1 $F1 = \frac{2}{\frac{1}{R}+\frac{1}{P}} = \frac{2TP}{2TP+FP+FN}$(Bert)

ROC 曲线（横坐标为 $FPR = \frac{FP}{FP+TN}$，纵坐标为 $TPR = \frac{TP}{TP+FN}$）

数据集

数据集一般划分为训练集、验证集和测试集。

1. 训练集，用于提升任务的度量，一般有数据和标签（需要数据预处理，Shuffle，可选数据增强）；
2. 验证集，用于判断训练是否完成，一般有数据和标签，但一般不参与直接序列（同训练集）；
3. 测试集，用于评估任务最后的指标（需要数据预处理）。

大部分任务拥有自己的公共数据集。
例如，ImageNet12 图像分类任务，有训练集（大概 100 万量级）、验证集（5 万），测试集（标签仅官方可见，一般用于比赛）。在训练任务中，可将验证集用做测试集，测试模型训练的效果; 若参加官方的评估，由于验证集有标签，也可以加入训练，提升模型训练的评价指标。

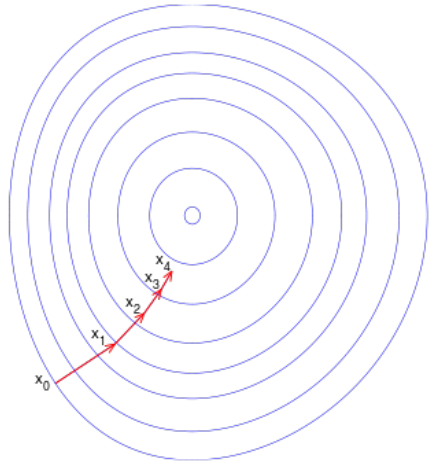
优化

深度学习一般使用优化的方法提升深度学习模型在任务上的评价指标，其基本流程如下：

1. 构建深度学习模型，例如：图像分类的 resnet50 模型、文本翻译的 transformer 模型等；
2. 选择学习损失函数，例如：图像分类的 CrossEntropy、目标检测的 L1/L2 Loss 等；
3. 选择学习优化器，例如：简单易用的 Adam、基础但广泛使用的 SGD 等；
4. 选择学习率下降策略，例如：StepLR、余弦模拟退火等；

梯度下降

梯度下降是求解可导函数的局部最小值的一阶迭代优化算法。基于函数 L 在当前 θ_i 的邻域内，往梯度的负方向走下降最快的假设，$\theta_{n+1} = \theta_n - \lambda_i \nabla L(\theta_n)$。(Optimizer，优化器)
当学习率 λ_n 足够小的时候，$L(\theta_0) \geq L(\theta_1) \geq L(\theta_2) \ldots$ 是一个递减的序列，最终能得到局部最小值[1]。



