No.

Date.

第6章 网络优化与正则化

优化问题：梯度消失 + 参数多

泛化问题：拟合能力过强

回顾

机器学习理论　VC 维

Rademacher 复杂度

L2 和 L1 正则化在 NN 之作用有限，另一些经验做法 提前停止 和 SGD 又有效。

泛化能力的规律。

6.1 网络优化的难点

结构多样性

高维变量的非凸优化

鞍点

平坦底部

No.

Date.

6.2 优化算法

1. 小批量梯度下降
2. 随机梯度下降
3. 批量梯度下降

} 共同问题 ⇒

1. 如何初始化参数
2. 设定超参数据
3. 如何选择合适的学习率

6.2.1 小批量梯度下降

$$\theta_t = \theta_{t-1} - \lambda \frac{\partial R(\theta)}{\partial \theta}$$

- 学习率衰减
- 梯度方向优化.

6.2.2 学习率衰减

⇒ AdaGrad 算法

原因：每个参数的维度上收敛速度都不相同，因此根据不同参数的收敛情况分别设置学习率.

方案：借鉴L2正则化思想，每次迭代时自适应地调整每个参数的学习率。在第t次迭代时，先计算每个梯度参数平方的累计值.

$$G_t = \sum_{\tau} g_\tau \odot g_\tau$$

g_τ 是第 τ 次迭代时的梯度

No.

Date.

$$\Delta\theta_t = -\frac{\alpha}{\sqrt{G_t+\varepsilon}} \odot \vec{g}_t$$

α 初始学习率

ε 保持数值稳定性 $e^{-7} \sim e^{-10}$

优点：某个参数的偏导数累积较大，其学习率相对较小。

——— 较小，- - - - 较大。

整体上随着迭代次数增加，学习率逐渐缩小。

缺点：经过一定次数的迭代仍然没有找到最优点，学习率已经很小无法找了。

⇒ RMSprop 算法（改进 AdaGrad）

G_t 每次迭代梯度平方的指数衰减移动平均。

$$G_t = \beta G_{t-1} + (1-\beta)\vec{g}_t \odot \vec{g}_t$$

β decay rate

$$\Delta\theta_t = -\frac{\alpha}{\sqrt{G_t+\varepsilon}} \odot \vec{g}_t$$

与 AdaGrad 的区别

G_t 由累积变为指数衰减移动平均。

优点：一定程度上解决了学习率过早衰减的缺点。

No.

Date.

⇒ AdaDelta 算法（改进 AdaGrad）

- 梯度的平方指数衰减移动平均

- 每次参数更新差的 $\Delta\theta$ 的平方指数移动平均

$$\Delta X_{t-1}^2 = \beta_1 \Delta X_{t-2}^2 + (1-\beta_1)\Delta\theta_{t-1} \odot \Delta\theta_{t-1}$$

β_1 衰减率

$\Delta\theta_t$ 未知，故只计算到 ΔX_{t-1}

更新公式为

$$\Delta\theta_t = -\frac{\sqrt{\Delta X_{t-1}^2 + \varepsilon}}{\sqrt{G_t + \varepsilon}} \cdot \vec{g}_t$$

优点：一定程度上平抑了学习率的波动

6.2.3 梯度方向优化

⇒ 动量法

方案：用之前积累的动量来替代真正的梯度。

每次迭代的梯度看作加速度

$\Delta\theta_t$ 负梯度的移动加权平均。

$$\Delta\theta_t = \rho\Delta\theta_{t-1} - \lambda\vec{g}_t$$

ρ 动量因子，一般取 0.9

λ 学习率

No.

Date. . .

⇒ Nesterov 加速梯度（改进动量法）

动量法中两个部分 $\begin{cases} \Delta\theta_{t-1} \\ \vec{g}_t \end{cases}$

先根据 $\Delta\theta_{t-1}$ 更新得到参数 $\hat{\theta}$ $\quad \hat{\theta} = \theta_{t-1} + \rho\Delta\theta_{t-1}$

再用 $\vec{g}_t$ 更新 $\quad \theta_t = \hat{\theta} - \alpha\vec{g}_t$

更新问改为 $\hat{\theta}$ 处的梯度

$$\Delta\theta_t = \rho\theta_{t-1} - \alpha\vec{g}_t(\theta_{t-1} + \rho\Delta\theta_{t-1})$$

