

第 3 章

AI 模型训练与优化

【思政目标】

- （1）通过数据清洗与标注，引导学生掌握规范操作，培养诚信意识和科学精神。
- （2）通过模型选择与调参，引导学生理解复杂度与性能的平衡，培养创新思维。
- （3）通过模型评估与优化，引导学生学会数据驱动的理性决策，强化批判思维。

【学习目标】

- （1）理解 AI 模型训练的完整流程，包括数据准备、选择模型架构、设置超参数、初始化模型参数、训练模型、模型保存与部署。
- （2）掌握损失函数与优化算法的作用及其协同关系。
- （3）识别过拟合现象及常用的正则化方法。
- （4）理解常用模型评估指标和调参方法。
- （5）具备独立完成一个小规模 AI 模型训练与优化的能力。

3.1 模型训练流程

在机器学习和深度学习的世界中，模型训练是实现 AI 智能的核心步骤。无论是简单的线性回归模型，还是复杂的深度神经网络，模型训练的过程都遵循一定的流程。这个流程不仅是简单的数据输入和模型输出的过程，而是涉及数据准备、选择模型架构、设置超参数、训练循环等多个关键环节。下面将详细介绍模型训练的完整流程。

3.1.1 数据准备 (Data Preparation)

数据是机器学习的基石，模型的性能在很大程度上取决于数据的质量和数量。数据准备是模型训练的第一步，也是最为关键的一步。数据准备的过程通常包括以下几个方面。

- （1）数据收集：首先需要从各种来源收集数据。数据可以来自数据库、日志文件、网络、

传感器等。数据的数量和多样性对模型的泛化能力至关重要。

(2) 数据清洗: 收集到的原始数据通常包含噪声、缺失值、重复项等。数据清洗的过程包括去除重复数据、填充缺失值、纠正错误数据等。清洗后的数据更加干净、可靠, 有助于提高模型的学习效果。

(3) 数据标注: 对于监督学习任务 (如分类或回归), 需要对数据进行标注。标注可以是类别标签、数值标签或更复杂的结构化数据。标注的质量直接影响模型的学习能力。

(4) 数据分割: 为了评估模型的泛化能力, 通常将数据集划分为训练集、验证集和测试集。训练集用于模型的学习, 验证集用于调整超参数, 测试集用于最终评估模型的性能。

(5) 数据增强: 在图像处理、语音识别等任务中, 数据增强是一种常见的技术。通过对原始数据进行随机变换 (如旋转、缩放、剪裁、添加噪声等), 可以生成更多的训练样本, 从而提高模型的鲁棒性。

(6) 数据归一化/标准化: 为了加速模型的收敛, 通常需要对数据进行归一化或标准化处理。例如, 将特征值缩放到 $[0, 1]$ 范围或转换为均值为 0、方差为 1 的标准正态分布。

3.1.2 选择模型架构 (Model Architecture Selection)

模型架构是解决特定问题的核心工具。选择合适的模型架构对模型的性能至关重要。根据任务的不同, 可以选择不同的模型架构。

(1) 线性模型: 如线性回归、逻辑回归等, 适用于简单的回归或分类任务。

(2) 树模型: 如决策树、随机森林 (RF)、梯度提升树 (GBDT) 等, 适用于处理结构化数据。

(3) 神经网络: 如多层感知机 (MLP)、卷积神经网络 (CNN)、循环神经网络 (RNN)、变换模型 (Transformer) 等, 适用于图像处理、自然语言处理等复杂任务。

(4) 生成模型: 如变分自编码器 (VAE)、生成对抗网络 (GAN) 等, 适用于生成新的样本或数据增强。

在选择模型架构时, 需要考虑以下几个因素。

(1) 任务类型: 是分类任务、回归任务, 还是生成任务? 不同的任务需要不同的模型。

(2) 数据规模: 数据量的大小会影响模型的选择。对于小数据集, 通常选择更简单的模型; 对于大数据集, 可以使用更复杂的模型。

(3) 计算资源: 模型的训练需要大量的计算资源。复杂的模型 (如深度神经网络) 需要更多的 GPU 资源和更长的训练时间。

3.1.3 设置超参数 (Hyperparameter Setting)