图：梯度下降

[1]当 L 是凸函数时，可求得全局最小值

损失函数一览

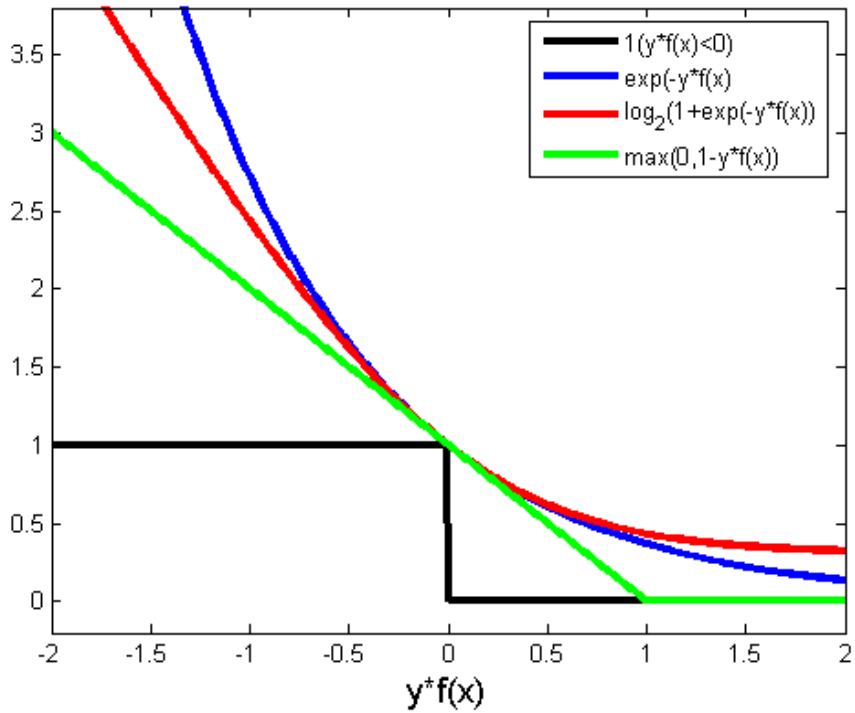


图: 常用损失函数

L1 损失

L1 损失，Mean Absolute Error(MAE)，即：

$$L(Y, f(X)) = \frac{1}{m}\sum_{i}^{m}|x_i - y_i|$$

基于欧式距离的绝对值，一般用于回归任务，例如检测。
平滑 L1 损失，即：

$$L(Y, f(X)) = \frac{1}{m}\sum_{i}^{m}\begin{cases}0.5(x_i - y_i)^2, if|x_i - y_i| < 1\\ |x_i - y_i| - 0.5, otherwise\end{cases}$$

L1 损失不仅可以用输出的损失，也可用于权重的正则化（一般用于产生稀疏权重）。

L2 损失

L2 损失，Mean Squared Error（MSE），即：

$$L(Y, f(X)) = \frac{1}{m}\sum_{i}^{m}(x_i - y_i)^2$$

基于欧式距离的绝对值，一般用于回归任务，例如检测。
L2 也可用于权重的正则化（一般可以用于防止过拟合）。

NLLLoss 负对数损失

NLLLoss, Negative Log Likelihood，用于分类任务，输入一般是类别的对数概率，即：

$$L(Y, P(Y|X)) = \sum_{i}^{m} -log(P(Y|X))$$

可以代入逻辑回归 (多分类 Softmax/二分类 Sigmoid) 的对数概率值，分别得到 CrossEntropy 和 BCELoss。
PoissonNLLLoss，目标是输入的泊松分布，即：

$$L(Y, X) = X - Y * log(X) + log(Y!)$$

CTC 损失

CTC 损失，基于最大化序列的概率，用于序列任务，例如：文字识别，语音识别，即：

$$L(l,x) = -log(p(l|x)), p(l|x) = \sum_{u=1}^{2|l|+1} \alpha(t,u)\beta(t,u)$$

最大化当前输入到最后标签序列 l 的概率值，其中每个节点的概率值通过前向后向算法得到。
对于长度为 T 的序列 $X = \{x_1, \ldots, x_T\}$
π 是输出序列，l 是标签序列，$\mathcal{F}$ 是输出路径到标签序列的映射关系。
设 y_k^t 为 t 时刻输出 k 的概率，则 $p(\pi|X) = \prod_{t=1}^{T} y_{\pi_t}^t$ 为基于输入 X 到的输出路径 π 的概率，
而 $p(l|X) = \sum_{\pi \in \mathcal{F}^{-1}(l)} p(\pi|X)$ 是所有可以映射到标签序列 l 的输出序列的概率之和。

CTC-前后向

其中 $\alpha(t,u), u \in [1, 2|l|+1]$ **为** t **时刻** u **节点的前向概率，则其公式如下**:

$$\alpha(t,u) = y^t_{l'_u} \sum_{i=f(u)}^{u} \alpha(t-1,i), f(u) =$$

$$\begin{cases} u-1, if\ l'_u = blank\ or\ l'_{u-2} = l'_u \\ u-2, otherwise \end{cases}$$