⇒ AdaM 算法

G_t 与 RMSprop 类似，计算 $\vec{g}_t$ 的平方指数加权平均

M_t 与动量法类似，计算 $\vec{g}_t$ 的指数加权平均

~~$G_t = \beta_1 G_{t-1} + (1-\beta_1$~~

M_t 与动量法类似，计算 $\vec{g}_t$ 的指数加权平均

G_t 与 RMSprop 类似，计算 $\vec{g}_t$ 的平方指数加权平均

$$M_t = \beta_1 M_{t-1} + (1-\beta_1)\vec{g}_t \qquad \beta_1 = 0.9$$

$$G_t = \beta_2 G_{t-1} + (1-\beta_2)\vec{g}_t \odot \vec{g}_t \qquad \beta_2 = 0.99$$

No.

Date.

当 $M_0 = 0$, $G_0 = 0$ 时, $\beta_1, \beta_2 \to 1$ 时偏差较大.

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

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

参数更新差值为

$$\Delta\theta_t = -\frac{\alpha}{\sqrt{G_t + \varepsilon}} \hat{M}_t$$

=) 梯度裁剪

目的: 梯度爆炸

⇒⇒ 按值裁断

~~小于a时, 设为a;~~

小于b时 设为b;

$$\vec{g}_t = \max(\min(\vec{g}_t, b), a)$$

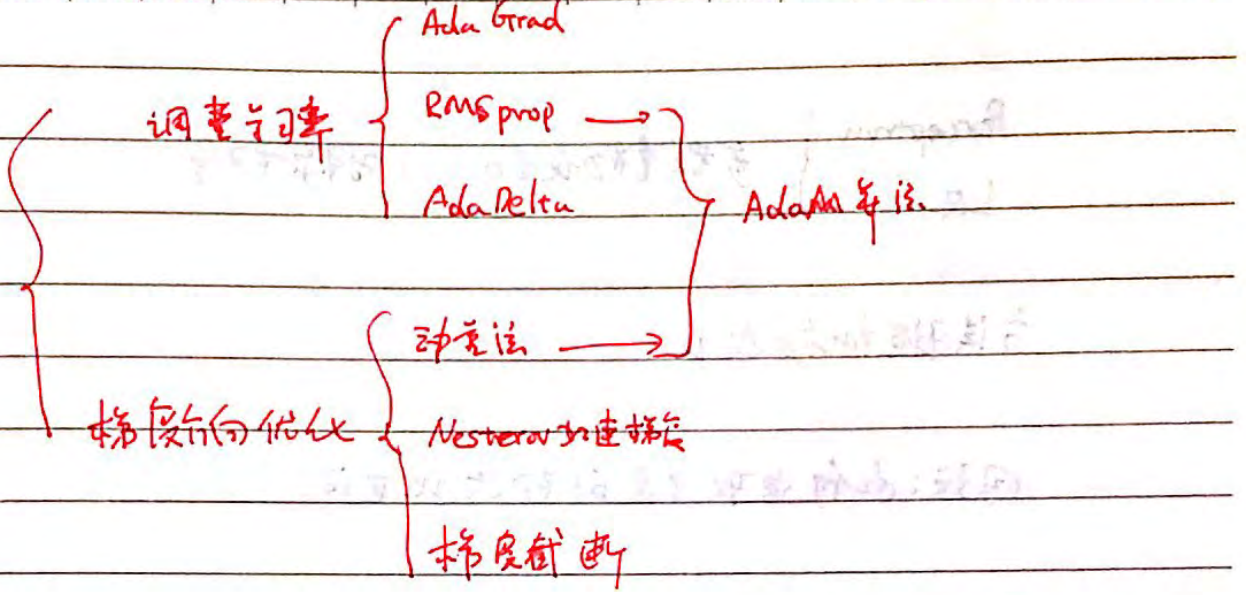
⇒⇒ 按模裁断

$$\vec{g}_t = \frac{b}{\|\vec{g}_t\|} \vec{g}_t, \text{ 当 } \|\vec{g}_t\|^2 > b \text{ 时.}$$

No.

Date. . .

6.2.4 优化算法小结.



No.

Date.

6.3 参数初始化

Perceptron
LR
} 参数都初化为0 ⇒ 对称权重

方法：随机初始化

问题：如何选取合适的初始化区间.

