

Task-1

1.MLP 及其结构：

MLP 是一种最基础，最经典的神经网络结构；

一个典型的 MLP 包括：

输入层：接收原始输入特征；

隐藏层：提取中间特征，每层包含多个神经元；

输出层：输出最终结果，如类别，概率等；

关键计算公式如下：

对于第 l 层的神经元，有：

$$z^{[l]} = W^{[l]}a^{[l-1]} + b^{[l]}$$

$$a^{[l]} = \sigma(z^{[l]})$$

符号	含义
$W^{[l]}$	第 l 层的权重矩阵
$b^{[l]}$	偏置向量
$a^{[l]}$	激活值，即当前层输出
σ	激活函数，如 ReLU, Sigmoid

2.数据在神经网络中的角色：

数据被视作神经网络实现“从数据到知识映射”的核心载体；角色可从统计学习理论和表示学习视角拆解为以下维度：

A：参数优化的监督信号载体：

神经网络的参数学习过程本质上是基于数据的经验风险最小化。训练数据通过输入特征 x 与标签 y 的映射关系， $D = \{(xi, yi)\}$ ，为损失函数 $\mathcal{L}(\hat{y}, y)$ 提供计算依据，而损失函数的梯度则通过反向传播算法指导参数 θ 的迭代更新。此时，数据的标签信息构成了参数从先验分布 $p(\theta)$ 向后验分布 $p(\theta|D)$ 的关键约束。

B：范化边界的分布准则：

根据 VC 维理论，模型的范化误差可分解为经验误差与置信区间之和，其中置信区间受训练数据分布 \hat{P} 与真实数据 P 分布的差异影响。若 \hat{P} 能近似 P ，模型可学到 P 的本质规

律，若 \hat{P} 存在偏差，则可能导致过拟合，对 P 的范化误差显著升高；

C：层级特征学习的隐性约束：

深度神经网络的表示学习能力依赖于数据的内部结构。在无监督场景中，数据的统计特性引导网络学习层级化特征。低层网络捕捉局部模式，高层网络聚合抽象概念。缺失关键结构的数据集会导致数据表征的信息缺失；

D：模型评估的独立基准：

模型性能的无偏评估需要基于独立测试集，核心要求测试数据未参与任何参数优化过程。

2.1 数据集的 split：

(1) 训练集：

用于参数优化的样本集合，常占原始数据的 60%-70%；

基于 ERM 原则，训练集需尽可能覆盖真实分布 P 的支撑集，以确保参数可尽可能收敛到全局最优解（近似）；

(2) 验证集：

用于超参数调优与早停的样本集合，占比通常 10%-20%；

超参数优化本质是对模型假设空间的探索，验证集性能作为搜索准则，可避免超参数过度拟合训练集；

(3) 测试集：

用于最终泛化能力评估的独立样本集合，通常占比 10%-20%；

(4) 特殊场景划分：

小数据集：采用 k -折交叉验证，将数据分 k 个子集，轮流以 $k-1$ 个子集为训练集，1 个子集为验证集，最终以平均性能为评估结果；

时序数据：采用时间序列划分；

2.2 数据集的处理：

(1) 数据预处理：

数据清洗：针对缺失值采用极大似然估计或多重插补；针对异常值采用 Z-score 过滤，以消除噪声对参数估计干扰；

标准化/归一化：通过 Z-score 标准化或 Min-Max 归一化统一特征尺度；

(2) 数据增强：

通过随机变换生成符合真实分布的虚拟样本，拓展训练集，但增强操作需满足“标签不变性”；

2.3 噪声，特征，标签：

(1) 噪声：

在神经网络的语境中，噪声是指数据中与任务目标无关，干扰模型学习真实模式的无用信息；

例如在图像分类任务中，图像采集时的传感器误差，环境随机干扰，噪声会增加模型学习难度，可能导致过拟合；

(2) 特征：

特征是从原始数据中提取，用于描述样本关键属性的表示。在模式识别与深度学习中，特征需要具备代表性与区分性。

(3) 标签：

标签是与样本关联，表示任务目标输出的信息。在监督学习范式中，标签是模型训练的“指导信号”，用于定义损失函数。

2.4 Batch：

(1) Batch Size：

是训练时一次性输入模型的样本数量。例如在随机梯度下降及其变种优化过程中，模型并非逐一样本更新参数，而是将若干样本组成 Batch，基于 Batch 内样本计算的梯度更新参数。

(2) Batch 加速运算本质：

从计算硬件与算法并行性分析，堆叠成 Batch 可提升运算速度的核心是利用了计算资源的并行性。现代计算设备具备单指令多数据架构，可同时对 Batch 内多个样本的矩阵运算进行并行处理。Batch 计算可降低内存访问频率，进一步提升计算效率；

3.神经元：

3.1 计算流程：

生物神经元经过树突接收信号，经细胞体整合，由轴突输出到其余神经元；而这一过程被抽象为“加权求和-非线性变化”的数学过程；

流程可形式化为：

$$a = \sum_{i=1}^n w_i x_i + b, \quad z = f(a)$$

其中：

- x_i 是输入信号（对应生物神经元接收的突触输入），
- w_i 是权重（模拟突触连接强度，刻画输入与神经元间的关联程度），
- b 是偏置（模拟生物神经元的兴奋阈值，调节神经元激活的难易程度），
- a 是加权和（整合所有输入信号的线性组合），
- $f(\cdot)$ 是激活函数（模拟生物神经元的非线性放电特性，赋予神经元非线性表达能力），
- z 是神经元输出（传递至下游神经元或作为模型最终输出）。

