第十二讲 神经网络学习之小批量梯度下降法

2025年5月

概要

- M-P神经元模型
- 单隐层前馈神经网络
- 多层前馈神经网络
- 感知机
- 误差反向传播算法

- M-P神经元模型
  - $y = f(x_1, x_2, \cdots, x_n) = a(z), z = \sum_{i=1}^{n} w_i x_i + b$
  - 设$\mathbf{x} = (x_1, x_2, \cdots, x_n)^T$, $\mathbf{w} = (w_1, w_2, \cdots, w_n)^T$, 则 $y = f(\mathbf{x}) = a(\mathbf{w}^T\mathbf{x} + b)$
- 单隐层前馈神经网络: $\mathbf{y} = (y_1, \cdots, y_l)^T = f(\mathbf{x}; \theta)$

$$\mathbf{W}^{(1)} = \begin{bmatrix} w_{11}^{(1)} & \cdots & w_{1m}^{(1)} \\ \vdots & & \vdots \\ w_{n1}^{(1)} & \cdots & w_{nm}^{(1)} \end{bmatrix}, \mathbf{W}^{(2)} = \begin{bmatrix} w_{11}^{(2)} & \cdots & w_{1l}^{(2)} \\ \vdots & & \vdots \\ w_{m1}^{(2)} & \cdots & w_{ml}^{(2)} \end{bmatrix},$$

$\mathbf{b}^{(1)} = (b_1^{(1)}, b_2^{(1)}, \cdots, b_m^{(1)})^T$, $\mathbf{b}^{(2)} = (b_1^{(2)}, b_2^{(2)}, \cdots, b_l^{(2)})^T$, 则

- $\mathbf{h}^{(1)} = f^{(1)}(\mathbf{x}) = a(\mathbf{z}^{(1)}) = a({\mathbf{W}^{(1)}}^T\mathbf{x} + \mathbf{b}^{(1)})$
- $\mathbf{y} = f^{(2)}(\mathbf{h}^{(1)}) = g(\mathbf{z}^{(2)}) = g({\mathbf{W}^{(2)}}^T\mathbf{h}^{(1)} + \mathbf{b}^{(2)})$
- $\mathbf{y} = f(\mathbf{x}; \theta) = f^{(2)}(f^{(1)}(\mathbf{x}))$

- 多层前馈神经网络: $\mathbf{y} = (y_1, \cdots, y_l)^T = f(\mathbf{x}; \theta)$

$$\begin{cases} \mathbf{h}^{(1)} = f^{(1)}(\mathbf{x}) = a(\mathbf{z}^{(1)}) = a({\mathbf{W}^{(1)}}^T \mathbf{x} + \mathbf{b}^{(1)}) \\ \mathbf{h}^{(2)} = f^{(2)}(\mathbf{h}^{(1)}) = a(\mathbf{z}^{(2)}) = a({\mathbf{W}^{(2)}}^T \mathbf{h}^{(1)} + \mathbf{b}^{(2)}) \\ \vdots \\ \mathbf{h}^{(s-1)} = f^{(s-1)}(\mathbf{h}^{(s-2)}) = a(\mathbf{z}^{(s-1)}) = a({\mathbf{W}^{(s-1)}}^T \mathbf{h}^{(s-2)} + \mathbf{b}^{(s-1)}) \\ \mathbf{y} = f^{(s)}(\mathbf{h}^{(s-1)}) = g(\mathbf{z}^{(s)}) = g({\mathbf{W}^{(s)}}^T \mathbf{h}^{(s-1)} + \mathbf{b}^{(s)}) \end{cases}$$

- $\mathbf{y} = f(\mathbf{x}; \theta) = f^{(s)}(\cdots f^{(2)}(f^{(1)}(\mathbf{x}))\cdots)$.

- 隐层激活函数 $a(z)$:
  - $\sigma(z) = \frac{1}{1+e^{-z}}$, $\sigma'(z) = \sigma(z)(1-\sigma(z))$
  - $\tanh(z) = 2\sigma(2z) - 1 = \frac{e^z - e^{-z}}{e^z + e^{-z}}$, $\tanh'(z) = 1 - (tanh(z))^2$
  - $relu(z) = \max(0, z)$, $relu'(z) = \begin{cases} 1, & z > 0 \\ 0, & otherwise. \end{cases}$
  - 左饱和: $\lim_{z \to -\infty} a'(z) = 0$
  - 右饱和: $\lim_{z \to +\infty} a'(z) = 0$
  - 饱和: 左饱和且右饱和

输出层神经元的激活函数

- 回归: $y \in \mathbb{R}$
  - $y = f(\mathbf{x})$
  - $y = g(z) = z$
- 二类分类: $y \in \{0, 1\}$
  - $P(y = 1|\mathbf{x}) = f(\mathbf{x})$
  - $P(y = 1|\mathbf{x}) = g(z) = \sigma(z)$
- 多类分类: 独热向量$[y_1, y_2, \cdots, y_l]$
  - $[P(y_k = 1|\mathbf{x})] = f(\mathbf{x})$
  - $P(y_k = 1|\mathbf{x}) = g(z_k) = \frac{e^{z_k}}{\sum_{i=1}^{l} e^{z_i}}$
- 多标签分类: $y_k \in \{0, 1\},\ k = 1, 2, \ldots . l.$
  - $[P(y_k = 1|\mathbf{x})] = f(\mathbf{x})$
  - $P(y_k = 1|\mathbf{x}) = g(z_k) = \frac{1}{1+e^{-z_k}}$