其中，$\alpha(1,1) = y^1_b, \alpha(1,2) = y^1_{l1}, \alpha(1,u) = 0, \forall u, u > 2$

其中 $\beta(t,u), u \in [1, 2|l|+1]$ **为** t **时刻** u **节点的后向概率，则其公式如下**:

$$\beta(t,u) = \sum_{i=u}^{g(u)} \beta(t+1,i) y^{t+1}_{l'_i}, g(u) =$$

$$\begin{cases} u+1, if\ l'_u = blank\ or\ l'_{u+2} = l'_u \\ u+2, otherwise \end{cases}$$

其中，

$$\beta(T, 2|l|+1) = 1, \beta(T, 2|l|) = 1, \beta(T,u) = 0, \forall u, u < 2|l| - 1$$

Ranking 损失

Ranking 损失，一般用于二分类，例如人脸识别。
MarginRankingLoss，即：

$$L(Y, f(X)) = max(0, -Y * (X1 - X2) + margin)$$

SoftMarginLoss，即：

$$L(Y, f(X)) = log(1 + exp(-YX))$$

TripletMarginLoss，（离正例更近，离负例更远）即：

$$L(A, B, C) = max(d(A, B) - d(A, C) + margin, 0)$$

优化器一览

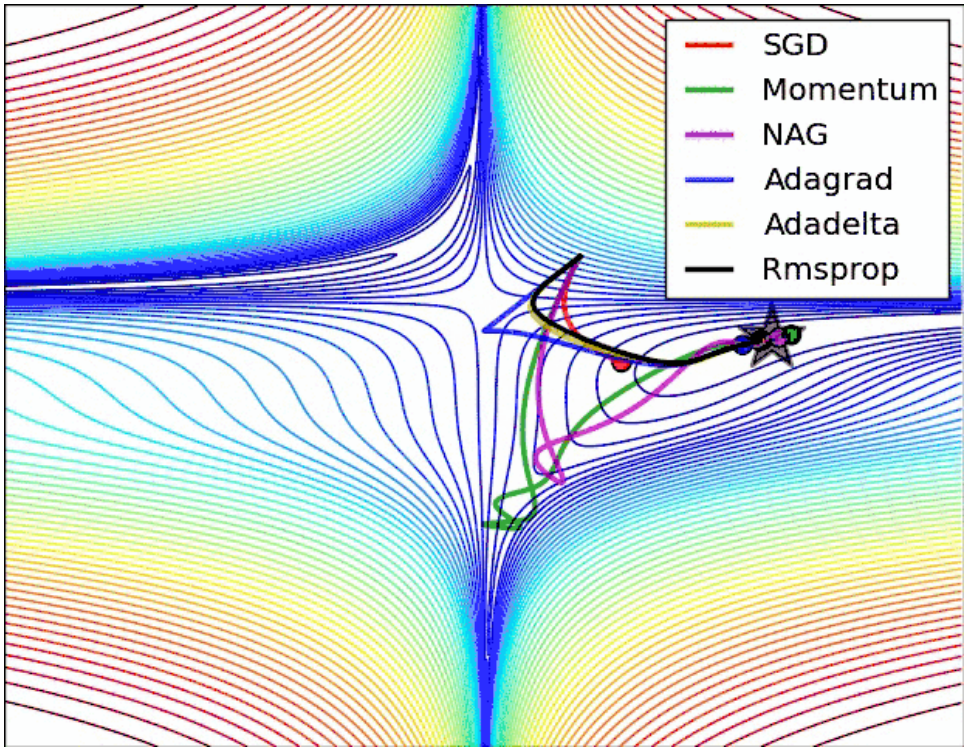


图: 常用优化器

Mini-batch Stochastic Gradient Descent

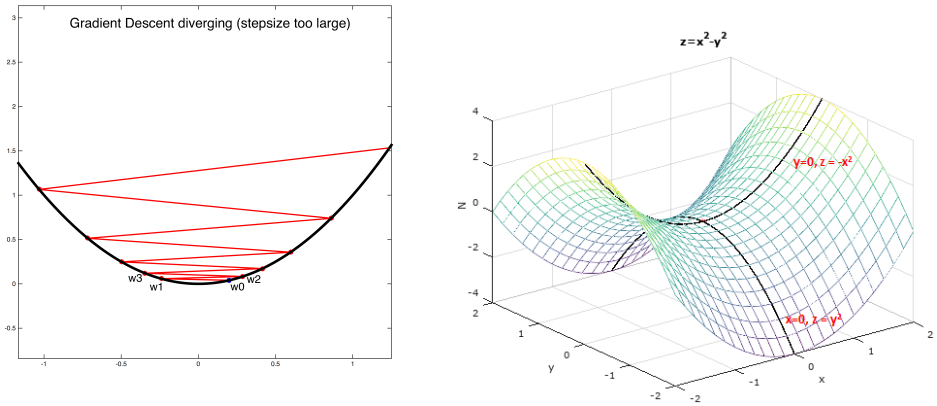
每批数据 (设 batchsize 为 m) 用于计算更新的梯度下降方向，

$\theta_{n+1} = \theta_n - \lambda_n \nabla_{\theta_n} J(\theta_n, x_{nm:(n+1)m}, y_{nm:(n+1)m})$

SGD 可以看作 BatchSize 为 1 的 MBGD。

SGD 的问题

1. 震荡，学习率过大不容易到局部最小值
2. 鞍点，当前值 θ 的梯度为 0，优化停止或者在该值附件震荡
3. 低频，对于更新不频繁的参数，增加单次更新的系数



图：鞍点

SGD with Momentum

