深度学习复习笔记

2024 年春-张晓平

作者：FANG Changrui

组织：School of Mathematics and Statistics, Wuhan University

时间：Friday 14th June, 2024

超越人类极限，做宇宙的主人

目录

第 1 章 声明 **1**

第 2 章 考试重点 **2**

第 3 章 线性回归 **3**

3.1 最小二乘法 (Least Square Method) . . . 3

3.1.1 最小二乘求解 (Least Square Method) . . . 3

3.1.2 线性最小二乘 . . . 3

3.2 正则化方法 (Regularization) . . . 3

3.3 梯度下降法 (GD) . . . 4

3.4 重要思考题及解析 . . . 4

第 4 章 浅层/深度神经网络 **5**

4.1 激活函数 . . . 5

4.2 逼近定理 . . . 5

4.3 术语 . . . 5

4.4 比较 . . . 5

4.5 重要思考题及解析 . . . 5

第 5 章 损失函数 **7**

5.1 损失函数的构造方法 . . . 7

5.2 回归问题损失函数 . . . 7

5.3 分类问题损失函数 . . . 8

5.3.1 二分类 . . . 8

5.3.2 多分类 . . . 8

5.4 多输出问题 . . . 8

5.5 交叉熵损失 . . . 9

5.6 重要思考题及解析 . . . 9

第 6 章 优化算法 **10**

6.1 梯度下降法 (Gradient Descent Method) . . . 10

6.1.1 线性模型 . . . 10

6.1.2 Gabor 模型 . . . 10

6.2 随机梯度下降 (Stochastic Gradient Descent) . . . 10

6.3 带动量的随机梯度下降 (SGD with Momentum) . . . 10

6.4 Nesterov 加速动量法 . . . 11

6.5 梯度归一化 (Normalized Gradient) . . . 11

6.6 AdaGrad(Adaptive Gradient Algorithm) . . . 12

6.7 RMSprop . . . 12

6.8 AdaDelta . . . 12

6.9 Adam(Adaptive Moment Estimation) . . . 13

6.10 超参数调优 . . . 13

6.11 重要思考题及解析 . . . 13

第 7 章 卷积神经网络 **14**
7.1 重要思考题及解析 14

第 8 章 残差网络 **15**
8.1 概念 15
8.2 梯度爆炸 15
8.3 常见残差架构 16
8.4 重要思考题及解析 16

第 9 章 Attention is all you need **17**
9.1 注意力机制 17
9.2 重要思考题及解析 17

第 1 章 声明

本文档仅做学习交流使用，仅在武汉大学数学与统计学院内部传阅，禁止用于商业用途，如有侵犯老师或各方权益，请联系删除，邮箱：作者邮箱。

本文档限于水平不足，一定存在诸多问题，如遇到问题，欢迎指正，邮箱：作者邮箱。

本文档唯一网上版本见 Github：WHU_MATH_course。其上包含武汉大学数学与统计学院课程的学习笔记、历年复习题、经验分享等，如有需要，欢迎下载。

第 2 章 考试重点

深度学习三要点：模型选择、损失函数、优化算法

1. 线性回归 (最小二乘问题，法方程组的推导，梯度下降法 GD，QR 分解 (不考))
2. 浅层神经网络：ReLU 表达的函数空间 (高维的话是分片线性函数)，节点，每一段斜率，ReLU、Sigmoid 基本性质
3. 深度神经网络：差不多
4. 损失函数：构造 (比如先验正态分布，可以推导出均方误差损失；若标签 yi 在 [0,1]，伯努利分布，二元交叉熵；分类的话，一般交叉熵；一般的分布，如何推导损失函数？)；似然函数 (最大似然估计，独立同分布)；这是一个完整体系，全面聚焦
5. 优化算法：全面掌握基本的优化算法，给一个示例改造成现代的算法；反向传播算法深刻理解 (给一个模型能够算出关于某个输入的梯度，公式)
6. 8-9 章 (正则化-性能评价) 不考
7. 卷积神经网络 (全面了解，基本操作，相关概念)
8. 残差网络：batch normalization，构造形式 (合理和不合理的)
9. Attention：深刻理解 (最后一节课的)，+self-attention，基本原理，基本推导，网络架构的搭配使用
10. 6-7 题，好好看给的题目

第 3 章 线性回归

重点：线性回归 (最小二乘问题，法方程组的推导，梯度下降法 GD，QR 分解 (不考))

3.1 最小二乘法 (Least Square Method)

曲线拟合：给定训练集 $\mathcal{D}=\{(\boldsymbol{x}_1,y_1),(\boldsymbol{x}_2,y_2),\cdots,(\boldsymbol{x}_N,y_N)\}$，其中 $\boldsymbol{x}_i\in\mathbb{R}^d$，$y_i\in\mathbb{R}$，我们的目标是找到一个函数 $f(\boldsymbol{x})$，使得 $f(\boldsymbol{x}_i)\approx y_i$。

线性回归：记 $\boldsymbol{x}=[1,x_1,\cdots,x_d]^T\in\mathbb{R}^{d+1}, w=[w_0,w_1,\cdots,w_K]^T\in\mathbb{R}^{K+1}, \varphi(\boldsymbol{x})=[1,\varphi_1(\boldsymbol{x}),\cdots,\varphi_K(\boldsymbol{x})]^T\in\mathbb{R}^{K+1}$，则 $y=f(\boldsymbol{x};\boldsymbol{w})=w_0+\sum\limits_{k=1}^{K}w_k\varphi_k(\boldsymbol{x})=\boldsymbol{w}^T\boldsymbol{\varphi}(\boldsymbol{x})$，其中 $\varphi:\mathbb{R}^{d+1}\to\mathbb{R}^{K+1}$ 为**特征提取函数**。

注 **线性**是指 y 关于 $\boldsymbol{w}$ 是线性的，而非 $\boldsymbol{x}$。常见基函数：多项式、径向、傅里叶、小波基函数。

3.1.1 最小二乘求解 (Least Square Method)