超参数是模型训练过程中需要手动设置的参数, 它们不会在训练过程中自动更新。常见的超参数如下。

(1) 学习率 (Learning Rate): 控制模型参数更新的步长。学习率过大会导致模型无法收敛, 而学习率过小则会导致训练速度过慢。

(2) 批量大小 (Batch Size): 每次更新模型参数时使用的样本数量。批量大小会影响模型

训练的稳定性和速度。

(3) 迭代次数 (Epochs): 模型在整个训练集上训练的次数。过多的迭代可能导致过拟合, 而过少的迭代则可能导致欠拟合。

(4) 正则化参数: 如 L1/L2 正则化的权重, 用于防止模型过拟合。

(5) 隐藏层神经元数量: 在神经网络中, 隐藏层神经元的数量将会影响模型的复杂度和学习能力。

(6) 优化算法: 如随机梯度下降 (SGD)、Adam、RMSprop 等, 不同的优化算法对模型的收敛速度和性能有不同的影响。

3.1.4 初始化模型参数 (Parameter Initialization)

模型参数在训练开始时需要初始化。参数的初始化方法对模型的收敛速度和最终性能具有重要影响。常见的模型参数初始化方法如下。

(1) 随机初始化: 将参数随机初始化为较小的值, 避免参数全部为 0 或相同的值。

(2) Xavier 初始化: 适用于线性激活函数的神经网络, 确保每层的输出方差与输入方差一致。

(3) He 初始化: 适用于 ReLU 激活函数的神经网络, 能够更好地平衡前向传播和反向传播的梯度。

3.1.5 训练循环 (Training Loop)

训练循环是模型训练的核心部分。在每次迭代中, 数据会被分批送入模型, 模型通过前向传播计算输出, 并通过反向传播更新参数。训练循环通常分为以下步骤。

(1) 前向传播: 输入数据通过模型的各个层, 计算输出值。

(2) 计算损失: 将模型的输出与真实标签进行比较, 计算损失函数的值。

(3) 反向传播: 通过链式法则计算损失函数对模型参数的梯度。

(4) 参数更新: 根据优化算法 (如 SGD、Adam 等), 使用计算出的梯度更新模型参数。

(5) 验证与调整: 在训练过程中, 可以使用验证集评估模型的性能。如果验证集的损失不再下降, 则可能要调整超参数或提前终止训练。

3.1.6 模型保存与部署 (Model Saving and Deployment)

当模型训练完成后, 可以将其保存为文件, 以便后续使用。常见的模型保存格式如下。

(1) HDF5 格式: 适用于 TensorFlow 模型和 Keras 模型。

(2) ONNX 格式: 适用于跨平台部署, 支持多种深度学习框架。

(3) TorchScript 格式: 适用于 PyTorch 模型。

保存后的模型可以直接部署到生产环境中, 用于实时推理或批量处理任务。

3.2 损失函数与优化算法

在模型训练的过程中，损失函数 (Loss Function) 和优化算法 (Optimization Algorithm) 是两个核心概念。它们分别用于衡量模型的性能和指导模型参数的更新。理解这两者的作用 and 选择方法是掌握机器学习与深度学习的关键。

3.2.1 损失函数的作用

损失函数是模型训练的动力源泉。它定义了模型的预测结果与真实标签之间的差距。通过最小化损失函数，模型可以逐步逼近最优解，从而在新的数据上表现出更好的预测能力。

在实际应用中，不同的任务需要使用不同的损失函数。

(1) 分类任务：常见损失函数包括交叉熵损失 (Cross Entropy Loss, 简称 CEL)，用于衡量模型预测的概率分布与真实标签之间的一致性。

(2) 回归任务：常见损失函数包括均方误差 (Mean Squared Error, 简称 MSE)，用于衡量模型预测值与真实值之间的平方差。

(3) 生成任务：常见损失函数包括对抗损失 (Adversarial Loss)，用于在生成对抗网络 (GAN) 中衡量生成器生成的样本与真实样本之间的相似度。

损失函数的选择直接影响模型的训练效果和最终性能，因此需要根据任务的特点进行合理选择。

3.2.2 优化算法的作用

优化算法用于更新模型的参数，以最小化损失函数。在训练过程中，模型通过优化算法不断调整参数，从而使损失函数的值逐渐下降。

优化算法的选择对模型的收敛速度和最终性能有着重要影响。不同的优化算法适用于不同的场景。

(1) 梯度下降 (Gradient Descent, 简称 GD)：经典的优化算法，通过计算损失函数对参数的梯度，沿梯度方向更新参数。

(2) 随机梯度下降 (Stochastic Gradient Descent, 简称 SGD)：是对梯度下降的改进，每次只使用一个样本更新参数，具有更快的收敛速度和更好的抗噪功能。