目标：当梯度方向不变时更新加速，当梯度方向变化时更新减慢；

$$v_{n+1} = \gamma v_n + \lambda_n \nabla_{\theta_n} J(\theta_n)^2$$

$$\theta_{n+1} = \theta_n - v_{n+1}$$

Image 2: SGD without momentum

Image 3: SGD with momentum

图: SGD with momentum

最为常用的优化器，在合适的学习率下，能够保证收敛，一般 momentum 设为 0.9。

[2] γ 一般取 0.9

Nesterov Accelerated Gradient, NAG

目标：根据未来梯度方向进行调整；

$$v_{n+1} = \gamma v_n + \lambda_n \nabla_{\theta_n} J(\theta_n - \gamma v_n)^3$$

$$\theta_{n+1} = \theta_n - v_{n+1}$$

看作先按照更新（γv_n）后计算梯度，然后再做更新。

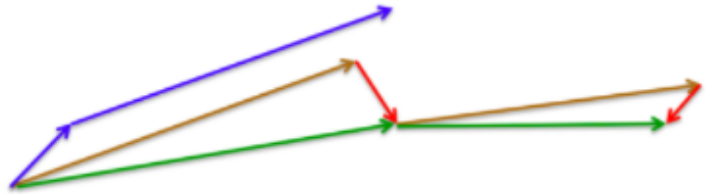


Image 4: Nesterov update (Source: G. Hinton's lecture 6c)

图：NAG

[3] γ 一般取 0.9

Adaptive gradient algorithm, Adagrad

目标：对较少更新的参数增加更新系数，对频繁更新的权重减小更新系数；

对第 i 个参数，其更新策略如下：

$$\theta_{n+1,i} = \theta_{n,i} - \frac{\lambda}{\sqrt{G_{n,i}+\epsilon}}$$

其中，$g_{n,i} = \nabla_{\theta_{n_i}} J(\theta_{n,i})$, $G_{n,i} = \sum_{k=1}^{n} g_{k,i}^2$

Adadelta|RMSprop

目标：调整系数不会因为累积不断减小；

$\theta_{n+1,i} = \theta_{n,i} - \frac{\lambda}{RMS[g_{n+1}]} g_{n,i}$

其中，$g_{n,i} = \nabla_{\theta_{n_i}} J(\theta_{n,i})$

$RMS[g_{n+1}] = \sqrt{E[g_{n+1}^2] + \epsilon}$[4]