6.3.1 Gaussian 初始化

固定均值 + 固定方差

输入连接为 n_{in} 时，$N(0, \sqrt{\frac{1}{n_{in}}})$

6.3.2 均匀分布初始化

给定一个区间 $[-r, r]$ 内采用均匀分布来初始化参数

r 超参数，按神经元数量自适应调整

⇒ Xavier 初始化方法

Logistic $r = \sqrt{\frac{6}{n^{(l)} + n^{(l+1)}}}$

No.

Date. . .

6.4 数据预处理

问题：特征值范围差异

⇒ 缩放归一化

将每个特征归一到 $[0,1]$ 或 $[-1,1]$ 之间.

$$x^{(i)} = \frac{x - \min}{\max - \min}$$

⇒ 标准归一化

处理成符合标准正态分布，每个特征.

先计算 μ 和 σ^2

$$\mu = \frac{1}{N}\sum x^{(i)}$$

$$\sigma^2 = \frac{1}{N}\sum (x^{(i)} - \mu)^2$$

新的特征值.

$$\hat{x}_i^{(i)} = \frac{x^{(i)} - \mu}{\sigma}$$

⇒ 白化

白化后，每个特征之间相关性较低，并且所有特征具有相同的方差.

方法：PCA

No.

Date.

6.5 逐层归一化

内部协变量偏移

6.5.1 批量归一化 (BN) 对中间层的单个神经元

$$\vec{a}^{(l)} = f(\vec{z}^{(l)}) = f(W\vec{a}^{(l)} + \vec{b})$$

逐层归一化，在中间层上操作，效率更高，使用标准归一化，将一个 $\vec{z}^{(l)}$ 的

$\vec{a}^{(l)}$ 分布性质和 $\vec{z}^{(l)}$ 稳定

$$\hat{\vec{z}}^{(l)} = \frac{\vec{z}^{(l)} - E(\vec{z}^{(l)})}{\sqrt{Var(\vec{z}^{(l)}) + \epsilon}}$$

如采用 sigmoid 时，在个取值区间刚好是接近线性变换的区间，减弱了网络的非线性性质。

方法：附加的缩放和平移变换改变取值区间。

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

$$= BN_{\gamma,\beta}(\vec{z}^{(l)})$$

批量归一化操作可以看作一个特殊的神经层，加在每一层非线性激活函数

$$\vec{a}^{(l)} = f(BN_{\gamma,\beta}(\vec{z}^{(l)})) = f(BN_{\gamma,\beta}(W\vec{a}^{(l-1)}))$$

No.

Date.

6.5.2 层归一化

BN要求批量不能太小，无法应用于循环神经网络。

层归一化是对中间层的所有神经元进行归一化

$$\mu^{(l)} = \frac{1}{n^l} \sum_i z_i^{(l)}$$

$$\sigma^{(l)} = \frac{1}{n^l} \sum_i (z_i^{(l)} - \mu^{(l)})^2$$

层归一化定义为

$$\hat{z}^{(l)} = \frac{z^{(l)} - \mu^{(l)}}{\sqrt{\sigma^{(l)2} + \epsilon}} \odot r + \beta$$

$$\triangleq LN_{r,\beta}(z^{(l)})$$

⇒ 循环网络中的层归一化

$$z_t = U h_{t-1} + W x_t$$

$$h_t = f(LN_{r,\beta}(z_t))$$

缓解梯度爆炸或消失

⇒ 对比BN和LN

对长个样本的小批量条件 $Z^{(l)} = [z^{(l,1)}; \dots; z^{(l,K)}]$，层归一化是对矩阵每列进行归一化，而BN是对每行进行归一化。

一般而言，BN是更好的选择。

Batch较小时，可以选择LN。

No.

Date.

6.5.3 权重归一化 (WN)

对神经网络的连接权重进行归一化，再通过参数化方法，将连接权重分解为长度和方向两种参数

$$\vec{a}^{(l)} = f(W\vec{a}^{(l-1)} + \vec{b})$$

$$w_{i:} = \frac{g_i}{\|\vec{v}_i\|}\vec{v}_i$$

$w_{i:}$：表示权重 W 的第 i 行

g_i 为标量

$\vec{v}_i$ 和 $\vec{a}^{(l-1)}$ 维数相同

优点：归一化可输较小

6.5.4 局部相应归一化 (LRN)

通常用于 CNN

$$\hat{Y}^p = Y^p / \left(k + \alpha \sum (Y^j)^2\right)^\beta$$

$$\triangleq LRN_{n,k,\alpha,\beta}(Y^p)$$

侧抑制

No.