从计算理论看，单个神经元本质是**线性分类器**（当激活函数为阶跃函数时，可实现简单模式划分），但通过多层神经元的堆叠与非线性激活，可逼近任意复杂函数（万能逼近定理，Hornik et al., 1989）。

3.2 在神经网络中的角色：

（1）输入层神经元：

直接映射原始数据特征，仅执行输入传递；

（2）隐藏层神经元：

通过学习到的权重 w_i 和偏置 b ，对输入特征进行线性变化和非线性编码，将低维原始特征映射到高维特征空间

（3）输出层神经元：

输出任务相关要求，其激活函数通常适配任务需求；

神经元通过权重 w_i 实现特征选择与加权聚合，通过激活函数，引入非线性决策边界；

4. 激活函数：

神经网络理论中，激活函数是赋予模型非线性表达能力的核心组件，其本质是对神经元加权和输出的非线性变换。核心作用可归纳为以下两点：

引入非线性：突破线性模型的表达限制，使网络能逼近任何复杂函数；

控制信息流动：通过非线性阈值（如 ReLU 的 0 值截断）实现特征的筛选和稀疏化；

常见的激活函数：

(1) Sigmoid 函数

形式：

$$f(a) = \frac{1}{1+e^{-a}}$$

特性：

- 输出映射到 $(0, 1)$ 区间，可模拟概率输出或生物神经元的连续放电率（Rumelhart et al., 1986）；
- 存在**梯度消失问题**（当 $|a| \gg 0$ 时，导数 $f'(a) = f(a)(1 - f(a)) \approx 0$ ），导致深层网络训练困难（Hochreiter et al., 2001）；
- 需注意输出非零均值（均值约为 0.5），可能导致梯度更新的偏差（需配合 Batch Normalization 缓解，Ioffe & Szegedy, 2015）。

(2) Tanh 函数

形式：

$$f(a) = \tanh(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}}$$

特性：

- 输出映射到 $(-1, 1)$ 区间，解决了 Sigmoid 非零均值问题（均值为 0，利于梯度对称更新）；
- 仍存在**梯度消失问题**（导数 $f'(a) = 1 - \tanh^2(a)$ ，当 $|a| \gg 0$ 时趋近于 0）；
- 常用于循环神经网络（RNN）的早期设计（如 LSTM 之前的 RNN 单元，Graves, 2012）。

(3) ReLU (Rectified Linear Unit)

形式：

$$f(a) = \max(0, a)$$

特性：

- **非饱和性**（当 $a > 0$ 时，导数 $f'(a) = 1$ ，避免梯度消失），大幅提升深层网络训练效率（Krizhevsky et al., 2012）；
- **稀疏激活**（当 $a \leq 0$ 时输出 0，模拟生物神经元的稀疏放电，减少计算量与过拟合风险）；
- 存在**死亡 ReLU 问题**（部分神经元因初始权重或梯度更新，长期输出 0，无法参与学习，可通过 Leaky ReLU 等变种缓解，Maas et al., 2013）。

(4) Leaky ReLU

形式：

$$f(a) = \begin{cases} a & a > 0 \\ \alpha a & a \leq 0 \end{cases} \quad (\alpha \text{ 为小常数, 如 } 0.01)$$

特性：

- 解决死亡 ReLU 问题 ($a \leq 0$ 时仍有小梯度 α , 维持神经元活性)；
- 超参数 α 需手动或自适应调整 (如 PReLU 让 α 可学习, He et al., 2015)。

(5) Softmax (特殊场景激活函数)

形式 (多分类场景)：

$$f(a_i) = \frac{e^{a_i}}{\sum_j e^{a_j}} \quad (i, j \text{ 为类别索引})$$

特性：

- 输出为概率分布 (和为 1), 专为多分类任务设计, 将线性输出转换为类别概率；
- 本质是 "归一化的指数函数", 可视为 Sigmoid 函数在多分类场景的推广 (Goodfellow et al., 2016)。

模型非线性表达能力：

神经网络的非线性表达能力指模型通过多层非线性变换, 逼近任意复杂函数的能力, 核心依赖于激活函数的非线性与网络层级的堆叠；

可从以下方面来理解：

(1) 万能逼近定理的支撑

Hornik et al. (1989) 证明：含至少一个隐藏层、采用非线性激活函数的前馈网络, 可在紧集上以任意精度逼近连续函数。这一结论的关键前提是激活函数的非线性 —— 若使用线性激活函数, 多层网络等价于单层线性模型 (所有层的线性变换可合并为 $z = W_{\text{全局}}x + b_{\text{全局}}$), 无法突破线性映射的限制。

(2) 多层非线性变换的 "特征空间扩展"