- 能否对应于其他模型？
  - 回顾二类分类问题的感知机模型
    - $y = \begin{cases} +1, & sign(\mathbf{w}^T\mathbf{x} + b) \geq 0 \\ -1, & \text{其他} \end{cases}$
    - 无隐层的前馈网络(感知机)

$$y = \tanh(\mathbf{w}^{(1)^T}\mathbf{x} + b^1)$$

  - 非线性支持向量机与多层前馈神经网络
  - 凸优化与非凸优化
- 神经网络的表示能力
- 深度、复杂度与样本复杂度

概要

- 给定训练数据集$T = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$，学习$f(\mathbf{x}; \hat{\theta})$:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \left[\sum_{i=1}^{N} L(f(\mathbf{x}_i; \theta), y_i) + \lambda \Omega(f) \right]$$

其中

  - $L(\cdot)$是损失函数
  - $\Omega(\cdot)$是正则项
  - $\lambda \geq 0$是正则化系数

- 当$\lambda = 0$, $L(f(\mathbf{x}_i; \theta), y_i) = -\log P_\theta(y_i|\mathbf{x}_i)$时，

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \left[-\sum_{i=1}^{N} \log P_\theta(y_i|\mathbf{x}_i) \right]$$

其中$P_\theta(y_i|\mathbf{x}_i)$由神经网络$f(\mathbf{x}; \theta)$决定.

- 回归: 平方损失
  - $P_\theta(y|\mathbf{x}) \sim \mathrm{N}(f(\mathbf{x};\theta), \sigma^2)$
  - $\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \left[\frac{1}{2\sigma^2} \sum_{i=1}^{N} (y_i - f(\mathbf{x}_i;\theta))^2 - N \log \frac{1}{\sqrt{2\pi}\sigma} \right]$
  - 假设σ^2固定, 则$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \left[\frac{1}{2} \sum_{i=1}^{N} (y_i - f(\mathbf{x}_i;\theta))^2 \right]$
- 二类分类: 交叉熵损失
  - $P_\theta(y = 1|\mathbf{x}) = f(\mathbf{x};\theta)$
  - $\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \left\{ -\sum_{i=1}^{N} [y_i \log f(\mathbf{x}_i;\theta) + (1 - y_i) \log(1 - f(\mathbf{x}_i;\theta))] \right\}$
- 多类分类:交叉熵损失
  - $P_\theta(y_k = 1|\mathbf{x}) = f(\mathbf{x};\theta)$: 遵循categorical distribtuion
  - $\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \left\{ -\sum_{i=1}^{N} \left[\sum_{k=1}^{l} y_{ik} \log f(\mathbf{x}_i;\theta) \right] \right\}$,

    其中 $y_{ik} \in \{0, 1\}, \sum_{k}^{l} y_{ik} = 1,\ 1 \le k \le l, 1 \le i \le N.$

令

$$L(\theta) = \frac{1}{N}\sum_{i=1}^{N} L_i(\theta) = \frac{1}{N}\sum_{i=1}^{N} L(f(\mathbf{x}_i;\theta), y_i),$$

则

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}}\, L(\theta) = \underset{\theta}{\operatorname{argmin}}\, \frac{1}{N}\sum_{i=1}^{N} L_i(\theta)$$

- 非凸优化：大量的等价局部最优点
- 梯度下降法
  - 需要计算针对所有样本的梯度：$\frac{\partial L}{\partial \theta} = \frac{1}{N}\sum_{i=1}^{N}\frac{\partial L_i}{\partial \theta}$
  - 解可能仅是局部最优

小批量梯度下降法

输入: 网络架构$f(\mathbf{x};\theta)$, $T=\{(\mathbf{x}_i,y_i)\}_{i=1}^{N}$
输出: 参数向量$\hat{\theta}$
超参数: 学习率η, 小批量样本容量 n

1. 随机打乱样本次序, 将样本分成m组(小批量), 每组n个样本
2. 随机初始化参数向量θ
3. Do while(θ不收敛){
   For(i=1,2,$\cdots$, m){
   针对第i组的n个样本, 更新参数θ:

$$\theta \leftarrow \theta - \eta \cdot \frac{1}{n}\sum_{j=1}^{n}\frac{\partial L_j(\theta)}{\partial \theta}$$

   }
   }

- 小批量梯度下降法
  - 1 Epoch = $\frac{N}{n} \times$ Iteration
  - 可分布式并行计算, 提高学习效率
  - $n = N$: 批量梯度下降法
    - 累计误差反向传播算法
  - $n = 1$: 随机梯度下降法
    - 标准误差反向传播算法
    - 每次更新只针对一个样本.
    - 更新比较频繁, 可能会出现更新效果"抵消"现象.
    - 需要更多次数的迭代.
- 如何进一步改进?
  - 修正当前的梯度
  - 动态调整学习率
  - 更好的参数初始化方法
  - 数据预处理方法
  - 修改网络结构来得到更好的优化地形
  - 优化超参数

概要

回看参数更新

$$\theta \leftarrow \theta - \eta \cdot \frac{1}{n} \sum_{j=1}^{n} \frac{\partial L_j(\theta)}{\partial \theta}$$

进一步写成

$$\theta_t \leftarrow \theta_{t-1} - \eta g_t$$

$$\Delta\theta = \theta_t - \theta_{t-1} = -\eta g_t$$

- 批量大小n的选择
- 学习率调整
- 梯度估计修正

- 批量大小n的选择
  - n的大小不影响随机梯度的期望，但会影响其方差.
  - n越大，方差越小，引入的噪声越小，训练越稳定，可设置较大的学习率.
  - 线性缩放规则：当批量大小n比较小时，批量大小增加k倍，学习率也增加k倍.
  - 从迭代角度看， n越大，损失下降效果越明显， 下降曲线越平滑.
  - 从回合角度看，适当小的n会导致更快的收敛.