(3) 小批量梯度下降 (Mini-Batch Gradient Descent, 简称 MBGD)：结合了梯度下降和随机梯度下降的优点，每次使用一小批数据更新参数，兼顾了计算效率和收敛稳定性。

(4) 自适应优化算法：如 Adam、RMSprop，它们根据历史梯度动态调整学习率，适用于大规模数据集和复杂模型。

优化算法的选择需要考虑以下几个因素。

(1) 任务类型：不同的任务可能需要不同的优化算法。例如，分类任务可以使用 Adam，而回归任务可能更适合使用随机梯度下降。

(2) 数据规模：大规模数据集通常可以使用小批量梯度下降或自适应优化算法，而小规模数据集可能更适合使用随机梯度下降。

(3) 计算资源：某些优化算法（如 Adam）计算复杂度较高，可能需要更多的计算资源。

3.2.3 损失函数与优化算法的协同作用

损失函数与优化算法是紧密相关的。损失函数定义了模型的目标，而优化算法则是实现这一目标的工具。两者共同决定了模型的训练过程和最终性能。

例如，在训练一个分类模型时，损失函数与优化算法的协同作用如下。

(1) 损失函数（如交叉熵损失）可用于衡量模型预测的概率分布与真实分布的差距。

(2) 优化算法（如 Adam）可用于根据损失函数的梯度，逐步调整模型的参数，使模型的预测结果逐渐接近真实标签。

通过不断迭代这个过程，模型最终能够在新数据上表现出良好的泛化能力。

3.2.4 选择合适的损失函数与优化算法

在实际应用中，选择合适的损失函数和优化算法是提高模型性能的关键。以下是一些实用的建议。

(1) 根据任务选择损失函数：明确任务的类型（分类、回归、生成等），并选择相应的损失函数。例如，分类任务可以选择交叉熵损失，回归任务可以选择均方误差。

(2) 根据数据特点选择优化算法：如果数据集较大且分布均匀，可以使用小批量梯度下降或自适应优化算法；如果数据集较小且噪声较多，可以使用随机梯度下降。

(3) 调整超参数：优化算法的超参数（如学习率、批量大小等）对训练效果具有重要影响，可以通过网格搜索或随机搜索找到最优的超参数组合。

(4) 结合正则化技术：在损失函数中加入正则化项（如 L1/L2 正则化），可以防止模型过拟合，从而提高模型的泛化能力。

通过理解损失函数与优化算法的作用、选择方法及协同作用，可以更好地控制模型的训练过程，从而实现更高效、更准确的模型训练。

3.3 过拟合与正则化

在机器学习和深度学习的模型训练过程中，过拟合（Overfitting）是一个常见且重要的问题。过拟合是指模型在训练数据上表现得非常好，但在新数据（测试数据或实际应用数据）上表现不佳。这种现象通常发生在模型过于复杂或数据量不足的情况下。为了解决过拟合问题，正则化（Regularization）技术应运而生，它通过限制模型的复杂度，帮助模型更好地泛化到新数据上。