$E[g_{n+1}^2] = \gamma E[g_n^2] + (1-\gamma) g_{n+1,i}^2$[5]

以上，Adadelta 和 RMSprop 相同 (实现上有先加 ϵ 和后加 ϵ 的区别)，同一时间提出。

Adadelta 将 $\lambda = RMS[\delta\theta_n]$ 代替，可以不需要设置学习率。

[4] ϵ 一般取 1e-8

[5] γ 一般取 0.9

Adaptive Moment Estimation, Adam

目标：自适应调整系数的同时，加入动量；

$\theta_{n+1,i} = \theta_{n,i} - \frac{\lambda}{\sqrt{\check{v}_{n+1,i}+\epsilon}} \check{m}_{n+1,i}$[6]

由于，m_0, v_0 被初始化为 0，需要进行修正。

其中，$m_{n+1,i} = \beta_1 m_{n,i} + (1-\beta_1) g_{n+1,i}$, $\check{m}_{n+1,i} = \frac{m_{n+1,i}}{1-\beta_1}$[7]

其中，$v_{n+1,i} = \beta_2 v_{n,i} + (1-\beta_2) g^2_{n+1,i}$, $\check{v}_{n+1,i} = \frac{v_{n+1,i}}{1-\beta_2}$[8]

其中，$g_{n,i} = \nabla_{\theta_{n,i}} J(\theta_{n,i})$

一般用于快速实验，能得到不错的结果，如果特征是稀疏的，建议使用 Adam 优化器。

---

[6] ϵ 一般取 1e-8

[7] β_1 一般取 0.9

[8] β_2 一般取 0.999

Adamax|AdamW

简化-Adamax, $\theta_{n+1,i} = \theta_{n,i} - \frac{\lambda}{v_{n+1,i}+\epsilon} \check{m}_{n+1,i}$[9]

由于，m_0, v_0 被初始化为 0，需要进行修正。

其中，$m_{n+1,i} = \beta_1 m_{n,i} + (1-\beta_1) g_{n+1,i}$, $\check{m}_{n+1,i} = \frac{m_{n+1,i}}{1-\beta_1}$[10]

其中，$v_{n+1,i} = max(\beta_2 v_{n,i}, |g_{n+1,i}|)$[11]

其中，$g_{n,i} = \nabla_{\theta_{n_i}} J(\theta_{n,i})$

改进-AdamW, L2 正则可以提升模型的泛化性，与 Adam 不兼容，AdamW 通过在更新中加入权重自身来引入：

$\theta_{n+1,i} = \theta_{n,i} - \lambda(\frac{1}{\sqrt{\check{v}_{n+1,i}}+\epsilon} \check{m}_{n+1,i} + \alpha\theta_{n,i})$[12]