误差平方和 $E(\boldsymbol{w})=\frac{1}{2}\sum\limits_{n=1}^{N}\left[y(\boldsymbol{x}_n;\boldsymbol{w})-y_n\right]^2\Rightarrow w^*=\arg\min_{\boldsymbol{w}}E(\boldsymbol{w})$

记 $z=\varphi(x), z_n=\varphi(x_n)\Rightarrow E(\boldsymbol{w})=\frac{1}{2}\sum\limits_{n=1}^{N}\left(\boldsymbol{z}_n^T\boldsymbol{w}-y_n\right)^2$，再记 $Z=\begin{bmatrix}z_1^T\\ \vdots\\ z_N^T\end{bmatrix}\in\mathbb{R}^{N\times(K+1)}$，$\boldsymbol{y}=[y_1,\cdots,y_N]^T\in\mathbb{R}^N\Rightarrow E(w)=\frac{1}{2}\|Zw-y\|^2=\frac{1}{2}w^TZ^TZw-y^TZw+\frac{1}{2}y^Ty$，令 $\nabla_{\boldsymbol{w}}E(w)=0\Rightarrow Z^TZw-Z^Ty=0\Rightarrow w^*=Z^\dagger y=(Z^TZ)^{-1}Z^Ty$

3.1.2 线性最小二乘

令 $f(x)=a_1\varphi_1(x)+a_2\varphi_2(x)+\cdots+a_n\varphi_n(x)$，$m>n$，将数据 $\{(x_i,y_i)\}_{i=1}^m$ 代入得到**矛盾方程组** $(m\gg n)$

$$Ca=y,\quad C\in\mathbb{R}^{m\times n}, a\in\mathbb{R}^n, y\in\mathbb{R}^m.$$

问题转化：定义**偏差向量** $\delta=Ca-y$，根据最小二乘原则将其转化为求解优化问题

$$\boldsymbol{a}^*=\arg\min_{\boldsymbol{a}\in\mathbb{R}^n}Q$$

其中 $Q=\|\delta\|_2^2=\|Ca-y\|_2^2=a^TCa-2a^TC^Ty+y^Ty$，$\boldsymbol{a}^*$ 称为矛盾方程组的**最小二乘解**。

法方程组推导：Q 取得极值必要条件：$\frac{\partial Q}{\partial a}=\frac{\partial}{\partial a}\left(a^TC^TCa-2a^TC^Ty+y^Ty\right)=0\Rightarrow C^TCa=C^Ty$，其中

$$C=\begin{bmatrix}a_{11}&a_{12}&\cdots&a_{1n}\\ a_{21}&a_{22}&\cdots&a_{2n}\\ \vdots&\vdots&&\vdots\\ a_{m1}&a_{m2}&\cdots&a_{mn}\end{bmatrix},\quad a=\begin{bmatrix}x_1\\ x_2\\ \vdots\\ x_n\end{bmatrix},\quad y=\begin{bmatrix}b_1\\ b_2\\ \vdots\\ b_m\end{bmatrix}$$

因此使用最小二乘方法求解矛盾方程组 $Ca=y$ 的步骤为：

1. 计算 C^TC 和 C^Ty 得**法方程组**
2. 求解法方程组 $C^TCa=C^Ty$ 得最小二乘解 $\boldsymbol{a}^*=(C^TC)^{-1}Cy=C^\dagger y$，其中 $C^\dagger=(C^TC)^{-1}C^T\in\mathbb{R}^{n\times m}$ 为 C 的 **Moore-Penrose 广义逆**

3.2 正则化方法 (Regularization)

为防止**过拟合**，令 $E_{\text{reg}}(w)=\frac{1}{2}\|Zw-y\|+\lambda J(w)=E(w)+\lambda J(w)\Rightarrow w_{reg}^*=\arg\min_{\boldsymbol{w}}E_{\text{reg}}(\boldsymbol{w})$

Ridge regression：$J(w)=\frac{\|w\|^2}{2}\Rightarrow\nabla_{\boldsymbol{w}}E(w)=\left(Z^TZ+\lambda I\right)w-Z^Ty=0\Rightarrow w_{ridge}^*=\left(Z^TZ+\lambda I\right)^{-1}Z^Ty$

Lasso regression： $J(w) = \|w\|_1$
ElasticNet： $J(w) = \rho\|w\|_1 + (1-\rho)\|w\|^2$

3.3 梯度下降法 (GD)

详见6.1节。
岭回归： $\nabla_{\boldsymbol{w}} E_{ridge}(\boldsymbol{w}) = (\boldsymbol{Z}^T\boldsymbol{Z} + \lambda\boldsymbol{I})\boldsymbol{w} - Z^T\boldsymbol{y}$

3.4 重要思考题及解析

法方程组如何推导？
答： 记误差平方和 $E(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}[y(\boldsymbol{x}_n;\boldsymbol{w}) - y_n]^2 \Rightarrow w^* = \arg\min_{\boldsymbol{w}} E(\boldsymbol{w})$

记 $z = \varphi(x), z_n = \varphi(x_n) \Rightarrow E(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}\left(\boldsymbol{z}_n^T\boldsymbol{w} - y_n\right)^2$，再记 $Z = \begin{bmatrix} z_1^T \\ \vdots \\ z_N^T \end{bmatrix} \in \mathbb{R}^{N\times(K+1)}$，$\boldsymbol{y} = [y_1, \cdots, y_N]^T \in \mathbb{R}^N \Rightarrow E(w) = \frac{1}{2}\|Zw - y\|^2 = \frac{1}{2}w^TZ^TZw - y^TZw + \frac{1}{2}y^Ty$，令 $\nabla_{\boldsymbol{w}}E(w) = 0 \Rightarrow Z^TZw - Z^Ty = 0 \Rightarrow w^* = Z^\dagger y = (Z^TZ)^{-1}Z^Ty$

第 4 章 浅层/深度神经网络

重点：ReLU 表达的函数空间 (高维的话是分片线性函数)，节点，每一段斜率，ReLU、Sigmoid 基本性质

4.1 激活函数

ReLU(Rectified Linear Unit)： $\mathsf{ReLU}(z)=\begin{cases}0, & z<0,\\ 1, & z>0.\end{cases}$