- 学习率η的调整：
  - η过大，可能导致梯度下降法不收敛；
  - η过小，则收敛速度太慢.
  - 常用的调整方法：
    - 学习率衰减方法
    - 学习率预热方法
    - 周期性学习率调整方法
    - 自适应学习率调整方法

- 学习率衰减(退火):
  - 在初始采用大的学习率来保证收敛速度, 在收敛到最优点附近时采用小的学习率来避免振荡.
  - 衰减按照每次迭代(也可以按每k次迭代/每个回合)进行.
  - 衰减率通常和迭代次数有关.
- 衰减(退火)方法: 设初始学习率η_0, 第t次迭代时的学习率η_t
  - 分段常数（阶梯）衰减: 每经过$T_1, T_2, \cdots, T_m$次迭代, 将学习率衰减为原来的$\beta_1, \beta_2, \cdots, \beta_m$倍.
  - 逆时衰减: $\eta_t = \frac{\eta_0}{1+\beta t}$, 其中$\beta$为衰减率.
  - 指数衰减: $\eta_t = \eta_0 \beta^t$, 其中$\beta < 1$为衰减率.
  - 自然指数衰减: $\eta_t = \eta_0 e^{-\beta t}$, 其中$\beta$为衰减率.
  - 余弦衰减: $\eta_t = \frac{1}{2}\eta_0(1 + \cos(\frac{t\pi}{T}))$, 其中$T$为总的迭代次数.

- 学习率预热：
  - 批量大小比较大时，通常需要设置比较大的学习率.
  - 训练初期，随机初始化的参数使得梯度也比较大，比较大的学习率会使得训练不稳定.
  - 预热：最初几轮训练先设置较小的学习率，等梯度下降到一定程度后再恢复初始学习率
    - 逐渐预热法：

$$\eta_t = \frac{t}{T'}\eta_0,$$

      其中T'为预热的迭代次数.
  - 预热过程结束后再考虑学习率衰减.

- 周期性学习率调整
  - 鞍点、 尖锐最小值与平坦最小值

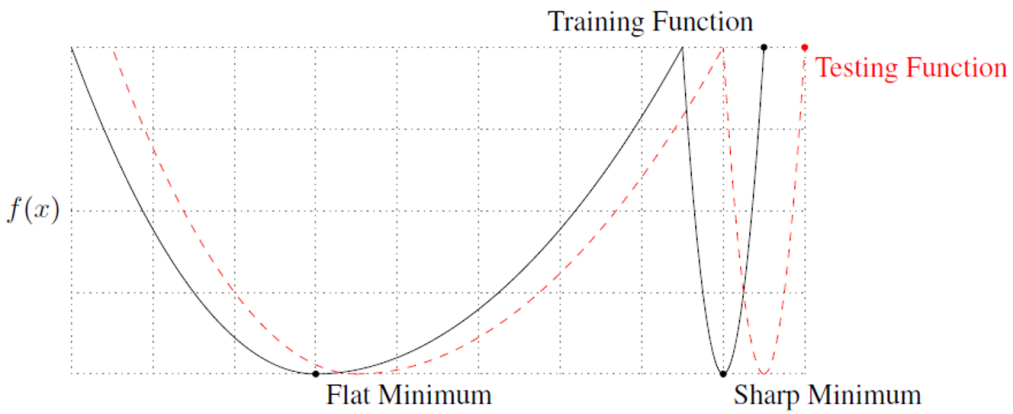

Figure 1: A Conceptual Sketch of Flat and Sharp Minima. The Y-axis indicates value of the loss function and the X-axis the variables (parameters)

Nitish Shirish Keskar et al., ON LARGE-BATCH TRAINING FOR DEEP LEARNING: GENERALIZATION GAP AND SHARP MINIMA, ICLR2017

Figure: 平坦最小值与尖锐最小值示意

- 周期性学习率调整： 帮助逃离鞍点或尖锐最小值的经验性方法
  - 当参数处于尖锐最小值附近时，增大学习率有助于参数逃离尖锐最小值；
  - 当参数处于平坦最小值附近时，增大学习率依然有可能处于该最小值的吸引域内；
  - 周期性增大学习率短期影响网络收敛的稳定性，长期来看有助于找到更好的局部最优解.
- 循环学习率
  - 三角循环学习率
- 带热重启的随机梯度下降

- 周期性学习率调整： 帮助逃离鞍点或尖锐最小值的经验性方法
- 循环学习率: 学习率在一个区间内周期性地增大和缩小
- 三角循环学习率
  - 每个循环周期前ΔT步学习率线性增大, 后ΔT步学习率线性衰减:

$$\eta_t = \eta_{\min}^m + (\eta_{\max}^m - \eta_{\min}^m)(\max(0, 1 - b)),$$

  其中
  - $m = \lfloor 1 + \frac{t}{2\Delta T} \rfloor$为第$t$次迭代所在的循环周期数,
  - $\eta_{\max}^m$ 和$\eta_{\min}^m$为当前周期中学习率的上下界(随m增大而递减).
  - $b = |\frac{t}{\Delta T} - 2m + 1| \in [0, 1]$.