---

[9] ϵ 一般取 1e-8

[10] β_1 一般取 0.9

[11] β_2 一般取 0.999

[12] α 一般取 0.005

衰减策略一览

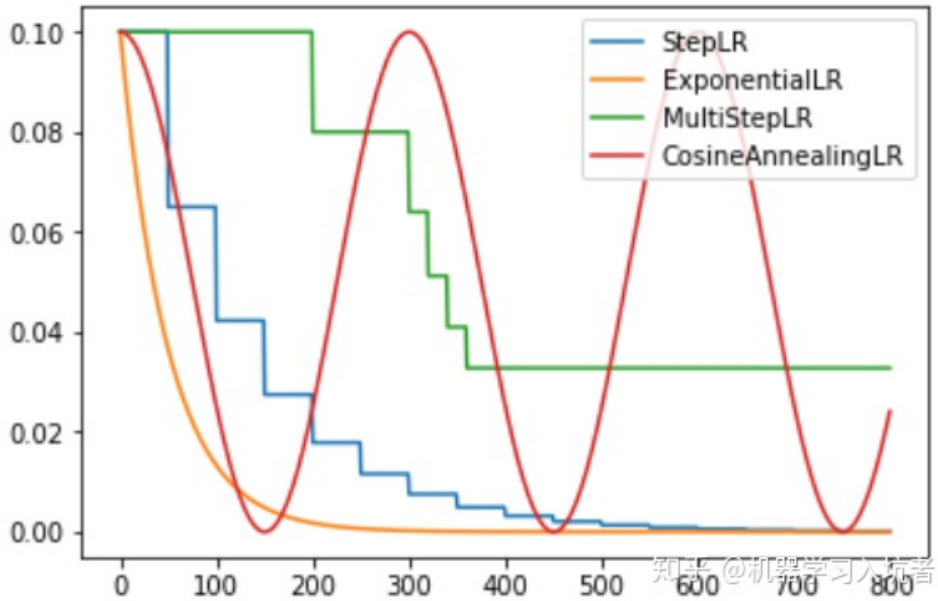


图: 常用衰减策略

ExponentialLR

通常以 Epoch(更为常用) 或者 Iter 为 n。

$\lambda_n = \lambda_0 \times \gamma^{n}$[13]

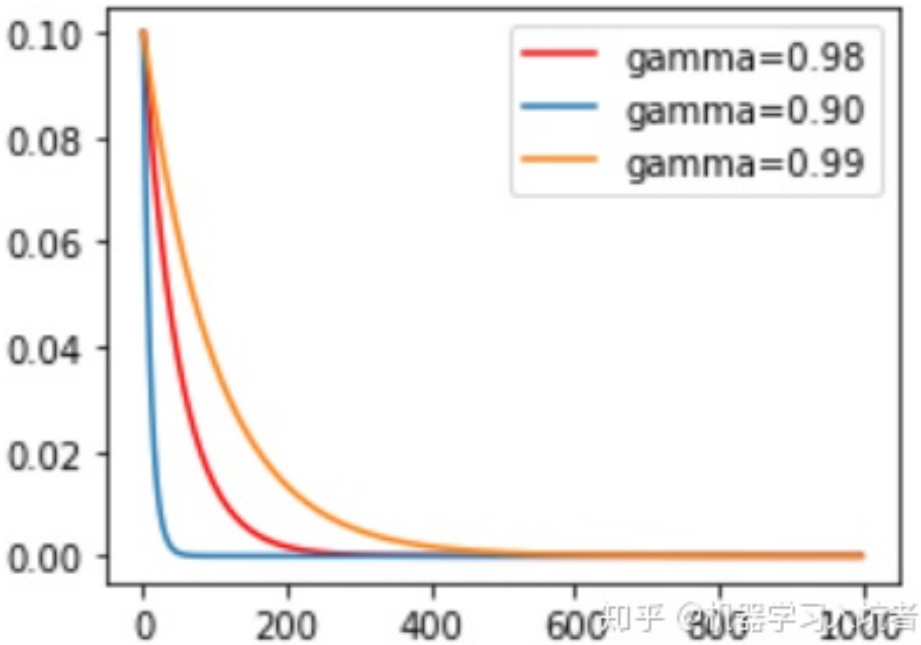


图: 指数衰减

[13] γ 可取 0.99

StepLR

通常以 Epoch(更为常用) 或者 Iter 为 n。

$\lambda_n = \lambda_0 \times \gamma^{(n//step)}$[14]