激活函数通过非线性变换, 将输入特征映射到更高维 (或更适合任务) 的特征空间。例如：

- 单层网络的线性变换只能学习 "直线 / 平面" 式的决策边界；
- 多层网络中, 每层激活函数对特征进行非线性扭曲, 最终可拟合 "任意形状" 的决策边界 (如异或问题需至少一层非线性激活才能解决, Minsky & Papert, 1969)。

这种 "特征空间的逐层扩展与扭曲", 使网络能捕捉数据的复杂模式 (如图像的纹理、文本的语义关联), 而激活函数是实现这一过程的 "非线性引擎" (Bengio et al., 2013)。

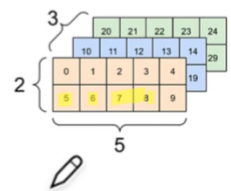
5.计算图：

5.1 计算图的基本数据结构：张量

基于计算图的AI框架：基本组成

基本数据结构：Tensor 张量

- 高维数组，对标量，向量，矩阵的推广
- Tensor形状 (Shape) : [3, 2, 5]
- 元素基本数据类型 (Data Type) : int, float, string, etc.



优点：

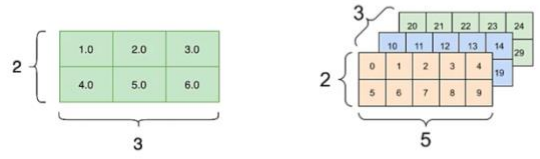
- 后端自动推断并完成元素逻辑存储空间向物理存储空间的映射
- 基本运算类型作为整体数据进行批量操作，适合单指令多数据（SIMD）并行加速

基于计算图的AI框架

基于计算图 (DAG) 的计算框架

基本数据结构：Tensor 张量

- Tensor形状：[2, 3, 4, 5]
- 元素类型：int, float, string, etc.



基本运算单元：Operator 算子

- 由最基本的代数算子组成
- 根据深度学习结构组成复杂算子
- N个输入Tensor，M个输出Tensor

Add	Log	While
Sub	MatMul	Merge
Mul	Conv	BroadCast
Div	BatchNorm	Reduce
Relu	Loss	Map
Floor	Sigmoid

5.2 计算图的流程图：

基于计算图的AI框架

基于数据流图 (DAG) 的计算框架

DAG 表示计算逻辑和状态

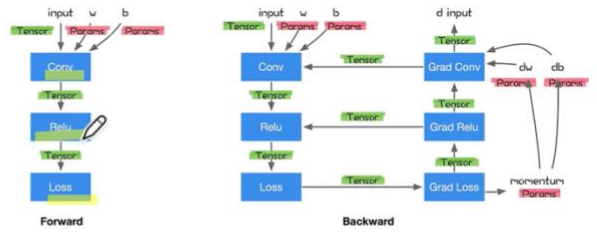
- 节点代表 Operator
- 边代表 Tensor

特殊的操作

- 如：For/While 等构建控制流

特殊的边

- 如：控制边表示节点间依赖



5.3 计算图的定义：

计算图是由运算结点和数据边组成的有向无环图；

其中运算结点表示基本的数学操作，如加法，乘法，激活函数，卷积等；

数据边表示运算间传递的张量或标量数据，对应函数的输入或者输出；

从自动微分理论来看，计算图是反向传播的实现基础，前向传播沿图的边传递数据，执行结点计算，反向传播沿边的反向计算梯度，通过链式法则实现参数更新；

5.4 与数据结构/离散数学中“图”的核心区别：

(1) 核心目标上：

计算图描述“数值计算流程”，支撑自动微分与模型训练；

而数/离中图主要描述“元素关系”，连接，依赖，拓扑等；

(2) 结点语义：

计算图中表示数学运算；

“图”中表示抽象元素（顶点，状态，实体）

(3) 边语义：

计算图边表示数据流动；

“图”中表示关系，连接；

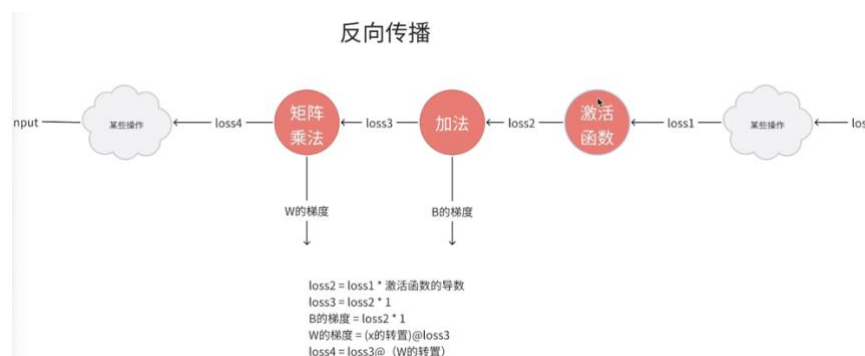
(4) 典型应用上：

计算图主要应用于深度学习模型的前向，反向传播实现；

“图”中应用于最短路径，社交网络分析，拓扑排序等；

5.5 计算图的构建：

计算图的构建需要先将符合运算拆解成原子操作，随后按计算顺序连接节点和边；例如：



6.MLP 参数相关

6.1MLP 参数量计算：

多层感知机的参数包含权重和偏置，参数数量需要逐层分析：

(1) 单层全连接层的参数计算：

对于相邻两层（第 l 层与第 $l + 1$ 层），若第 l 层有 n_l 个神经元，第 $l + 1$ 层有 n_{l+1} 个神经元，则：

- **权重参数数量：** $n_l \times n_{l+1}$ （每个第 l 层神经元与第 $l + 1$ 层神经元有一条权重连接）；
- **偏置参数数量：** n_{l+1} （第 $l + 1$ 层每个神经元对应一个偏置）。