3.3.1 过拟合的定义与表现

过拟合是指模型在训练数据上学习得“过于好”，以至于它不仅学到了数据中的模式，还

学到了噪声和细节。在这种情况下，模型对训练数据的拟合程度很高，但对外部数据的泛化能力较差。

过拟合的表现可以从以下几个方面观察。

(1) 训练集表现 VS 验证集表现：在训练集上，模型的损失值很低，准确率很高；但在验证集或测试集上，模型的损失值较高，性能明显下降。

(2) 模型复杂度：过拟合通常发生在模型过于复杂的情况下，例如，使用了太多参数、层数过深的神经网络，或者决策树的深度过大。

(3) 训练数据规模：如果训练数据量不足，模型可能会“记住”每个样本的细节，而无法学习到数据的普遍规律。

过拟合的原因可以归结为以下几个方面。

(1) 模型复杂度过高：模型的参数过多或模型的结构过于复杂（如深度神经网络的层数过多），导致模型能够“完美”拟合训练数据，但无法泛化到新数据。

(2) 训练数据不足：当数据量不够时，模型无法充分学习到数据中的普遍规律，而是过度关注于噪声和细节。

(3) 噪声数据：如果训练数据中存在噪声（如标注错误或异常值），则模型可能会错误地学习这些噪声，从而导致过拟合。

(4) 数据分布不均衡：如果训练数据中某些类别的样本数量远多于其他类别，则模型可能会倾向于预测这些多数类别，而忽略少数类别。

3.3.2 正则化的作用与实现

正则化是一种通过限制模型的复杂度来防止过拟合的技术。它的核心思想是在损失函数中加入一个正则化项，用于减少模型的复杂度。通过这种方式，模型在学习数据模式的同时，避免过度拟合训练数据。

常见的正则化方法如下。

1. L1 正则化

L1 正则化（Least Absolute Shrinkage and Selection Operator, Lasso 正则化）是指在损失函数中加入模型权重的 L1 范数（即权重的绝对值之和）。

其公式如下：

$$\text{Loss}_{\text{L1}} = \text{Loss} + \lambda \sum_{i=1}^n |\omega_i|$$

其中：Loss 是原始损失函数（如均方误差或交叉熵）； λ 是正则化系数，用于控制正则化的强度； ω_i 是模型中的权重参数。

L1 正则化具有以下特点。

(1) 稀疏性：L1 正则化倾向于将一些权重直接压缩为 0，从而实现特征选择。这意味着模型在训练过程中会自动忽略某些不重要的特征，从而减少模型复杂度。

(2) 特征选择：由于 L1 正则化的稀疏性，它可以用于高维数据的特征选择，尤其是在特征数量远大于样本数量的情况下。

(3) 鲁棒性: L1 正则化对异常值具有一定的鲁棒性, 因为其对权重的惩罚是绝对值形式, 不会过度放大异常值的影响。

L1 正则化适用于以下场景。

- (1) 高维数据: 当特征数量非常多时, L1 正则化可以用于筛选出重要的特征。
- (2) 稀疏解需求: 当需要模型参数具有稀疏性时, L1 正则化是一种理想的选择。
- (3) 特征选择: 在特征工程中, L1 正则化可以用于自动选择对目标变量影响最大的特征。

2. L2 正则化

L2 正则化 (Ridge 正则化) 是指在损失函数中加入模型权重的 L2 范数 (即权重的平方和)。其公式如下:

$$\text{Loss}_{L2} = \text{Loss} + \lambda \sum_{i=1}^n |\omega_i|^2$$

其中: Loss 是原始损失函数; λ 是正则化系数; ω_i 是模型中的权重参数。

L2 正则化具有以下特点。

(1) 平滑性: L2 正则化倾向于使权重值较小但不为 0, 从而使模型更加平滑。这种平滑性有助于提高模型的泛化能力。

(2) 防止过拟合: 通过限制权重的大小, L2 正则化可以有效防止模型过拟合, 尤其是在数据量较小或特征相关性较高的情况下。

(3) 稳定性: L2 正则化对异常值相对敏感, 但通过限制权重的大小, 通常会增强模型的稳定性。

L2 正则化适用于以下场景。

(1) 防止过拟合: 当训练数据量较小或特征之间存在较强相关性时, L2 正则化可以有效防止模型过拟合。

(2) 平滑性需求: 当需要模型参数具有较小的数值时, L2 正则化是一种理想的选择。

(3) 大规模数据集: 在处理大规模数据集时, L2 正则化通常比 L1 正则化更容易优化。

L1 正则化与 L2 正则化的比较如下。

(1) 正则化强度。

L1 正则化: 对权重的惩罚是绝对值的, 倾向于将一些权重直接压缩为 0。

L2 正则化: 对权重的惩罚是平方的, 倾向于使权重值较小但不为 0。

(2) 稀疏性与平滑性。

L1 正则化: 产生稀疏解, 适合特征选择和简化模型。

L2 正则化: 产生平滑解, 适合防止过拟合和提高模型泛化能力。

(3) 对异常值的敏感性。

L1 正则化: 对异常值相对鲁棒, 因为其对权重的惩罚是绝对值形式。

L2 正则化: 对异常值较为敏感, 因为其对权重的惩罚是平方形式, 会放大对异常值的影响。

(4) 优化难度。

L1 正则化: 由于 L1 正则化是不可微的, 优化过程相对复杂, 通常需要使用特定算法 (如

坐标下降法)。

L2 正则化: L2 正则化是可微的, 优化过程更加简单, 可以直接使用梯度下降法。

3. Dropout

Dropout 是一种在深度学习中广泛使用的正则化技术。它在训练过程中随机丢弃一部分神经元, 从而防止模型对特定神经元的过度依赖。

通过随机丢弃神经元, Dropout 可以模拟多个子网络的学习过程, 从而提高模型的泛化能力。

4. 数据增强

数据增强 (Data Augmentation) 是一种通过对训练数据进行随机变换 (如旋转、缩放、裁剪、添加噪声等) 来增加数据多样性的技术。

数据增强可以有效地防止模型过拟合, 因为它能够提供更多的训练样本, 让模型学习到更广泛的数据特征。

5. 早停

早停 (Early Stopping) 是一种简单而有效的正则化方法。它通过监控验证集的性能, 在模型性能不再提升时提前终止训练。

早停可以防止模型在训练数据上过度学习, 从而避免过拟合。

3.3.3 正则化技术的实际应用

在实际应用中, 多种正则化方法可以结合使用, 以防止过拟合。

1. 正则化方法的组合应用

(1) L1 正则化和 L2 正则化的结合: 在某些情况下, 可以同时使用 L1 正则化和 L2 正则化 (又被称为 “Elastic Net 正则化”), 以获得两者的优点。

在实际应用中, L1 正则化和 L2 正则化可以单独使用, 也可以同时使用。选择哪种正则化方法取决于具体的任务需求。

- L1 正则化: 用于高维数据、特征选择或稀疏解需求。
- L2 正则化: 用于防止过拟合、提高模型泛化能力或需要平滑解的场景。
- Elastic Net 正则化: 结合 L1 正则化和 L2 正则化的优点, 适用于既有特征选择需求, 又需要防止过拟合的场景。

其公式如下:

$$\text{Loss}_{\text{Elastic Net}} = \text{Loss} + \lambda_1 \sum_{i=1}^n |\omega_i| + \lambda_2 \sum_{i=1}^n \omega_i^2$$

其中, λ_1 和 λ_2 分别用控制 L1 正则化和 L2 正则化的强度。

(2) Dropout 与数据增强的结合: 在图像分类任务中, 可以同时使用 Dropout 和数据增强, 以进一步提高模型的泛化能力。

(3) 早停与调度策略的结合: 可以通过早停策略结合学习率调度 (Learning Rate

Scheduling)，在训练过程中动态调整学习率，从而提高模型的性能。

2. 正则化与模型选择

正则化技术不仅可以帮助防止过拟合，还可以用于模型选择。通过调整正则化系数，可以在模型的复杂度和泛化能力之间找到平衡。

较大的：模型更倾向于简单，可能会导致欠拟合。

较小的：模型更复杂，可能会导致过拟合。

因此，选择合适的正则化系数是模型训练中的一个重要环节。

3. 过拟合与正则化的实验观察

为了更直观地理解过拟合和正则化的作用，可以通过实验观察模型在不同情况下的表现。

(1) 不使用正则化：模型在训练集上性能很好，但在验证集上性能较差，说明模型过拟合。

(2) 使用 L2 正则化：模型的复杂度受到限制，训练集和验证集的性能差距缩小，说明模型泛化能力增强。

(3) 使用 Dropout：模型的随机性增加，训练过程更加稳健，验证集的性能进一步提升。

通过上述实验，可以更好地理解正则化技术对模型性能的影响。

通过上述内容，我们详细介绍了过拟合的定义、原因、正则化技术的作用及实际应用方法。理解过拟合与正则化的关系，能够帮助我们在模型训练中更好地控制模型的复杂度，从而提高模型的泛化能力和实际应用效果。