- 周期性学习率调整： 帮助逃离鞍点或尖锐最小值的经验性方法
- 带热重启的随机梯度下降： 学习率每隔一定周期后重新初始化为某个预定值， 然后逐渐衰减， 参数在重启前的基础上继续优化.
  - 在第m重启周期内(从上次重启开始第T_m个回合后第m次重启)采用余弦衰减来降低学习率:

$$\eta_t = \eta_{\min}^m + \frac{1}{2}(\eta_{\max}^m - \eta_{\min}^m)(1 + \cos(\frac{T_{cur}}{T_m}\pi)),$$

其中

  - T_{cur}为上次重启之后的回合数,
  - $\eta_{\max}^m$ 和$\eta_{\min}^m$为当前周期中学习率的上下界, 可随着m增大而递减,
  - $T_m = \kappa T_{m-1}, \kappa \geq 1$.

- 自适应学习率调整方法： 为不同收敛速度的参数设置不同的学习率
  - AdaGrad 算法
  - RMSprop算法
  - AdaDelta算法

- AdaGrad算法
  - 回顾 $\Delta\theta_t = \theta_t - \theta_{t-1} = -\eta g_t$
  - 首先计算每个参数梯度平方的累积值

$$G_t = \sum_{\tau=1}^{t} g_\tau \odot g_\tau$$

这里$\odot$为Hadamard乘积.

  - 然后用这个累积值去修正学习率, 得到基于针对各参数的学习率的更新:

$$\Delta\theta_t = \theta_t - \theta_{t-1} = -\frac{\eta}{\sqrt{G_t + \epsilon}} \odot g_t$$

这里ϵ是为了保持数值运算的稳定性, 一般在 e^{-10} 和 e^{-7} 之间取值.

- 自适应调整, 但随着迭代次数的增加, 学习率都在减小.
- 在一定次数的迭代后没找到最优点时, 很难再继续找到.

- RMSprop算法
  - 希望在有些情况下避免AdaGrad算法学习率单调下降以致过于早衰的缺点.
  - 参数更新的形式仍为:

$$\Delta\theta_t = \theta_t - \theta_{t-1} = -\frac{\eta}{\sqrt{G_t + \epsilon}} \odot \mathrm{g}_t$$

  - 但G_t的计算改写成

$$G_t = \beta G_{t-1} + (1-\beta)\mathrm{g}_t \odot \mathrm{g}_t = (1-\beta)\sum_{\tau=1}^{t} \beta^{t-\tau}\mathrm{g}_\tau \odot \mathrm{g}_\tau$$

    其中β为衰减率, 一般设为0.9.
  - η为初始学习率, 一般设为0.001.

- AdaDelta算法
  - 在RMSprop算法的基础上, 进一步改进AdaGrad算法:
  - 引进参数更新差值$\Delta\theta$的平方的指数衰减移动平均

$$\begin{aligned}\Delta X_{t-1}^2 &= \beta_1 \Delta X_{t-2}^2 + (1-\beta_1)\Delta\theta_{t-1} \odot \Delta\theta_{t-1} \\ &= (1-\beta_1)\sum_{\tau=1}^{t-1} \beta_1^{t-1-\tau} \Delta\theta_\tau \odot \Delta\theta_\tau,\end{aligned}$$

    其中β_1为衰减率.
  - 参数如下更新

$$\Delta\theta_t = -\frac{\sqrt{\Delta X_{t-1}^2 + \epsilon}}{\sqrt{G_t + \epsilon}} \odot g_t$$

  - 以$\sqrt{\Delta X_{t-1}^2}$来代替初始学习率η, 在一定程度上平抑了学习率的波动.

- 对小批量梯度下降法中，如果每次选择的样本比较少，每次迭代的梯度具有一定的随机性.
- 如何缓解梯度随机性？
  - 增大批量大小
  - 用最近一段时间内的平均梯度来代替当前的梯度
- 动量法：
  - 利用负梯度的移动平均来更新参数：

$$\Delta\theta_t = \rho\Delta\theta_{t-1} - \eta g_t = -\eta\sum_{\tau=1}^{t}\rho^{t-\tau}g_\tau,$$

这里ρ为动量因子，通常设为0.9.

  - 最近一段时间内的梯度方向一致，参数更新幅度变大，加速，更快到最优点.
  - 最近一段时间内的梯度方向不一致，参数更新幅度变小，减速，增加稳定性.

- Nesterov加速梯度:
  - 动量法的参数更新:

$$\Delta\theta_t = \rho\Delta\theta_{t-1} - \eta g_t(\theta_{t-1}).$$

  - 重新分为两步更新:

$$\hat{\theta} = \theta_{t-1} + \rho\Delta\theta_{t-1}$$

$$\theta_t = \hat{\theta} - \eta g_t(\hat{\theta})$$

  - 合并以后的更新方向为

$$\Delta\theta_t = \rho\Delta\theta_{t-1} - \eta g_t(\theta_{t-1} + \rho\Delta\theta_{t-1}).$$

- Adam算法: 结合动量法与RMSprop算法
  - 计算梯度的指数加权平均:

$$M_t = \beta_1 M_{t-1} + (1-\beta_1)\mathrm{g}_t,$$

其中衰减率β_1通常设为0.9, $M_0 = 0$.