因此，单层全连接层的总参数数量为：

$$\text{Params}_{\text{layer}} = n_l \times n_{l+1} + n_{l+1} = n_{l+1}(n_l + 1)$$

(2) 多层感知机的总参数计算：

对于包含 L 个隐藏层的 MLP（总层数为 $L + 2$ ，含输入层、输出层），输入层维度 d_{in} ，隐藏层维度依次为 d_1, d_2, \dots, d_L ，输出层维度 d_{out} ，则总参数数量为各层参数之和：

$$\text{Total Params} = \sum_{l=0}^L [d_{l+1}(d_l + 1)]$$

其中， $d_0 = d_{\text{in}}$ （输入层维度）， $d_{L+1} = d_{\text{out}}$ （输出层维度）。

示例：假设 MLP 结构为“输入层 784 → 隐藏层 128 → 输出层 10”（用于 MNIST 分类），则：

- 输入层→隐藏层： $128 \times (784 + 1) = 128 \times 785 = 100,480$
- 隐藏层→输出层： $10 \times (128 + 1) = 10 \times 129 = 1,290$
- 总参数： $100,480 + 1,290 = 101,770$

6.2 超参数定义：

超参数是在模型训练前需要手动设置，无法通过梯度下降学习的网络配置参数；超参数直接决定模型结构，训练流程与优化策略，对模型的泛化能力和收敛效率有显著影响；

6.3MLP 超参数及应用：

多层感知机的超参数可分为结构超参数和训练超参数，具体如下：

(1) 结构超参数：

隐藏层数量：决定网络的深度。深层网络可学习更抽象的特征，但容易引发过拟合和梯度消失；浅层网络计算高效，但表达能力有限；

每层神经元数：决定网络的宽度；





输入/输出层维度：由系统任务决定；

(2) 训练超参数：

- **学习率 (Learning Rate, η)**：控制参数更新的步长 ($\theta \leftarrow \theta - \eta \cdot \nabla \mathcal{L}$)。过大导致震荡不收敛，过小导致训练缓慢 (Bottou, 2010)。
- **批量大小 (Batch Size)**：训练时单次输入的样本数。影响梯度估计的稳定性与计算并行性 (Iandola et al., 2016)。
- **迭代次数 / 轮数 (Epochs)**：数据集完整遍历的次数。过少导致欠拟合，过多导致过拟合 (需结合早停法, Prechelt, 1998)。
- **激活函数 (Activation Function)**：如 ReLU、Sigmoid 等，决定神经元的非线性变换方式 (影响特征表达能力, Goodfellow et al., 2016)。
- **正则化参数 (Regularization Parameter)**：如 L_2 正则化的权重衰减系数 λ ，控制模型复杂度以缓解过拟合 (Hastie et al., 2009)。

7. 隐藏层含义：

隐藏层由一个或多个层组成，每个隐藏层包含若干个神经元，每个神经元接收来自上一层的输入信号，进行加权求和，并通过激活函数生成输出信号，该输出信号又作为下一层的输入。

- **主要功能：**
 - **特征提取** ：隐藏层可以将输入数据映射到更高维度的特征空间中，通过学习和提取输入数据的特征，帮助网络更好地理解数据。每个隐藏层都可以学习到不同层次的抽象特征，随着隐藏层数量的增加，网络可以逐渐学习到更加抽象和复杂的特征表示。
 - **非线性建模** ：隐藏层通过引入非线性激活函数，如 ReLU、Sigmoid 等，帮助神经网络对非线性关系进行建模。线性模型只能处理线性可分的问题，而隐藏层的非线性变换使得神经网络能够逼近任意非线性函数。
 - **学习复杂映射** ：神经网络的隐藏层使得模型能够学习输入和输出之间的复杂映射关系。随着隐藏层的增加，模型可以学习到更复杂的模式和关系，从而提高模型的预测能力。
 - **解决多类别分类问题** ：对于多类别分类问题，隐藏层可以帮助网络学习到类别之间的相关性和区分性。隐藏层的神经元可以捕捉到不同类别之间的隐含关系，从而提高分类性能。

而隐藏层之所以被称为“隐藏层”，是因为其位于神经网络的输入层和输出层间，不能直接从系统的输入和输出中观察到，用户只能看到输入的数据和得到的输出结果，无法知晓隐藏层内部的具体运算和状态，因此将其命名为隐藏层；