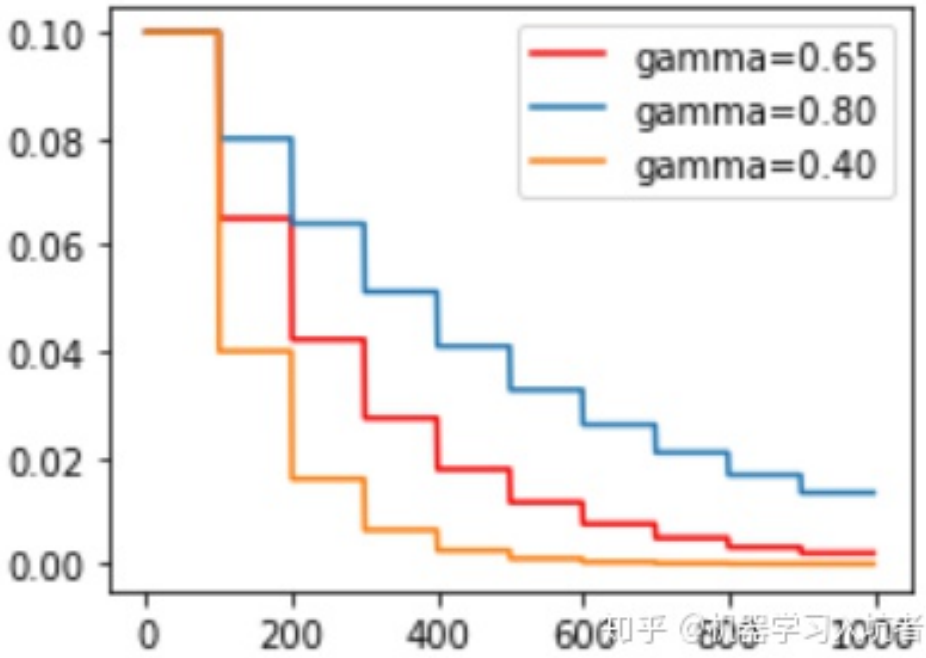


图: 固定步长衰减

[14] γ 可取 0.99

MultiStepLR

通常以 Epoch(更为常用) 或者 Iter 为 n, set 是跳变的 n 集合。

$\lambda_n = \lambda_0 \times \gamma^{(k)}, k = \sum n >= set_i$[15]

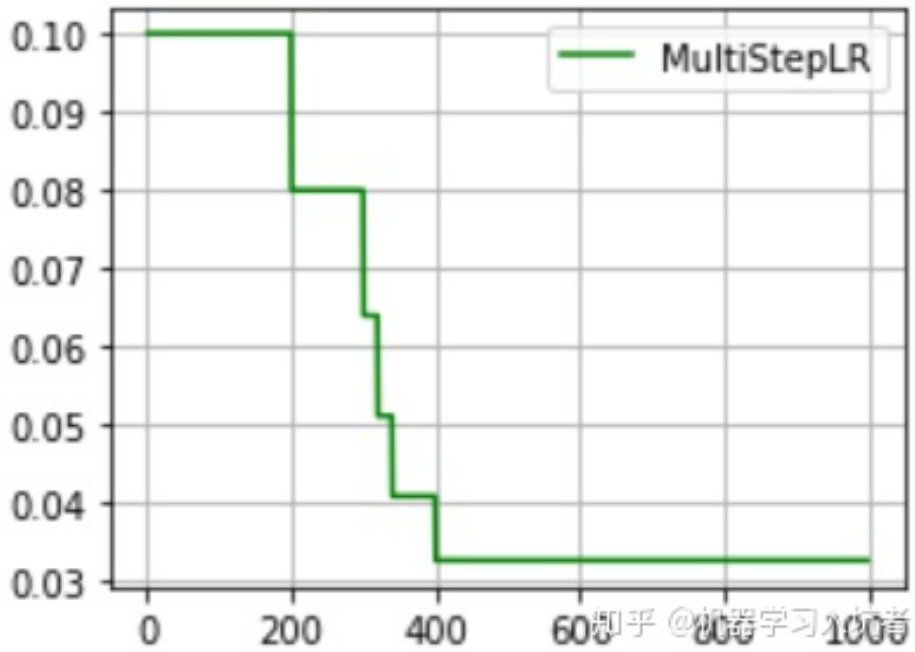


图: 多步长衰减

[15] γ 可取 0.99

CosineAnnealingLR

通常以 Iter 为 n。

$$\lambda_n = \lambda_{min} + \frac{1}{2}(\lambda_{max} - \lambda_{min})\left(1 + \cos\left(\frac{n}{n_{max}}\pi\right)\right), n \neq (2k+1)n_{max};$$

$$\lambda_{n+1} = \lambda_n + \frac{1}{2}(\lambda_{max} - \lambda_{min})\left(1 - \cos\left(\frac{1}{n_{max}}\pi\right)\right), n = (2k+1)n_{max}.$$