Date. . .

6.6 超参数优化

常见超参数有

- 网络结构
- 优化参数
- 正则化系数

6.6.1 网格搜索

尝试所有的超参数组合.

6.6.2 随机搜索

随机组合 ⇒ 选最好的. -

一般比网格搜索更加高效.

6.6.3 贝叶斯优化

时序模型优化 SMBO

$$EI(x,H)=\int_{-\infty}^{+\infty}\max(y^*-y,0)\,p_{GP}(y|x,H)\,dy$$

No.

Date. . .

6.6.4 动态资源分配

最优解问题

逐次逼近

多臂赌博机

⇒ 神经架构搜索

No.

Date.

6.7 网络正则化

6.7.1 L1和L2正则化

正则化⇒限制模型复杂度

问题：过多参数下，L1和L2正则化效果一般？

方案：数据增强、提前停止、丢弃法、集成法

$$\theta^* = \arg\min_{\theta} \frac{1}{N}\sum L(y^{(n)}, f(\vec{x}^{(n)}, \theta)) + \lambda l_p(\theta)$$

等价于

$$\theta^* = \arg\min_{\theta} \frac{1}{N}\sum L(y^{(n)}, f(\vec{x}, \theta))$$

subject to $l_p(\theta) < 1$

L_1范数在零点不可导，常用下式来近似

$$L_1(\theta) = \sum\sqrt{\theta^2 + \epsilon}$$

弹性网络正则化

同时加入L_1和L_2正则化

$$\theta^* = \arg\min_{\theta} L(y^{(n)}, f(\vec{x}^{(n)}, \theta)) + \lambda_1 L_1(\theta) + \lambda_2 l_2(\theta)$$

No.

Date.

6.7.2 权重衰减

$$\theta_t = (1-w)\theta_{t-1} - \alpha \vec{g}_t$$

在标准的SGD中，权重衰减与L2正则化效果相同。

在Adam中，并不等价。

6.7.3 提前停止

验证集，

用验证集上的错误来代替期望错误。

6.7.4 丢弃法

$d(\cdot)$ 丢弃函数

$$\vec{y} = f(W d(\vec{x}) + \vec{b})$$

$$d(\vec{x}) = \begin{cases} \vec{m} \odot \vec{x} & \text{当训练阶段} \\ p\vec{x} & \text{当测试阶段} \end{cases}$$

$\vec{m} \in \{0,1\}^d$ 丢弃掩码

在测试时，将每个神经元的输出乘以p，相当于不同的网络做了平均。

p一般等于0.5

⇒ 集成学习的角度解释

每次Dropout，相当于从原网络中采样得到一个子网络。

n个神经元 ⇒ 2^n个子网络。

集成了指数级个不同的组合模型。

No.

Date. . .

⇒ 贝叶斯学习的解释

假设参数 θ 为随机向量，且先验分布为 $q(\theta)$，模型为 $f(\vec{x},\theta)$

贝叶斯方法的预测为

$$E_{q(\theta)}[y] = \int_{\theta} f(\vec{x},\theta)\, q(\theta)\, d\theta$$

$$\approx \frac{1}{m}\sum_{m} f(\vec{x},\theta_m)$$

$f(\vec{x},\theta_m)$ 第 m 次应用 ~~Dropout~~ Dropout 后的网络。

⇒ 循环神经网络上的丢弃法。

问题：

不能对每个时刻的隐状态进行随机丢弃，这样会损害神经网络在时间维度上的记忆能力。

变分丢弃法：

对参数矩阵的每个元素进行随机丢弃，并且在所有时刻都使用相同的掩码。

6.7.5 数据增强（图像）

旋转（Rotation）

翻转（Flip）

缩放（Zoom In/Out）

平移（Shift）

加噪声（Noise）

No.

Date

6.7.6 标签平滑

在输出标签中添加噪声来避免模型过拟合.

不变目标：one-hot向量

软目标：假设样本以ϵ的概率为其它类

$$\tilde{y} = [\frac{\epsilon}{k-1}, \cdots, 1-\epsilon, \frac{\epsilon}{k-1}]^T$$