8. 损失函数

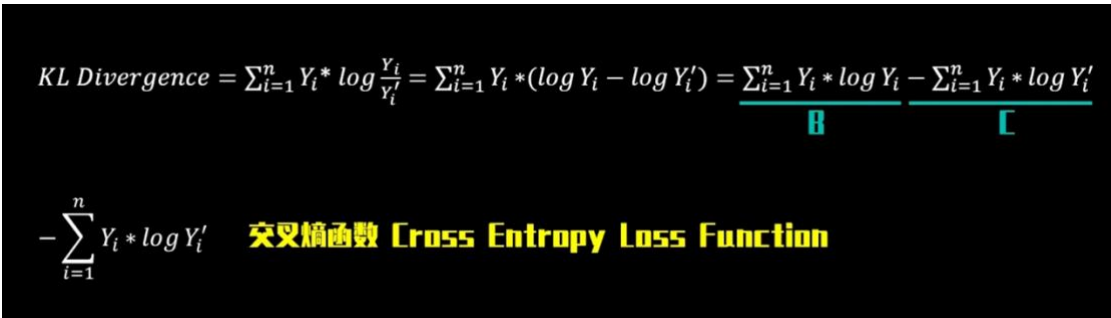
损失函数是衡量模型预测与真实标签差异的核心指标，是模型预测 \hat{y} 与真实标签 y 之间差异的量化函数。损失函数是经验风险的基本组成单元，经验风险可表示为：

$$\mathcal{R}_{\text{emp}}(\theta) = \frac{1}{N} \sum_{i=1}^N \mathcal{L}(f_{\theta}(\mathbf{x}_i), y_i)$$

其中 θ 为模型参数， $f_{\theta}(\mathbf{x})$ 为模型的预测函数， N 为样本数量。损失函数的设计需满足**非负性**（ $\mathcal{L}(\hat{y}, y) \geq 0$ ）与**一致性**（当 $\hat{y} \rightarrow y$ 时， $\mathcal{L}(\hat{y}, y) \rightarrow 0$ ）（Vapnik, 1998）。

不同任务下损失函数分类：

(1) 多分类任务下采用交叉熵损失（等价于 KL 散度的期望）


$$KL \text{ Divergence} = \sum_{i=1}^n Y_i * \log \frac{Y_i}{Y'_i} = \sum_{i=1}^n Y_i * (\log Y_i - \log Y'_i) = \underbrace{\sum_{i=1}^n Y_i * \log Y_i}_B - \underbrace{\sum_{i=1}^n Y_i * \log Y'_i}_C$$
$$-\sum_{i=1}^n Y_i * \log Y'_i \quad \text{交叉熵函数 Cross Entropy Loss Function}$$

(2) 二分类任务采用二元交叉熵损失：

适用于二分类任务（如垃圾邮件检测、疾病诊断），是交叉熵损失的特例（ $K = 2$ ）：

$$\mathcal{L}_{\text{BCE}} = -\frac{1}{N} \sum_{i=1}^N [y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)]$$

其中 $y_i \in \{0, 1\}$ 为真实标签， $\hat{y}_i \in (0, 1)$ 为模型预测为正类的概率（通常由 Sigmoid 函数输出）。

(3) 回归任务下采用均方误差和平均绝对误差：

(1) 均方误差（Mean Squared Error, MSE）

数学形式为：

$$\mathcal{L}_{\text{MSE}} = \frac{1}{N} \sum_{i=1}^N (\hat{y}_i - y_i)^2$$

理论依据：MSE 假设预测误差服从高斯分布（正态分布），其损失函数等价于极大似然估计（MLE）的负对数似然（Bishop, 2006）。当误差为独立同分布的高斯噪声时，MSE 是最优损失函数。

(2) 平均绝对误差（Mean Absolute Error, MAE）

数学形式为：

$$\mathcal{L}_{\text{MAE}} = \frac{1}{N} \sum_{i=1}^N |\hat{y}_i - y_i|$$

理论依据：MAE 假设误差服从拉普拉斯分布（Laplace Distribution），对异常值（Outlier）更鲁棒（Hastie et al., 2009）。当数据中存在噪声导致的极端值时，MAE 的梯度更稳定（避免 MSE 因大误差导致的梯度爆炸）。

(4) 用于生成任务中的对抗损失：

在生成对抗网络（GAN）中，生成器 G 与判别器 D 的损失函数为：

$$\mathcal{L}_{\text{GAN}} = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} [\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}} [\log(1 - D(G(\mathbf{z})))]$$

其中 $D(\mathbf{x})$ 为判别器判断真实样本为“真”的概率， $G(\mathbf{z})$ 为生成器生成的假样本。

(5) 用于生成任务中的重构损失：

在变分自编码器（VAE）中，重构损失通常采用 MSE 或 BCE（根据数据类型）：

$$\mathcal{L}_{\text{VAE}} = \mathbb{E}_{q(\mathbf{z}|\mathbf{x})} [\log p(\mathbf{x}|\mathbf{z})] - \text{KL}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))$$

其中 $\log p(\mathbf{x}|\mathbf{z})$ 为重构损失（衡量生成样本与真实样本的差异），KL项为正则化项（约束隐变量分布接近先验分布）（Kingma & Welling, 2014）。