性质：1. 置负为零; 2. 非负齐次性质 (non-negative homogeneity property)：$\mathsf{ReLU}(az)=a\mathsf{ReLU}(z)$ for $a\geq 0$

Sigmoid 函数： $\sigma(z)=\frac{1}{1+\exp(-z)}$，**性质：**取值在 (0,1) 之间，导数 $\sigma'(z)=\sigma(z)(1-\sigma(z))$

4.2 逼近定理

定理 4.1 (万能逼近定理 (Universal approximation theorem))

在有足够的隐藏单元的情况下，浅层神经网络可以以任意精度近似在 $\mathbb{R}^D$ 的紧子集上定义的任何连续函数。

定理 4.2 (万能逼近定理)

含有限数量的隐藏单元和激活函数的浅层神经网络可以以任意精度逼近 $\mathbb{R}^n$ 的紧致子集上的任何连续函数。

4.3 术语

输入层、输出层、隐藏层

宽度：每个隐藏层中的隐藏单元数；**深度：**隐藏层的数量

4.4 比较

浅层神经网络是两层深度神经网络的特例，深度神经网络可以表示更复杂的函数。

1 个输入、1 个输出和 $D>2$ 个隐藏单元的浅层神经网络最多可创建 $D+1$ 个线性区域，其有 $3D+1$ 个参数。1 个输入、1 个输出和 K 个隐藏层、每层有 $D>2$ 个隐藏单元的深度神经网络最多可创建 $(D+1)^K$ 个线性区域，其有 $3D+1+(K-1)D(D+1)$ 个参数。

4.5 重要思考题及解析

***Di* 个输入，*D* 个隐藏单元，D_o 个输出的浅层 *NN* 有多少参数？**

答：$(D_i+1)\times D+(D+1)\times D_o=D_i\times D+D+D\times D_o+D_o$

Zaslavsky(1975)：D 个超平面分割 D_i 维空间所得的线性区域的最大数量为 $\sum_{j=0}^{D_i}\binom{D}{j}$

D_i 个输入，D 个隐藏单元，D_o 个输出的深度为 K 的 *NN* 有多少参数？
答： $(D_i+1)\times D+(D+1)\times(K-1)\times D+(D+1)\times D_o$

第 5 章 损失函数

重点：构造 (比如先验正态分布，可以推导出均方误差损失；若标签 y_i 在 [0,1]，伯努利分布，二元交叉熵；分类的话，一般交叉熵；一般的分布，如何推导损失函数？)；似然函数 (最大似然估计，独立同分布)；这是一个完整体系，全面聚焦。

5.1 损失函数的构造方法

新观点：深度学习模型输出的是一个**概率分布**。将其视为在给定输入 $\boldsymbol{x}$ 的可能输出 $\boldsymbol{y}$ 上计算条件概率分布 $\mathsf{Pr}(\boldsymbol{y}|\boldsymbol{x})$。损失函数鼓励每个输出 $\boldsymbol{y_i}$ 在分布 $\Pr(\boldsymbol{y_i}|\boldsymbol{x_i})$ 下具有较高概率。因此模型输出为一个**条件概率分布**，想要输出单点选择分布的最大值即可。

举例：假设预测域为 $y \in \mathbb{R}$，可以选择正态分布，参数为 $\theta = \{\mu, \sigma^2\}$，可以使用模型 $f(x, \phi)$ 预测 μ，将 σ 设置为未知固定常数即可。

独立同分布假设： $\Pr(y_1, y_2, \cdots, y_I|x_1, x_2, \cdots, x_I) = \prod_{i=1}^{I} \Pr(y_i|x_i)$

最大似然准则：对于每个输入 $\boldsymbol{x_i}$，模型计算相应的分布参数 $\boldsymbol{\theta_i = f(x_i, \phi)}$，我们的目标是使得对于每个 $\boldsymbol{y_i}$ 都有高概率即最大化 $\mathsf{Pr}(\boldsymbol{y}|\boldsymbol{\theta})$。得到 I 个训练样本的联合概率为 $\prod_{i=1}^{I} \mathsf{Pr}(y_i|f(x_i, \phi))$，从而得到**最大似然准则**：

$$\hat{\phi} = \arg\max_{\boldsymbol{\phi}} \prod_{i=1}^{I} \Pr(\boldsymbol{y}_i|f(\boldsymbol{x}_i, \phi))$$

最大对数似然：为防止 $\Pr(y_i|f(x_i, \phi))$ 导致乘积结果更小，采用取对数的方式将累乘改为累加：

$$\hat{\phi} = \arg\max_{\boldsymbol{\phi}} \log \prod_{i=1}^{I} \mathsf{Pr}(\boldsymbol{y}_i|f(\boldsymbol{x}_i, \phi)) = \arg\max_{\boldsymbol{\phi}} \sum_{i=1}^{I} \log \mathsf{Pr}(\boldsymbol{y}_i|f(\boldsymbol{x}_i, \phi))$$

最小负对数似然：为从最小化损失的角度出发，需要将最大对数似然改为最小化问题：

$$\hat{\phi} = \arg\min_{\boldsymbol{\phi}} \left[-\sum_{i=1}^{I} \log \mathsf{Pr}(\boldsymbol{y}_i|f(\boldsymbol{x}_i, \phi)) \right] := \arg\min_{\boldsymbol{\phi}} L(\boldsymbol{\phi})$$

构造损失函数：

1. 选择预测域上合适的概率分布 $\mathsf{Pr}(y|\theta)$，其中 θ 为分布参数
2. 构建机器学习模型预测参数 θ 中的一个或多个
3. 在训练集 $\{x_i, y_i\}$ 上通过最小化负对数似然损失函数训练模型
4. 返回完整分布 $\mathsf{Pr}(y|f(x, \hat{\phi}))$ 或该分布最大值

5.2 回归问题损失函数

分布选择：单变量正态分布 $\Pr(y|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(y-\mu)^2}{2\sigma^2}\right]$($\sigma$ 未知固定常数)

模式设置：使用机器学习模型 $f(x, \phi)$ 预测 μ，将 σ 设置为未知固定常数。似然函数为 $\mathsf{Pr}(y|f(x, \phi), \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(y-f(x,\phi))^2}{2\sigma^2}\right]$

损失函数：负对数似然：$L(\phi) = -\sum_{i=1}^{I} \log \mathsf{Pr}(\boldsymbol{y}_i|f(\boldsymbol{x}_i, \phi)) = -\sum_{i=1}^{I} \log\left[\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(y-f(\boldsymbol{x},\phi))^2}{2\sigma^2}\right]\right]$.