  - 计算梯度平方的指数加权平均:

$$G_t = \beta_2 G_{t-1} + (1-\beta_2)\mathrm{g}_t \odot \mathrm{g}_t,$$

其中衰减率β_2通常设为0.99, $G_0 = 0$.

  - 令

$$\hat{M}_t = \frac{M_t}{1-\beta_1^t},$$

$$\hat{G}_t = \frac{G_t}{1-\beta_2^t}$$

.

- Adam算法
  - 参数更新如下:

$$\Delta\theta_t = -\frac{\eta}{\sqrt{\hat{G}_t + \epsilon}} \odot \hat{M}_t.$$

  - η为初始学习率, 一般设为0.001.
  - 学习率也可以进行衰减, 比如$\eta_t = \frac{\eta_0}{\sqrt{t}}$.
- Nadam: 用Nesterov加速梯度法代替动量法, 对Adam算法进行改进.

- 梯度截断
  - 梯度消失 与 梯度爆炸
  - 如果梯度突然增大, 此时更新参数反而会导致其远离最优点.
  - 当梯度的模大于一定阈值时, 对梯度进行截断:
    - 按值截断: 一个参数的梯度值不在设置的阈值区间$[a, b]$内,就进行截断:

$$g_t = \max(\min(g_t, b), a)$$

    - 按模截断: 将梯度的模截断到给定的阈值b:

$$g_t = \begin{cases} g_t, & \| g_t \| \leq b, \\ \frac{b}{\|g_t\|} g_t, & \| g_t \| > b \end{cases}$$

      超参数b可以根据一段时间内的平均梯度来设定, 但实验发现训练过程对超参数并不十分敏感.
  - 按模截断是循环神经网络训练中避免梯度爆炸的有效方法.

概要

- 参数初始化: 关乎优化效率和泛化能力
  - 预训练初始化
    - 预训练+精调(Fine-tuning)
  - 固定值初始化
    - 根据经验用特殊值初始化一些参数.
    - 比如对使用Relu函数的神经元, 有时将偏置设置为0.01 可使得神经元在训练初期更容易激活.
  - 随机初始化
    - 对称权重现象: 参数都初始化为0, 会导致隐层神经元没有区分性.
    - 基于固定方差的参数初始化
    - 基于方差缩放的参数初始化
    - 正交初始化

- 基于固定方差的参数初始化
  - 从一个固定均值μ (通常$\mu = 0$)和方差 σ^2的分布中采样来生成参数的初始值.
    - 依高斯分布$\mathcal{N}(0, \sigma^2)$来初始化参数.
    - 以均匀分布$U(-r, r)$来初始化参数, 其中$r = \sqrt{3\sigma^2}$.
  - 如何设置σ^2:
    - 如果参数范围太小, 会导致神经元的输出过小, 经过多层传递信号就会慢慢消失; 也会使得sigmoid型激活函数丢失非线性的能力.
    - 如果参数范围太大, 会导致输入过大, 使得sigmoid型激活函数的激活值变得饱和, 梯度接近于0.
  - 基于固定方差的随机初始化方法一般配合逐层归一化使用来降低固定方差对网络性能以及优化效率的影响.

- 基于方差缩放的参数初始化
  - 参数初始化的区间应该根据神经元的性质进行差异化的设置.
  - 为了缓解梯度消失或爆炸问题, 尽可能保持每个神经元的输入与输出的方差一致.
  - 根据神经元的连接数量来自适应地调整初始化分布的方差.
  - Xavier初始化
  - He初始化

- Xavier初始化(1)
  - 考虑神经元 $y = a\left(\sum_{i=1}^{n} w_i x_i\right)$, 不妨设$a(z) = z$, 假定$w_i$和$x_i$相互独立且均值为0, 则

$$E(y) = 0, \quad \mathrm{Var}(y) = \sum_{i=1}^{n} \mathrm{Var}(w_i)\mathrm{Var}(x_i) = n\mathrm{Var}(w_i)\mathrm{Var}(x_i).$$

    要保持神经元的输入输出的方差尽可能一致, 则$\mathrm{Var}(w_i) = \frac{1}{n}$比较合理.
  - 在前馈神经网络中, 既要考虑信号正向传播, 也要考虑误差反向传播, 即对第l层神经元来说, 要兼顾

$$\mathrm{Var}(w_i^{(l)}) = \frac{1}{M_{l-1}}$$

    和

$$\mathrm{Var}(w_i^{(l)}) = \frac{1}{M_l}.$$

- Xavier初始化(2)
  - 作为折中, $\mathrm{Var}(w_i^{(l)}) = \frac{2}{M_{l-1}+M_l}$.
  - 进一步, 可以
    - 依高斯分布$\mathcal{N}(0, \frac{2}{M_{l-1}+M_l})$来初始化参数.
    - 以均匀分布$U(-\sqrt{\frac{6}{M_{l-1}+M_l}}, \sqrt{\frac{6}{M_{l-1}+M_l}})$来初始化参数.
  - 如果 $a(z) = \sigma(z)$, 则
    - 考虑在0附近的一阶泰勒展开, 则$\sigma(z) \approx 0.25z + 0.5$.
    - $\mathrm{Var}(w_i^{(l)}) = 16 \times \frac{2}{M_{l-1}+M_l}$.
  - 如果 $a(z) = \tanh(z)$, 则 $\mathrm{Var}(w_i^{(l)}) = \frac{2}{M_{l-1}+M_l}$.
    - 考虑在0附近的一阶泰勒展开, 则$\tanh(z) \approx z$.
    - $\mathrm{Var}(w_i^{(l)}) = \frac{2}{M_{l-1}+M_l}$.
- He初始化
  - 对采用Relu作为激活函数的神经元来说,