9.前向传播/梯度/反向传播/学习率/优化器：

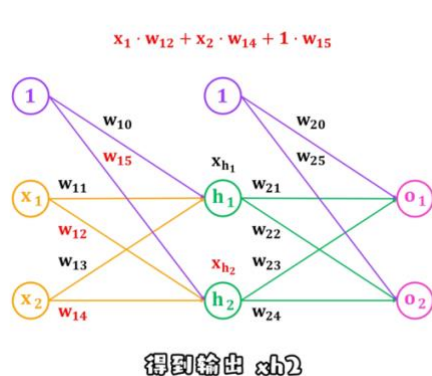
9.1 前向传播：

前向传播是神经网络从输入层到输出层的信号传递过程，其本质是逐层执行线性变换与非线性激活的计算流程，对于 L 层神经网络，第 l 层的前向传播可形式化为：

$$\begin{aligned} \mathbf{z}^{(l)} &= \mathbf{W}^{(l)} \mathbf{h}^{(l-1)} + \mathbf{b}^{(l)} \\ \mathbf{h}^{(l)} &= \sigma^{(l)}(\mathbf{z}^{(l)}) \end{aligned}$$

其中：

- $\mathbf{h}^{(l-1)}$ 为第 $l-1$ 层的输出（输入层 $\mathbf{h}^{(0)} = \mathbf{x}$ 为原始输入），
- $\mathbf{W}^{(l)}, \mathbf{b}^{(l)}$ 为第 l 层的权重矩阵与偏置向量，
- $\mathbf{z}^{(l)}$ 为第 l 层的线性变换结果（加权和），
- $\sigma^{(l)}(\cdot)$ 为第 l 层的激活函数（如 ReLU、Sigmoid），
- $\mathbf{h}^{(l)}$ 为第 l 层的输出（作为下一层的输入）。



以某前向传播流程为例：

9.2 梯度：

梯度是损失函数关于模型参数的偏导数向量；

$$\nabla_{\theta} \mathcal{L} = \left(\frac{\partial \mathcal{L}}{\partial \theta_1}, \frac{\partial \mathcal{L}}{\partial \theta_2}, \dots, \frac{\partial \mathcal{L}}{\partial \theta_D} \right)^{\top}$$

其中 D 为参数总数。在神经网络中，梯度的核心作用是**指示参数更新方向**：沿梯度的反方向（负梯度）更新参数，可使损失函数 \mathcal{L} 减小（Bishop, 2006）。

9.3 学习率：

学习率是控制参数更新步长的超参数，参数更新公式为：

$$\theta_{t+1} = \theta_t - \eta \cdot \nabla_{\theta} \mathcal{L}(\theta_t)$$

其中 θ_t 为 t 时刻的参数值。学习率的选择需平衡**收敛速度与稳定性**：

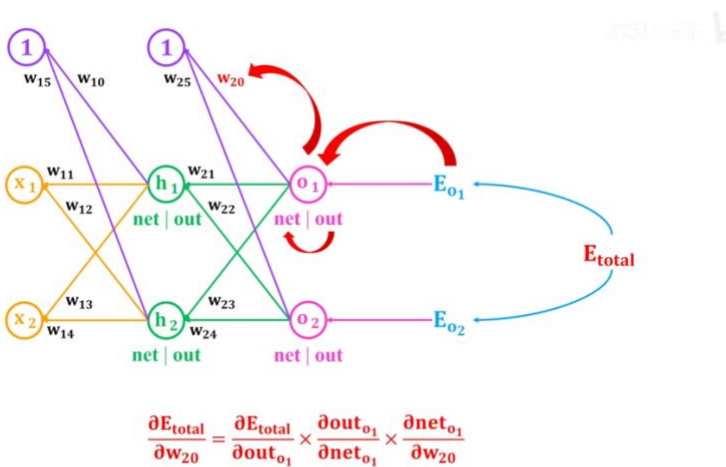
如果 η 过大，可能导致参数震荡（无法收敛到最优解）；

如果 η 过小，可能导致收敛缓慢；

9.4 反向传播：

反向传播是计算损失函数关于模型参数梯度的核心算法，利用链式法则，从输出层开始，逐层计算损失函数关于各层参数梯度；

示例如下所示：



反向传播流程：

(1) 输出层梯度计算：

损失函数关于输出层 $\mathbf{z}^{(L)}$ 的梯度为：

$$\delta^{(L)} = \nabla_{\mathbf{z}^{(L)}} \mathcal{L} = \frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(L)}} \odot \sigma^{(L)'}(\mathbf{z}^{(L)})$$

其中 $\frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(L)}}$ 为损失函数关于输出 $\mathbf{h}^{(L)}$ 的梯度， $\sigma^{(L)' }(\cdot)$ 为输出层激活函数的导数（如 Softmax 的导数）。

(2) 隐藏层梯度回传：

第 l 层 ($l < L$) 的梯度 $\delta^{(l)}$ 通过上一层梯度 $\delta^{(l+1)}$ 回传：

$$\delta^{(l)} = \left(\mathbf{W}^{(l+1)\top} \delta^{(l+1)} \right) \odot \sigma^{(l)'}(\mathbf{z}^{(l)})$$

(3) 参数梯度计算：

利用 $\delta^{(l)}$ 计算第 l 层的权重与偏置梯度：

$$\nabla_{\mathbf{W}^{(l)}} \mathcal{L} = \delta^{(l)} \mathbf{h}^{(l-1)\top}, \quad \nabla_{\mathbf{b}^{(l)}} \mathcal{L} = \delta^{(l)}$$