最小二乘损失函数：由于 $\hat{\phi} = \arg\min_{\boldsymbol{\phi}} \left[-\sum_{i=1}^{I} \log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) + \frac{[y_i - f(\boldsymbol{x}_i,\phi)]^2}{2\sigma^2}\right] = \arg\min_{\boldsymbol{\phi}} \sum_{i=1}^{I} \frac{[y_i - f(x_i,\phi)]^2}{2\sigma^2}$，从而得到**最小二乘损失函数**：$L(\phi) = \frac{1}{2}\sum_{i=1}^{I}[y_i - f(x_i, \phi)]^2$。

方差估计：可以将方差也视为模型参数，从而最小化问题变为：

$$\hat{\phi}, \hat{\sigma}^2 = \arg\max_{\phi,\sigma^2} \left[-\sum_{i=1}^{I} \log \left[\frac{1}{\sqrt{2\pi\sigma^2}} \exp \left[-\frac{(y_i - f(\boldsymbol{x}_i, \phi))^2}{2\sigma^2} \right] \right] \right]$$

异方差回归：前面的模型方差固定，可以设置两个输出，分别为 μ 和 σ，从而得到异方差回归模型：

$$\hat{\phi} = \arg\min_{\phi} \left[-\sum_{i=1}^{I} \log \left[\frac{1}{\sqrt{2\pi f_2(\boldsymbol{x}, \phi)^2}} \right] - \frac{(y_i - f_1(\boldsymbol{x}_i, \phi))^2}{2 f_2(\boldsymbol{x}, \phi)^2} \right]$$

其中 $\sigma^2 = f_2(x, \phi)^2$。

5.3 分类问题损失函数

5.3.1 二分类

分布选择：Bernoulli 分布：$\Pr(y|\lambda) = (1-\lambda)^{(1-y)} \lambda^y = \begin{cases} 1-\lambda, & y=0 \\ \lambda, & y=1 \end{cases} \quad \lambda \in [0,1].$

模式设置：使用机器学习模型 $f(x,\phi)$ 预测 λ，为保证 $\lambda \in [0,1]$，使用 **sigmoid 函数** $\sigma(z) = \frac{1}{1+\exp(-z)}$，即 $\lambda = \sigma(f(x,\phi))$。似然函数为 $\mathsf{Pr}(y|f(x,\phi)) = [1 - \mathsf{sig}(f(x,\phi))]^{1-y} \mathsf{sig}(f(x,\phi))^y$.

损失函数：使用负对数似然得到**二分类交叉熵损失：**$L(\phi) = -\sum_{i=1}^{I} \left[(1-y_i) \log \left[1 - \mathrm{sig}(f(x_i,\phi)) \right] + y_i \log \mathrm{sig}(f(x_i,\phi)) \right]$

推断输出： $y = \begin{cases} 1, & \lambda > 0.5 \\ 0, & \text{其他} \end{cases}$

5.3.2 多分类

分布选择：类别分布，设有 K 个类别，则有 K 个参数，分别为 $\lambda_1, \cdots, \lambda_K$，满足 $0 \le \lambda_k \le 1, \sum_{i=1}^{K} \lambda_i = 1$ 且 $\mathsf{Pr}(y=k|x) = \lambda_k$。

模式设置：使用机器学习模型 $f(x,\phi)$ 预测 $\lambda_1, \cdots, \lambda_K$，输出为一 K 维向量。为保证 $\lambda_k \in [0,1]$，使用 **softmax 函数** $\mathrm{softmax}(z)_k = \frac{\exp(z_k)}{\sum_{j=1}^{K} \exp(z_j)}$，即 $\mathsf{Pr}(y=k|x) = \lambda_k = \mathrm{softmax}(f(x,\phi))_k$。

损失函数：使用负对数似然得到**多分类交叉熵损失：**

$$L(\phi) = -\sum_{i=1}^{I} \log \left[\mathrm{softmax}_{y_i} \left[f(x_i,\phi) \right] \right] = -\sum_{i=1}^{I} \left[f_{y_i}(\boldsymbol{x}_i,\phi) - \log \sum_{k'=1}^{K} \exp \left[f_{k'}(\boldsymbol{x}_i,\phi) \right] \right]$$

推断输出：取最大值 $\hat{y} = \arg\max_k \mathsf{Pr}(y=k|f(x,\hat{\phi}))$

5.4 多输出问题

当为多输出问题时可以定义一个多变量概率分布，通常将每个预测视为独立的，同样使用负对数似然定义损失函数：

$$L(\phi) = -\sum_{i=1}^{I} \log \mathsf{Pr}(y|f(x_i,\phi)) = -\sum_{i=1}^{I} \sum_{d} \log \mathsf{Pr}(y_{id}|f_d(x_i,\phi))$$

其中 y_{id} 表示第 i 个训练样本的第 d 个输出。

5.5 交叉熵损失

Kullback–Leibler(KL) 散度： $\mathrm{KL}(q\|p)=\int_{-\infty}^{\infty}q(z)\log q(z)dz-\int_{-\infty}^{\infty}q(z)\log p(z)dz$，可以定义距离。

交叉熵损失思想： 寻找参数 $\boldsymbol{\theta}$ 使数据 y 的经验分布 $q(y)$ 和模型分布 $\Pr(y|\boldsymbol{\theta})$ 间距离 (KL 散度定义) 最小化。