$$\mathrm{Var}(w_i^{(l)}) = \frac{2}{M_{l-1}}.$$

- 正交初始化(1)
  - 对每个参数的独立随机初始化可能依然存在梯度消失或爆炸问题.
  - 考虑一个L层的等宽线性网络

$$\mathbf{y} = W^{(L)^T} W^{(L-1)^T} \dots W^{(1)^T} \mathbf{x},$$

其中$W^{(l)} \in R^{M \times M}, 1 \le l \le M$, 则误差反向传播中, $\delta^{(l-1)} = W^{(l)^T} \delta^{(l)}$, 为了避免梯度消失或爆炸问题, 希望误差项具有保范性:

$$\| \delta^{(l-1)} \|^2 = \| \delta^{(l)} \|^2 = \| W^{(l)} \delta^{(l-1)} \|^2 .$$

  - 如果依分布$\mathcal{N}(0, \frac{1}{M})$采样, 则需要$M$足够大.

- 正交初始化(2)
  - 回顾: 如果 $W^{(l)}$是正交矩阵, 则:

$$\| \delta^{(l-1)} \|^2 = \| \delta^{(l)} \|^2 = \| W^{(l)} \delta^{(l-1)} \|^2 .$$

  - 依分布$\mathcal{N}(0,1)$采样, 则初始化一个矩阵M.
  - 对进行奇异值分解, 得到

$$M = U \Sigma V^T .$$

  - 以U 或 V 作为初始化的权重矩阵 $W^{(l)}$.

概要

- 尺度不变性与尺度敏感
  - 如果学习算法在缩放全部或者部分特征后不影响它的学习与预测, 则称该算法具有尺度不变性.
- 对尺度敏感的模型, 必须对样本进行预处理, 将各维特征转化到相同的取值区间, 并且消除不同特征之间的相关性.
- 神经网络理论上可以通过参数的调整来适应不同特征的尺度， 从而具有尺度不变性.
- 但尺度不同的输入特征会增加训练难度.
  - 参数初始化要考虑各维特征的不同尺度 .
  - 当各维特征的尺度差异比较大的时候也影响梯度下降法的效率.

- 归一化：把数据特征转换为相同尺度的方法.
  - 最小最大值归一化：给定样本集 $\{\mathbf{x}_i\}_{i=1}^N$，对$\mathbf{x}_i$的每一维特征$x_i^{(k)}$进行如下归一化：

$$\hat{x}_i^{(k)} = \frac{x_i^{(k)} - \min\limits_{1\le j\le N} x_j^{(k)}}{\max\limits_{1\le j\le N} x_j^{(k)} - \min\limits_{1\le j\le N} x_j^{(k)}}$$

  - 标准化：

$$\hat{x}_i^{(k)} = \frac{x_i^{(k)} - u^{(k)}}{\sigma^{(k)}}$$

- 白化：降低特征之间的冗余性，并且白化后所有特征具有相同的方差。例如可用PCA方法消除各个成分之间的相关性。

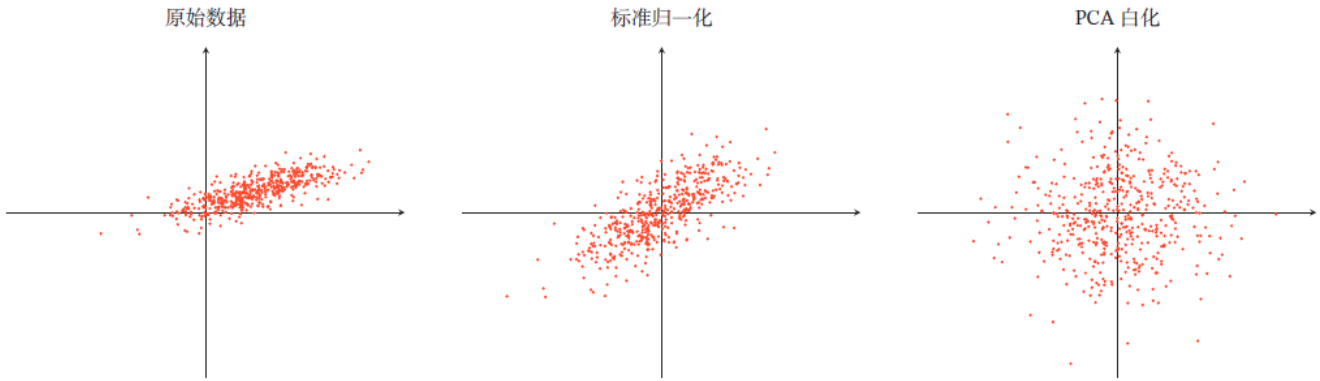

图 7.10　标准归一化和 PCA 白化

- 将隐层的输入进行归一化，可使得网络训练更容易.
  - 更好的尺度不变性:
    - 对网络中的神经元来说，其前面层的参数修正可能引起该神经元输入分布的改变.
    - 对每层神经元的输入进行归一化来保持分布稳定.
  - 更平滑的优化地形:
    - 逐层归一化可使得大部分神经元的输入处于不饱和区域以避免梯度消失.
    - 更平滑的优化地形使得梯度更加稳定，可允许采用更大的学习率来提高收敛速度.
  - 逐层归一化方法:
    - 批量归一化、层归一化、权重归一化、局部响应归一化

- 批量归一化
  - 考虑第l层:

$$\mathbf{h}^{(l)} = f^{(l)}(\mathbf{h}^{(l-1)}) = a(\mathbf{z}^{(l)}) = a({\mathbf{W}^{(l)}}^T \mathbf{h}^{(l-1)} + \mathbf{b}^{(l)})$$

    在实践中归一化操作一般应用在$\mathbf{z}^{(l)}$上, 即仿射变换之后, 激活函数之前.