9.5 常见优化器：

(1) 梯度下降法：

基本思想：先设定一个学习率 η ，参数沿梯度的反方向移动。假设需要更新的参数为 ω ，梯度为 g ，则更新策略可表示为 $\omega = \omega - \eta * g$ ；

梯度下降有三种不同的形式：

BGD：批量梯度下降，每次参数更新使用所有样本；

SGD：随机梯度下降，每次参数更新仅使用一个样本；

MBGD：小批量梯度下降，每次参数更新使用小部分样本；

优缺点：

优点：算法简洁，当学习率取值恰当时，可收敛到全局或者局部最优解；

缺点：对超参数学习率较为敏感，过小收敛速度过慢，过大越过极值点；且当在较平坦区域，梯度变化不明显，优化算法可能在误判，在还未进入极值点就提前结束迭代了；有时候会被卡在鞍点位置；

某一维梯度下降法案例：

1.1 一维梯度下降法

我们以目标函数（损失函数） $f(x) = x^2$ 为例来看一看梯度下降是如何工作的（这里 x 为参数）。

迭代方法为：

$$x \leftarrow x - \eta * g = x - \eta * \frac{\partial loss}{\partial x}$$

虽然我们知道最小化 $f(x)$ 的解为 $x = 0$ ，这里依然使用如下代码来观察 x 是如何迭代的

这里 x 为模型参数，使用 $x = 10$ 作为初始值，并设学习率 $\eta = 0.2$ ，使用梯度下降法对 x 迭代10次

```
import numpy as np
import matplotlib.pyplot as plt

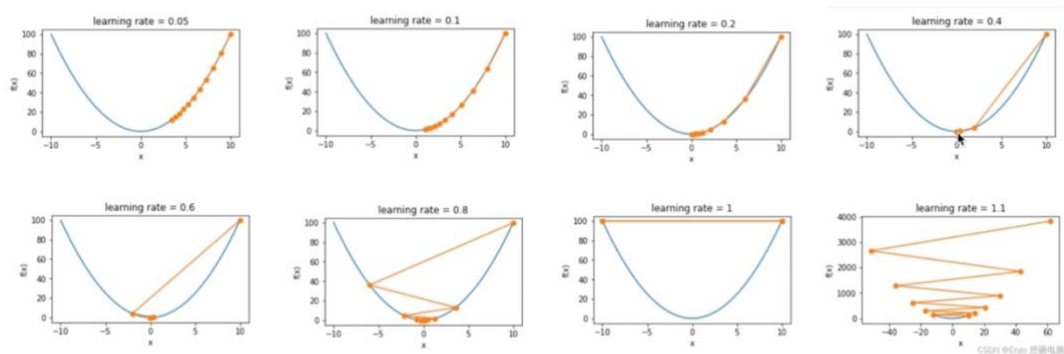
x = 10
lr = 0.2
result = [x]

for i in range(10):
    x -= lr * 2 * x
    result.append(x)

f_line = np.arange(-10, 10, 0.1)
plt.plot(f_line, [x * x for x in f_line])
plt.plot(result, [x * x for x in result], '-o')
plt.title('learning rate = {}'.format(lr))
plt.xlabel('x')
plt.ylabel('f(x)')
plt.show()
```

可以探究在学习率设置得不同条件下迭代情况：

大家可以尝试使用不同的学习率进行训练，会得到如下结果：



- 如果使用的学习率太小，将导致 x 的更新非常缓慢，需要更多的迭代。
- 相反，当使用过大的学习率， x 的迭代不能保证降低 $f(x)$ 的值，例如，当学习率为 $\eta = 1.1$ 时， x 超出了最优解 $x = 0$ ，并逐渐发散

(2) 带动量（Momentum）的梯度下降：

思想：让参数的更新具有惯性，每一步的更新都由前面梯度的累积和当前梯度 g 组合而成；

累积梯度更新： $v \leftarrow \alpha v + (1 - \alpha)g$; α 是动量参数

梯度更新： $x \leftarrow x - \eta * v$;

优点：加速收敛，帮助在正确道路上加速；可以帮忙跳出局部最优解；

(3) Adagrad 优化算法：

又叫自适应学习率优化算法：

核心思路：对应不同的梯度设置不同的学习率：

对于每个参数，设置一个累积平方梯度 $r \leftarrow r + g^2$

更新这个参数时候，学习率变化为： $\frac{\eta}{\sqrt{r+\delta}}$;

再在此基础上对参数更新；

(4) RMSProp

在 adagrad 的基础上进一步在学习率上进行优化； $r \leftarrow \lambda r + (1 - \lambda)g^2$

(5) Adam

在Gradient Descent 的基础上，做了如下几个方面的改进：

1、梯度方面增加了momentum，使用累积梯度： $v \leftarrow \alpha v + (1 - \alpha)g$

2、同 RMSProp 优化算法一样，对学习率进行优化，使用累积平方梯度： $r \leftarrow \lambda r + (1 - \lambda)g^2$