交叉熵 (cross entropy)： 经验分布 $q(y)$ 和模型分布 $\Pr(y|\boldsymbol{\theta})$ 间 KL 散度：

$$\hat{\boldsymbol{\theta}}=\arg\min_{\boldsymbol{\theta}}\left[\int_{-\infty}^{\infty}q(y)\log q(y)dy-\int_{-\infty}^{\infty}q(y)\log\Pr(y|\boldsymbol{\theta})dy\right]=\arg\min_{\boldsymbol{\theta}}\left[-\int_{-\infty}^{\infty}q(y)\log\mathsf{Pr}(y|\boldsymbol{\theta})dy\right]$$

定理 5.1

最小化交叉熵等价于最小化负对数似然。 ♡

证明

$$\begin{aligned}\hat{\boldsymbol{\theta}}&=\arg\min_{\boldsymbol{\theta}}\left[-\int_{-\infty}^{\infty}q(y)\log\mathsf{Pr}(y|\boldsymbol{\theta})dy\right]\\&=\arg\min_{\boldsymbol{\theta}}\left[-\int_{-\infty}^{\infty}\left[\frac{1}{I}\sum_{i=1}^{I}\delta(y-y_i)\right]\log\mathrm{Pr}(y|\boldsymbol{\theta})dy\right]\\&=\arg\min_{\boldsymbol{\theta}}\left[-\frac{1}{I}\sum_{i=1}^{I}\log\mathsf{Pr}(y_i|\boldsymbol{\theta})\right]=\arg\min_{\boldsymbol{\theta}}\left[-\sum_{i=1}^{I}\log\mathsf{Pr}(y_i|\boldsymbol{\theta})\right]\\(&=\arg\min_{\boldsymbol{\phi}}\left[-\sum_{i=1}^{I}\log\mathsf{Pr}(y_i|\boldsymbol{f}(\boldsymbol{x}_i,\boldsymbol{\phi}))\right])\end{aligned}$$

5.6 重要思考题及解析

第 6 章 优化算法

重点：全面掌握基本的优化算法，给一个示例改造成现代的算法；反向传播算法深刻理解 (给一个模型能够算出关于某个输入的梯度，公式)

6.1 梯度下降法 (Gradient Descent Method)

6.1.1 线性模型

局部极小值点：梯度为 0 的点，且沿任意方向移动时函数值都会增加

鞍点：梯度为 0 的点，但函数值在某些方向上增加，而在其他方向上减少

流程：1. 随机选取初值 w_0；2. 计算 $\frac{dE}{dw}|_{w=w_0}, w_1 \leftarrow w_0 - \eta \frac{dE}{dw}|_{w=w_0}$；3. 计算 $\frac{dE}{dw}\big|_{w=w_1}, w_2 \leftarrow w_1 - \eta \frac{dE}{dw}\big|_{w=w_1} \cdots$ 其中 η 为**学习率**，固定学习率效率低。

优点：实现简单，学习率调整得当的情况下表现良好。**缺点：**收敛速度较慢 (大模型/数据集)；对学习率选择敏感；容易陷入局部极小值点。

梯度计算：如 $L(\phi) = \sum_{i=1}^{I} l_i = \sum_{i=1}^{I} [f(x_i, \phi) - y_i]^2 = \sum_{i=1}^{I} (\phi_0 + \phi_1 x_i - y_i)^2$，则 $\frac{\partial L}{\partial \phi} = \frac{\partial}{\partial \phi} \sum_{i=1}^{I} l_i = \sum_{i=1}^{I} \frac{\partial l_i}{\partial \phi}$，其中 $\frac{\partial l_i}{\partial \phi} = \begin{bmatrix} \frac{\partial l_i}{\partial \phi_0} \\ \frac{\partial l_i}{\partial \phi_1} \end{bmatrix} = \begin{bmatrix} 2(\phi_0 + \phi_1 x_i - y_i) \\ 2x_i(\phi_0 + \phi_1 x_i - y_i) \end{bmatrix}$.

6.1.2 Gabor 模型

$y = f(x; \phi) = \sin(\phi_0 + 0.06 \cdot \phi_1 x) \cdot \exp(-\frac{(\phi_0 + 0.06 \cdot \phi_1 x)^2}{32})$，$\phi_0 \in \mathbb{R}$ 决定函数位置，$\phi_1 \in \mathbb{R}^+$ 决定函数形状。随着 ϕ_0 的增大, 函数向左移动; 随着 ϕ_1 的增大, 函数逐渐变窄

6.2 随机梯度下降 (Stochastic Gradient Descent)

批次 (batch)：将训练数据以**不放回**的方式抽取拆分为多个子集，每个子集包含一定数量的样本数据，称之为一个**批次 (batch)**。

机制：每次迭代仅使用一个批次的数据更新模型：$\phi_{t+1} = \phi_t - \alpha \cdot \sum_{i \in \mathcal{B}_t} \frac{\partial l_i(\phi_t)}{\partial \phi}$，其中 $\mathcal{B}_t$ 是当前批次样本索引集合，α 为学习率。

轮 (epoch)：模型遍历所有批次数据的一次迭代称之为一**轮 (epoch)**，注意每一轮的 batch 都随机划分。

优点：容易逃脱局部极小；大数据集上训练更快。**缺点：**对 batch size 选择可能敏感。

理解：单独批次中通过 GD 可降低该批次 Loss，但整体数据集上的 Loss 可能上升，因此可能逃脱局部极小。

6.3 带动量的随机梯度下降 (SGD with Momentum)

动量：所有历史梯度的指数衰减移动平均，由上一步动量和当前步梯度组合而成。当前步更新使用当前动量代替梯度。

Algorithm 1 SGD with Momentum

Input: 初始值 ϕ_0, 学习率 α, 动量衰减率 ρ

1: **初始化一阶动量：** $\boldsymbol{m} = 0$
2: **for** $t = 1$ to T **do**
3: **计算梯度：** $\boldsymbol{g_t} = \sum_{i \in \mathcal{B}_t} \nabla_{\boldsymbol{\phi}} l_i(\phi_t)$
4: **计算一阶动量：** $\boldsymbol{m_t} = \rho \boldsymbol{m_{t-1}} + (1-\rho)\boldsymbol{g}_t = \rho^t \boldsymbol{m_0} + (1-\rho) \sum_{i=1}^{t} \rho^{t-i} \boldsymbol{g_i}$
5: **更新参数：** $\phi_{t+1} = \phi_t - \alpha \boldsymbol{m_t}$
6: **end for**