- 使用标准化将$\mathbf{z}^{(l)}$的每一维都归一到标准正态分布:

$$\hat{\mathbf{z}}^{(l)} = \frac{\mathbf{z}^{(l)} - E[\mathbf{z}^{(l)}]}{\sqrt{\mathrm{Var}(\mathbf{z}^{(l)}) + \epsilon}}.$$

- 小批量梯度下降法通常用当前小批量样本集对应的 经验均值和方差近似估计$E[\mathbf{z}^{(l)}]$和$\mathrm{Var}(\mathbf{z}^{(l)})$.

- 批量归一化
  - 小批量梯度下降法通常用当前小批量样本集对应的 经验均值和方差近似估计$E[\mathbf{z}^{(l)}]$和$\mathrm{Var}(\mathbf{z}^{(l)})$:
    - 设批量大小为n, 则$\mathbf{z}^{(l)}$关于批量样本的经验均值和方差分别为:

$$\mu_B = \frac{1}{n}\sum_{i=1}^{n}\mathbf{z}^{(i,l)}, \quad \sigma_B^2 = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{z}^{(i,l)} - \mu_B) \odot (\mathbf{z}^{(i,l)} - \mu_B).$$

    - 如果直接标准化, 会使得$\mathbf{z}^{(l)}$的取值集中在0附近, 接近 sigmoid 型激活函数的线性近似区间, 可能减弱神经网络的非线性性质.
    - 引进缩放和平移变换:

$$BN_{\gamma,\beta}(\mathbf{z}^{(l)}) = \frac{\mathbf{z}^{(l)} - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \odot \gamma + \beta$$

- 批量归一化
  - 引进缩放和平移变换:

$$BN_{\gamma,\beta}(\mathbf{z}^{(l)}) = \frac{\mathbf{z}^{(l)} - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \odot \gamma + \beta$$

  - 批量归一化可以看作在激活函数之间加的一个特殊层:

$$\mathbf{h}^{(l)} = a(BN_{\gamma,\beta}(\mathbf{z}^{(l)})) = a(BN_{\gamma,\beta}(\mathbf{W}^{(l)^T}\mathbf{h}^{(l-1)})),$$

    此时仿射变换中不在需要偏置参数.
  - 在训练完成时, 用整个数据集上的均值和方差来分别代替每次小批量样本的μ_B和σ_B^2.
  - 批量归一化的正则化意义: 在针对一给定样本训练时, 引入其他样本信息进行扰动.

- 层归一化
  - 批量归一化只针对单个神经元，而且批量大小不能太小.
  - 批量归一化操作不适合如循环神经网络等中神经元的净输入的分布是动态变化的情形.
  - 层归一化对一个中间层的所有神经元进行归一化.
    - 第l层净输入$\mathbf{z}^{(l)}$ 的均值和方差分别为

$$\mu^{(l)} = \frac{1}{M_l}\sum_{i=1}^{M_l}\mathbf{z}_i^{(l)}, \quad \sigma^{(l)2} = \frac{1}{M_l}\sum_{i=1}^{M_l}(\mathbf{z}_i^{(l)} - \mu^{(l)})^2.$$

则层归一化为

$$LN_{\gamma,\beta}(\mathbf{z}^{(l)}) = \frac{\mathbf{z}^{(l)} - \mu^{(l)}}{\sqrt{\sigma^{(l)2} + \epsilon}} \odot \gamma + \beta.$$

- 权重归一化
  - 考虑第l层:

$$\mathbf{h}^{(l)} = f^{(l)}(\mathbf{h}^{(l-1)}) = a(\mathbf{z}^{(l)}) = a({\mathbf{W}^{(l)}}^T \mathbf{h}^{(l-1)} + \mathbf{b}^{(l)})$$

通过再参数化, 将权重分解为长度和方向两种参数:

$$W_{:,i} = g_i \frac{\mathbf{v}_i}{\| \mathbf{v}_i \|}.$$

这里$1 \leq i \leq M_l$.

- 在权重共享的情况下, 权重归一化的开销会比较少.
- 通过梯度下降法优化长度和方向:

$$\nabla_g L = \frac{\nabla_{\mathbf{w}} L \cdot \mathbf{v}}{\| \mathbf{v} \|}$$

$$\nabla_{\mathbf{v}} L = \frac{g}{\| \mathbf{v} \|} \nabla_{\mathbf{w}} L - \frac{g \nabla_g L}{\| \mathbf{v} \|^2} \mathbf{v}$$
$$= \frac{g}{\| \mathbf{v} \|} \left(\mathbf{I} - \frac{\mathbf{w}\mathbf{w}'}{\| \mathbf{w} \|^2} \right) \nabla_{\mathbf{w}} L$$

- 缩放权重梯度的长度;
- 矫正权重梯度的方向.