3、偏差纠正： $\hat{v} = \frac{v}{1-\alpha^t}$ ， $\hat{r} = \frac{r}{1-\lambda^t}$

再如上3点改进的基础上，权重更新： $w \leftarrow w - \frac{\eta}{\sqrt{\hat{r}+\delta}} * \hat{v}$

为啥要偏差纠正

10.归一化/正则化

10.1 归一化：

归一化是对神经网络输入或中间特征进行进行变换，使其满足特定统计分布的操作。

在数学上，对特征 x 进行归一化可表示为：

$$\hat{x} = \frac{x-\mu}{\sigma}$$

其中 μ 为特征均值， σ 为特征标准差。

典型方法及原理：

- **批量归一化 (Batch Normalization, BN) :**

对每个 mini-batch 内的特征，计算均值与方差并归一化，公式为 (Ioffe & Szegedy, 2015) :

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \cdot \gamma + \beta$$

其中 μ_B 、 σ_B^2 是 batch 内特征的均值与方差， γ 、 β 是可学习的缩放与平移参数，用于恢复特征表达能力。BN 通过固定 batch 内特征分布，加速网络收敛，减少对初始化的依赖。

- **层归一化 (Layer Normalization, LN) :**

对单一样本的所有特征维度计算均值与方差并归一化 (Ba et al., 2016)，适用于序列长度变化的任务 (如 NLP 中的 RNN、Transformer)，公式为：

$$\hat{x} = \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} \cdot \gamma + \beta$$

与 BN 不同，LN 的统计量基于单一样本，而非 batch，因此更适合对样本独立性要求高的场景。

10.2 正则化：

正则化是向损失函数添加额外约束，以限制模型复杂度，缓解过拟合的技术，过拟合的模型会学习训练数据中的噪声，导致泛化能力下降，通过正则化可引导模型优先学习“更简单，更具泛化性的模式”；

典型方法：

L_2 正则化 (权重衰减, Weight Decay) :

在损失函数中添加权重的 L_2 范数惩罚，公式为 (Hastie et al., 2009) :

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{task}} + \lambda \cdot \frac{1}{2} \|\mathbf{W}\|_2^2$$

其中 λ 是正则化强度， $\|\mathbf{W}\|_2^2 = \sum_{i,j} W_{i,j}^2$ 是权重矩阵的 L_2 范数。 L_2 正则化会使权重趋向于更小的值，鼓励模型学习更平滑的特征映射，避免过拟合。

L_1 正则化：

损失函数中添加权重的 L_1 范数惩罚，公式为：

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{task}} + \lambda \cdot \|\mathbf{W}\|_1$$

$\|\mathbf{W}\|_1 = \sum_{i,j} |W_{i,j}|$ 是权重矩阵的 L_1 范数。 L_1 正则化会使部分权重变为 0，起到特征选择的作用，生成更稀疏的模型 (Tibshirani, 1996)。

11. 过拟合和欠拟合：

在机器学习与深度学习中，过拟合和欠拟合是模型泛化能力不足的两种典型表现，其本质是模型复杂度与数据模式匹配失衡的结果。

模型的泛化误差可理解为偏差，方差与噪声

泛化误差 = 偏差² + 方差 + 噪声

- **偏差**：模型对真实模式的系统性偏离（由模型假设空间的局限性导致）；
- **方差**：模型对训练数据波动的敏感程度（由模型复杂度过高导致）。

欠拟合模型过于简单，过拟合模型过于复杂

欠拟合	模型无法捕捉训练数据中的基本模式，导致训练误差与测试误差均较高（Hastie et al., 2009）。	高偏差、低方差
过拟合	模型过度学习训练数据中的细节（包括噪声），导致训练误差极低但测试误差显著升高（Bishop, 2006）。	低偏差、高方差

欠拟合的核心成因：

- 模型复杂度不足**：模型假设空间无法覆盖数据的真实模式。例如，用线性模型拟合非线性分布的数据（如“异或”问题），或浅层神经网络处理高维复杂特征（如图像语义）（Goodfellow et al., 2016）。
- 特征表示缺陷**：输入特征缺乏区分性（如文本分类中仅用词频而忽略语义），导致模型无法学习有效模式（Bengio et al., 2013）。
- 训练不充分**：迭代次数过少或学习率设置不当，导致模型未收敛至最优解（Bottou, 2010）。

过拟合的核心成因：

- 模型复杂度过剩**：模型容量（如参数数量）远超过数据复杂度。例如，深层神经网络在小数据集上训练时，易记住样本噪声（Krizhevsky et al., 2012）。
- 数据质量缺陷**：训练数据量不足、分布偏移（如训练集与测试集类别分布差异大）或噪声过多，导致模型学习到虚假模式（Scheffer, 2001）。
- 优化过程偏差**：梯度下降过度收敛至训练数据的局部最优，而非真实分布的全局最优（Hochreiter & Schmidhuber, 1997）。

那么怎么诊断是欠拟合或者过拟合呢？

- 欠拟合的训练误差和验证误差均较高；随着训练迭代，两者缓慢下降并且逐渐靠近；
- 过拟合的训练误差极低，但验证误差显著高于训练误差；随着训练迭代，训练误差持续下降，验证误差先降后升；