优点：有效穿越损失景观中平坦部分；增加收敛速度。**缺点：**动量超参数需要调整。

理解：当前为 a，先沿 a 负梯度方向移动至 b，再沿着原动量方向移动至 c / 当前为 a，先沿原动量方向移动至 b，再沿 a 处负梯度移动至 c

6.4 Nesterov 加速动量法

Algorithm 2 Nesterov Accelerate Momentum

Input: 初始值 ϕ_0, 学习率 α, 动量衰减率 ρ

1: **初始化一阶动量：** $\boldsymbol{m} = 0$
2: **for** $t = 1$ to T **do**
3: **计算梯度：** $\boldsymbol{g_t} = \sum_{i \in \mathcal{B}_t} \nabla_{\boldsymbol{\phi}} l_i(\phi_t - \alpha \boldsymbol{m}_{t-1})$
4: **计算一阶动量：** $\boldsymbol{m_t} = \rho \boldsymbol{m_{t-1}} + (1-\rho)\boldsymbol{g}_t = \rho^t \boldsymbol{m_0} + (1-\rho) \sum_{i=1}^{t} \rho^{t-i} \boldsymbol{g_i}$
5: **更新参数：** $\phi_{t+1} = \phi_t - \alpha \boldsymbol{m_t}$
6: **end for**

理解：当前为 a，先沿原动量方向移动至 b，再沿 b 处负梯度移动至 c

6.5 梯度归一化 (Normalized Gradient)

Algorithm 3 Normalized Gradient

Input: 初始值 ϕ_0, 学习率 α, 常数 ϵ

1: **for** $t = 1$ to T **do**
2: **计算梯度：** $\boldsymbol{g_t} = \sum_{i \in \mathcal{B}_t} \nabla_{\boldsymbol{\phi}} l_i(\phi_t)$
3: **计算二阶动量：** $\boldsymbol{v_t} = \boldsymbol{g_t} \odot \boldsymbol{g_t}$
4: **更新参数：** $\phi_{t+1} = \phi_t - \frac{\alpha}{\sqrt{v_t + \epsilon}} \odot \boldsymbol{g_t}$
5: **end for**

注

1. $\frac{1}{\sqrt{v_t+\epsilon}} \odot \boldsymbol{g_t}$ 进行了梯度归一化操作，确保在梯度较大的方向上步长较小，而在梯度较小的方向上步长较大，从而提高优化的效率和稳定性。
2. ϵ 是一个很小的常数，用于数值稳定性，防止除零错误。

6.6 AdaGrad(Adaptive Gradient Algorithm)

Algorithm 4 AdaGrad(Adaptive Gradient Algorithm)

Input: 初始值 ϕ_0, 学习率 α, 常数 ϵ

1: **初始化二阶动量：** $\boldsymbol{v} = 0$
2: **for** $t = 1$ to T **do**
3: **计算梯度：** $= \boldsymbol{g_t} \sum_{i \in \mathcal{B}_t} \nabla_{\boldsymbol{\phi}} l_i(\phi_t)$
4: **计算二阶动量：** $\boldsymbol{v_t} = \boldsymbol{v_{t-1}} + \boldsymbol{g_t} \odot \boldsymbol{g_t}$
5: **更新参数：** $\phi_{t+1} = \phi_t - \frac{\alpha}{\sqrt{v_t + \epsilon}} \odot \boldsymbol{g_t}$
6: **end for**

优点：解决了 SGD 中学习率一直不变的问题。**缺点：**需手动指定初始学习率；初始梯度很大时，学习率会变得很小，导致训练过慢。

6.7 RMSprop

Algorithm 5 RMSprop

Input: 初始值 ϕ_0, 学习率 α, 常数 ϵ, 衰减率 ρ

1: **初始化二阶动量：** $\boldsymbol{v} = 0$
2: **for** $t = 1$ to T **do**
3: **计算梯度：** $\boldsymbol{g_t} = \sum_{i \in \mathcal{B}_t} \nabla_{\boldsymbol{\phi}} l_i(\phi_t)$
4: **计算二阶动量：** $\boldsymbol{v_t} = \rho \boldsymbol{v_{t-1}} + (1-\rho) \boldsymbol{g_t} \odot \boldsymbol{g_t}$
5: **更新参数：** $\phi_{t+1} = \phi_t - \frac{\alpha}{\sqrt{v_t + \epsilon}} \odot \boldsymbol{g_t}$
6: **end for**

优点：避免 AdaGrad 算法中学习率不断下降以至于过早衰减；**缺点：**仍需手动选择初始学习率

6.8 AdaDelta

多元函数问题 $\hat{x} = \arg\min_{\boldsymbol{x}} f(\boldsymbol{x})$ 牛顿法：$\boldsymbol{x_{t+1}} = \boldsymbol{x_t} - \boldsymbol{H_t}^{-1} \nabla f(\boldsymbol{x})$，其中 $\boldsymbol{H}$ 为 Hessian 矩阵。