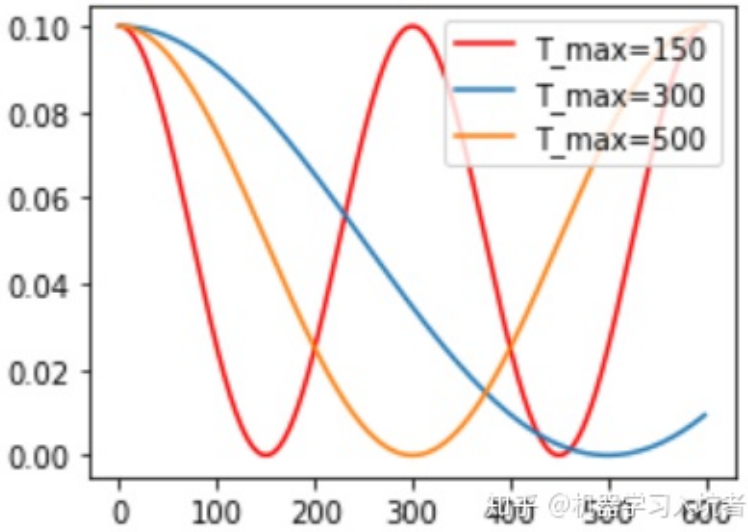


图: 余弦退火衰减

SqrtLR

通常以 Iter 为 n，用于 Transformer 训练。

$$\lambda_n = \begin{cases} \frac{n}{warmup} \times (lr - warmup\_lr) + warmup\_lr, if\ n <= warmup \\ \frac{\sqrt{warmup}}{\sqrt{n}} \times lr, if\ n > warmup \end{cases}$$

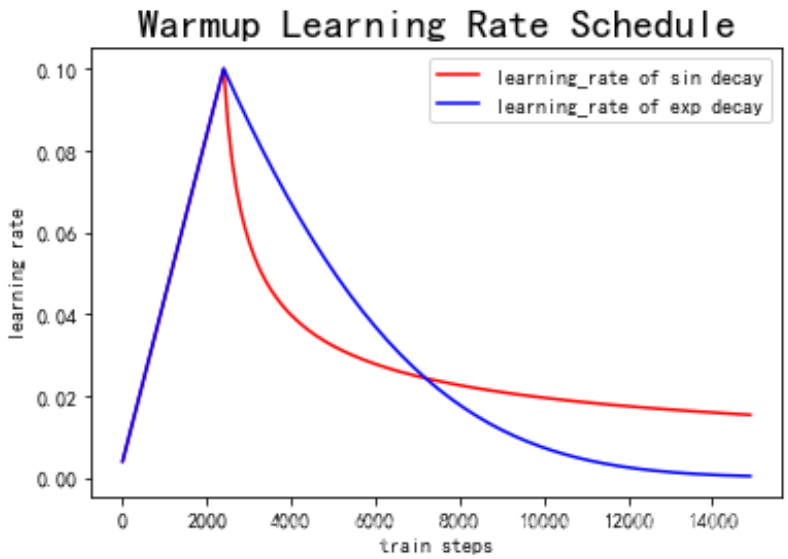


图: Warmup 衰减

常用超参数

数据集

1. batchsize，单卡与多卡
2. img size，图像大小
3. max len，最长语句长度

模型

1. dropout 系数

训练

1. learning rate，单卡与多卡
2. clip gradient，梯度裁剪
3. label smoothing，分类的标签平滑
4. weight decay，正则系数

推理

1. nms thred，NMS 阈值
2. max bounding box，box 的数量

课后作业

(Pytorch) 跑示例代码，并把优化器改为 Adam，看结果差异。
(Tesnsor) 跑示例代码的三种模式，看结果差异。

THANKS!