局部响应归一化:

- 对邻近的神经元(的激活值)进行归一化, 类似于生物神经元中的侧抑制现象, 即活跃神经元对相邻神经元具有抑制作用, 通常用于卷积神经网络.

概要

- 常见的超参数:
  - 正则化系数.
  - 优化参数, 如优化方法、学习率、批量大小等.
  - 网络结构, 如神经元之间的连接关系、层数、每层的神经元数量、激活函数的类型等.
- 超参数优化的困难:
  - 超参数优化是一个组合优化问题, 既不能像一般参数优化一样通过梯度下降法求解, 也没有一种通用有效的优化方法.
  - 评估一组超参数配置的时间代价通常很高.
- 常用的超参数配置方法:
  - 网格搜索 与 随机搜索
  - 贝叶斯优化 与 动态资源分配
  - 神经架构搜索

- 网络正则化
  - 回顾给定训练数据集$T = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$，学习$f(\mathbf{x}; \hat{\theta})$:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \left[\frac{1}{N} \sum_{i=1}^{N} L(f(\mathbf{x}_i; \theta), y_i) + \lambda \Omega(f) \right].$$

  - 在训练深度神经网络时，特别是在过度参数化时，l_1 或l_2 正则化的效果往往不如浅层机器学习模型中显著.
  - 权重衰减正则化:

$$\theta_t \leftarrow (1 - \beta)\theta_{t-1} - \eta \mathrm{g}_t.$$

    - 在标准的随机梯度下降法中，权重衰减正则化与l_2 正则化的效果相同.
    - 但在如Adam等较为复杂的优化方法中，二者并不等价.

- 丢弃法或暂退法(dropout)
  - 在训练网络时, 可以随机丢弃一部分神经元(包括其对应的连接边)来避免过拟合.
  - 可以对层$\mathbf{h} = a(W^T\mathbf{x} + b)$设置一个固定的概率$p$来判定每个神经元是否被保留, 即以概率$p$保留, 以概率$1-p$丢弃.
    - 引入掩蔽函数

$$mask(\mathbf{x}) = \begin{cases} \mathbf{m} \odot \mathbf{x}, & \text{训练阶段} \\ p\mathbf{x}, & \text{测试阶段} \end{cases}$$

      其中$\mathbf{m} \in \{0,1\}^{|\mathbf{x}|}$是丢弃掩码, 通过以概率为$p$的伯努利分布随机生成.
    - $\mathbf{h} = a(W^T mask(\mathbf{x}) + b)$
  - 一般对隐层来说, 选择$p = 0.5$时效果最好. 对输入层来说, p设置为更接近1的值.

- 逆(或者倒置)丢弃法(inverted dropout)
  - 为了实现的方便, 一般采用倒置丢弃法.
  - 我们设保留概率为p, 丢弃概率为$1-p$, 随机变量ξ为1, 0的概率分别为p 和$1-p$, 对神经元h 来说, 以概率p做$\frac{1}{p}$倍拉伸, 以概率$1-p$被清零, 即新的神经元为

$$h' = \frac{\xi}{p} h.$$

  - $E[h'] = \frac{E[\xi]}{p} h = h.$
  - 在测试模型时, 为了得到更加确定性的结构, 不使用丢弃法.

- 数据增强:
  - 通过对训练数据做一些随机改变， 产生相似但不相同的样本增加数据量(多样性)， 来提高模型鲁棒性,避免过拟合.
  - 图像数据增强的方法:
    - 旋转
    - 翻转
    - 缩放
    - 平移
    - 加噪声

- 标签平滑
  - 在样本标签中添加噪声来避免模型过拟合.
    - 多类分类任务中$\mathbf{x}$的标签(硬目标)表示为独热向量

$$\mathbf{y} = [0, \cdots, 0, 1, 0 \cdots, 0]^T.$$

如果使用softmax分类器并使用交叉熵损失函数，要使得正确类(1)的概率接近1，其未归一化的得分需要远大于其它类(0)的得分，可能会导致其权重越来越大.

    - 如果标签错误, 会导致更严重的过拟合.
    - 引入噪声: 假设一共有K个标签, 样本以ϵ的概率为其它类, 则平滑后的标签(软目标)为

$$\hat{\mathbf{y}} = [\frac{\epsilon}{K-1}, \cdots, \frac{\epsilon}{K-1}, 1-\epsilon, \frac{\epsilon}{K-1} \cdots, \frac{\epsilon}{K-1}]^T.$$

  - 更多可参看
    Rafael M ü ller, Simon Kornblith, Geoffrey E. Hinton: When does label smoothing help? NeurIPS 2019: 4696-4705

小结

- 从单一层前馈网络当多层前馈网络
- 小批量梯度下降法
- 优化方法
  - 批量大小的选择
  - 学习率的调整
    - 学习率衰减方法、 学习率预热方法
    - 周期性学习率调整方法
    - 自适应学习率调整方法
  - 梯度估计修正
    - 动量法、 Nesterov加速梯度
    - Adam算法
    - 梯度截断

小结

- 参数初始化
- 数据预处理
- 逐层归一化
- 超参数优化
- 网络正则化

相关参考

本资料讲内容主要参考[1]书的第23、29 章以及[2]书的第7章.

[1] 李航, 机器学习方法, 第三版, 清华大学出版社, 2022.

[2] 邱锡鹏, 神经网络与深度学习, 机械工业出版社, 2020.