对于 NN，$\hat{\boldsymbol{\phi}} = \arg\min_{\boldsymbol{\phi}} L(\boldsymbol{\phi})$，其牛顿法为 $\boldsymbol{\phi_{t+1}} = \boldsymbol{\phi_t} - \boldsymbol{H_t}^{-1} \boldsymbol{g_t}$。此时取 $\hat{\boldsymbol{H}}_{\boldsymbol{t}} = \text{diag}(\boldsymbol{H_t})$ 作为近似：$\boldsymbol{\phi_{t+1}} = \boldsymbol{\phi_t} - \hat{\boldsymbol{H}}_{\boldsymbol{t}}^{-1} \boldsymbol{g_t}$，即 $\phi_i^{t+1} = \phi_i^t - \frac{\frac{\partial L}{\partial \phi_i}}{\frac{\partial^2 L}{\partial \phi_i^2}}$，令 $\Delta \phi_i^t \approx \frac{\frac{\partial L}{\partial \phi_i}}{\frac{\partial^2 L}{\partial \phi_i^2}}$，则 $\frac{\partial^2 L}{\partial \phi_i^2} \approx \frac{\frac{\partial L}{\partial \phi_i}}{\Delta \phi_i^t}$，从而 $\phi_i^{t+1} = \phi_i^t - \frac{\Delta \phi_i}{\frac{\partial L}{\partial \phi_i}} g_i^t$，定义 $v_t = \rho v_{t-1} + (1-\rho) g_t \odot g_t, s_t = \rho s_{t-1} + (1-\rho) \Delta \phi_t \odot \Delta \phi_t$，得到近似：$\Delta \phi_i^t = \sqrt{s_{t-1} + \epsilon}$，$\frac{\partial L}{\partial \phi_i} = \sqrt{g_t + \epsilon}$，故 $\phi_i^{t+1} = \phi_i^t - \frac{\sqrt{s_{t-1} + \epsilon}}{\sqrt{v_t + \epsilon}} g_i^t$

Algorithm 6 AdaDelta

Input: 初始值 ϕ_0, 常数 ϵ, 衰减率 ρ

1: **初始化二阶动量：** $\boldsymbol{v} = 0$，累加变量 $\boldsymbol{s} = 0$
2: **for** $t = 1$ to T **do**
3: **计算梯度：** $\boldsymbol{g_t} = \sum_{i \in \mathcal{B}_t} \nabla_{\boldsymbol{\phi}} l_i(\phi_t)$
4: **计算二阶动量：** $\boldsymbol{v_t} = \rho \boldsymbol{v_{t-1}} + (1-\rho)\boldsymbol{g_t} \odot \boldsymbol{g_t}$
5: **差值更新：** $\Delta\phi_t = -\frac{\sqrt{s_{t-1}+\epsilon}}{\sqrt{v_t+\epsilon}} \odot g_t$
6: **累加更新：** $s_t = \rho s_{t-1} + (1-\rho)\Delta\phi_t \odot \Delta\phi_t$
7: **参数更新：** $\phi_{t+1} = \phi_t + \Delta\phi_t$
8: **end for**

6.9 Adam(Adaptive Moment Estimation)

Algorithm 7 Adam(Adaptive Moment Estimation)

Input: 初始值 ϕ_0, 常数 ϵ, 衰减率 β, ρ, 学习率 α

1: **初始化一、二阶动量：** $\boldsymbol{m} = 0, \boldsymbol{v} = 0$
2: **for** $t = 1$ to T **do**
3: **计算一阶动量：** $\boldsymbol{m_t} = \beta \boldsymbol{m_{t-1}} + (1-\beta)\boldsymbol{g_t}$
4: **计算二阶动量：** $\boldsymbol{v_t} = \rho \boldsymbol{v_{t-1}} + (1-\rho)\boldsymbol{g_t} \odot \boldsymbol{g_t}$
5: **调整动量：** $\hat{\boldsymbol{m}}_{\boldsymbol{t}} = \frac{\boldsymbol{m_t}}{1-\beta^t}, \hat{\boldsymbol{v}}_{\boldsymbol{t}} = \frac{\boldsymbol{v_t}}{1-\gamma^t}$
6: **参数更新：** $\phi_{t+1} = \phi_t - \frac{\alpha}{\sqrt{\hat{\boldsymbol{v}}_{\boldsymbol{t}}+\epsilon}} \odot \hat{\boldsymbol{m}}_{\boldsymbol{t}}$
7: **end for**

优点：收敛快；处理噪声优。**缺点：**更多的超参数调优。

6.10 超参数调优

网格搜索，随机搜索，贝叶斯搜索等

6.11 重要思考题及解析

第 7 章 卷积神经网络

重点：全面了解，基本操作，相关概念 **概念 卷积核 (convolution kernel)：**又称**滤波器 (filter)**，是一个小矩阵，用于提取图像的特征。

核大小 (kernel size)：对输入进行加权与求和的区域大小。如 $\omega = [\omega_1, \omega_2, \omega_3]^T$ 为大小为 3 的核 (一维)，输出 $z_i = \omega_1 x_{i-1} + \omega_2 x_i + \omega_3 x_{i+1}$。

填充 (padding)：处理第一个和最后一个输出的方式。有如下方式：

1. same padding：进行填充处理
   (a). zero padding：用 0 填充边缘
   (b). reflect padding：以矩阵边缘为对称轴，将矩阵中的元素对称的填充到外围
   (c). replicate padding：将矩阵的边缘复制并填充到矩阵的外围
   (d). circular padding：将输入矩阵从左到右，从上到下进行重复延伸
2. valid padding：不进行任何处理

步距 (stride)：卷积核每次滑动元素的个数。可压缩一部分信息，使得输出的尺寸小于输入的尺寸。

空洞卷积：将卷积核的某些权值设置为零，以保证在不增加权重的前提下增加核大小，从而更大的区域上进行加权求和。

空洞率 (dilation rate)：空洞卷积中零的数量。

卷积层：卷积运算 $\rightarrow$ 加偏置 (bias) $\rightarrow$ 激活函数。如一维时 $h_i = a(\beta + \omega_1 x_{i-1} + \omega_2 x_i + \omega_3 x_{i+1}) = a\left(\beta + \sum_{j=1}^{3} \omega_j x_{i+j-2}\right)$，二维时 $h_{ij} = a\left(\beta + \sum_{m=1}^{2}\sum_{n=1}^{3} \omega_{mn} x_{i+m-2,j+n-2}\right)$。

特征图 (feature map)/通道 (channel)：为避免丢失信息并行计算数个卷积，每个卷积产生的新隐藏变量称为特征图。设传入层 (incoming layer) 有 C_i 个通道，传出层 (outgoing layer) 有 C_o 个通道，卷积核大小为 $K \times K$，则权重矩阵 $\Omega \in \mathbb{R}^{C_i \times C_o \times K \times K}$，偏置 $\beta \in \mathbb{R}^{C_o}$。

感受野 (Receptive Fields)：指隐藏单元能看到的输入图像的区域。一般后续层中隐藏单元的感受野会增加。

下采样 (Downsampling)：通过池化操作 (如最大池化和平均池化) 减少特征图的尺寸 (但通道数不变)，以降低分辨率。

上采样 (Upsampling)：通过复制、最大反池化、双线性插值、转置卷积等操作增加特征图的尺寸，以增大特征图的尺寸。

改变通道数：使用 1×1 卷积核改变通道数，如 $\Omega \in \mathbb{R}^{C_i \times C_o \times 1 \times 1}$，即通道数由 C_i 变为 C_o。

7.1 重要思考题及解析

第 8 章 残差网络

重点： batch normalization，构造形式 (合理和不合理的)

8.1 概念

基本思想： 每一次训练目标是得到**残差函数** $f(x)=h(x)-x$，故 $h(x)=x+f(x)$，一个残差网络可以写为：

$$h_1=x+f_1(x,\phi_1)\to h_2=h_1+f_2(h_1,\phi_2)\to h_3=h_2+f_3(h_2,\phi_3)\to y=h_3+f_4(h_3,\phi_4)$$

其中每一个等式称为一个**残差块**。据此

$$\begin{aligned}y=x&+f_1(\boldsymbol{x})\\&+f_2(x+f_1(x))\\&+f_3(x+f_1(x)+f_2(x+f_1(x)))\\&+f_4(x+f_1(x)+f_2(x+f_1(x))+f_3(x+f_1(x)+f_2(x+f_1(x))))\end{aligned}$$

残差连接 (residual connection)/跳跃连接 (skip connection)： 残差网络的连接方式

合理的连接方式： 输入 1：x，输入 2：x →Linear→ReLU→Linear

8.2 梯度爆炸

残差网络不必担心梯度消失，因为存在一条路径，每一层都直接对网络输出做出贡献。

He 初始化： 将偏置初始化为 $\mathcal{N}(0,\sqrt{2/n})$，其中 n 为前一层隐藏单元的数量。此法保证了输入与输出预期方差不变。

由于残差网络将处理结果与输入相加作为输出，因此方差翻倍，随着残差块数量的增加，方差会呈指数级增长，可能产生梯度爆炸。解决方法之一为：使用 He 初始化再将每个残差块输组合乘以 $\frac{1}{\sqrt{2}}$，防止方差翻倍。另一个方式是**批归一化**。

批归一化 (Batch Normalization)： 在每个 batch$\mathcal{B}$ 上对每一层的输出进行归一化处理，将输入调整为均值为 0，方差为 1。流程为：

1. **计算**输入 h 的均值 $m_h=\frac{1}{|\mathcal{B}|}\sum\limits_{i\in\mathcal{B}}h_i$ 和标准差 $s_h=\sqrt{\frac{1}{|\mathcal{B}|}\sum\limits_{i\in\mathcal{B}}(h_i-m_h)^2}$
2. **归一化**：$\hat{h_i}\leftarrow\frac{h_i-m_h}{s_h+\epsilon}$，　$\forall i\in\mathcal{B}$，ϵ 很小以防止被零除
3. **缩放和平移**：$\tilde{h_i}\leftarrow\gamma\hat{h_i}+\beta$，　$\forall i\in\mathcal{B}$，其中 γ,β 为可学习参数

超参数数量： K 层、每层 D 个隐藏单元的标准 *NN* 中：KD 个 β，KD 个 γ；K 层、每层 C 个通道的 *CNN* 中：KC 个 β，KC 个 γ。

批归一化的优点：

1. **正向传播稳定：** 若初始化 $\gamma=1,\beta=0$，1. 则初始化时前向传播方差稳定 2. 虽然每一层可能添加非 1 方差源，但其随残差块线性增长
2. **学习率更高：** 可以使损失函数曲面及其梯度变化更平滑，提高学习率
3. **正则化：** 批归一化与批次相关，引入了噪声 (随机性)，可以提升模型泛化能力

为什么残差神经网络可行？ 1. 可以训练更深的网络 (但是存疑)；2. 残差网络的损失函数曲面在最小值附近更加平滑、更可预测，可能更具泛化能力。

8.3 常见残差架构

ResNet：每个残差块包含一个 BN 操作、一个 ReLU 激活和一个卷积层及其重复，然后再添加输入。**对图像分类很有效**。

DenseNet：每个残差块包含一个 BN 操作、一个 ReLU 激活和一个卷积层，再将输出附加至输入上，从而**增加输入通道数**。**在图像分类上很有效，同参数量下优于 ResNet**。可能因为其可以更灵活地利用早期层的信息。

U-Net：是一种编码器-解码器架构。最初用于分割医学图像，但也适用于其他任务。编码器部分是一个典型的卷积神经网络 (下采样)，解码器部分是一个反卷积神经网络 (上采样)。

Hourglass：沙漏网络，在跳跃连接中应用了更多的卷积层，并将结果添加回解码器。

8.4 重要思考题及解析

第 9 章 Attention is all you need

重点：深刻理解 (最后一节课的内容)，+self-attention，基本原理，基本推导，网络架构的搭配使用

9.1 注意力机制

注意力机制 (Attention Mechanism) 是深度学习中的一种技术，最早应用于神经机器翻译任务。它的核心思想是通过赋予每个输入元素不同的重要性权重，来决定哪些部分更值得关注。

假设有一个输入序列 $X = [x_1, x_2, ..., x_n]$ 和一个查询 q，注意力机制的目标是计算每个输入元素 x_i 的权重 w_i，并根据这些权重计算加权和 (即注意力输出)。计算方式如下：

计算相似度：$e_i = \text{score}(q, x_i)$，常见的相似度函数包括点积 (Dot-Product)、可学习的线性变换 (Learnable Linear Transformation) 和其他函数。

计算注意力权重：$\alpha_i = \frac{\exp(e_i)}{\sum_{j=1}^{n} \exp(e_j)}$

计算加权和：$\text{output} = \sum_{i=1}^{n} \alpha_i x_i$

9.2 重要思考题及解析