3.4 模型评估与调参

在机器学习和深度学习的模型开发过程中，模型评估和调参是两个至关重要的环节。通过科学的评估方法和精细的参数调整，人们能够全面地了解模型的性能，并优化模型的表现。

3.4.1 模型评估

模型评估的目的是验证模型在实际应用中的表现，确保其具备良好的泛化能力。由于训练数据和实际数据可能存在分布差异，仅依赖训练集上的表现无法准确判断模型的优劣。因此，需要通过验证集和测试集对模型进行评估。

模型评估的重要性体现在以下几个方面。

(1) 验证泛化能力：通过验证集或测试集评估模型，判断其在新数据上的表现。

(2) 发现模型问题：评估结果可以揭示模型存在的不足，如对特定类别的预测不准确、对噪声敏感等。

(3) 指导优化方向：根据评估结果，可以选择调整模型架构、超参数或训练策略，以提升模型性能。

模型评估通常涉及以下几种方法。

(1) 数据集划分：将数据集划分为训练集、验证集和测试集是模型评估的基础。常见的数据集划分方法如下。



- 随机划分：将数据按一定比例随机分为训练集（如 70%）、验证集（如 15%）和测试集（如 15%）。
- 分层抽样（Stratified Sampling）：在分类任务中，确保训练集和验证集中的类别比例与总体数据一致，避免产生类别不均衡的问题。
- 时间序列划分：对于时间序列数据，按照时间顺序划分数据集，以确保测试集中的数据发生在训练集之后。

（2）交叉验证（Cross-Validation）：交叉验证是一种更稳健的评估方法，适用于数据量有限的情况。常见的交叉验证方法如下。

- K 折交叉验证（K-Fold Cross-Validation）：将数据集分为 K 个子集（称为“折”），依次将每个子集作为验证集，其余子集作为训练集，最终取 K 次验证结果的平均值。
- 留一交叉验证（Leave-One-Out）：当数据量较少时，可以将每个样本单独作为验证集，其余样本作为训练集。
- 分层交叉验证（Stratified K-Fold Cross-Validation）：在分类任务中，确保每层中的类别比例与总体数据一致。

3.4.2 常用的性能指标

根据任务的不同（如分类、回归、聚类等），需要使用不同的性能指标来评估模型的表现。常用的性能指标包括准确率（Accuracy）、精确率（Precision）、召回率（Recall）、F1 分数（F1 Score）、ROC 曲线与 AUC。

ROC 曲线以假正率（False Positive Rate）为横轴、真正率（True Positive Rate）为纵轴，绘制模型的性能曲线。

AUC（Area Under Curve）是 ROC 曲线下的面积，值越大表示模型性能越好。

3.4.3 模型调参

在机器学习中，模型的性能往往受到许多因素的影响，如模型的超参数、数据的质量、特征选择等。其中，调整超参数是模型优化中最重要的一环之一，因为正确的调参可以使模型的效果最大化。

1. 什么是超参数

超参数（Hyperparameters）是机器学习算法中需要人为设定的参数，它们不能直接从训练数据中学习得出。与之对应的是模型参数（Model Parameters），它们是模型内部学习得来的参数。

为了更好地理解这一区别，我们可以看一个具体算法。例如，在支持向量机中，用户必须在训练前设定惩罚系数 C、选择核函数 kernel() 并配置其参数 gamma，这些都属于超参数。而算法在训练过程中自动学习到的决策边界所对应的权重 w 和偏置 b 则是模型参数。

2. 调整超参数方法

（1）调整数据预处理和特征选择方法：在进行模型选择之前，可以尝试不同的数据预处理

理和特征选择技术，以提高模型的准确性和泛化能力。

(2) 逐步调整超参数：建议先调整较重要的参数（如学习率、迭代次数等），再尝试调整其他参数。同时，建议进行逐步的超参数调整，而非大幅度调整。

(3) 交叉验证：使用 K 折交叉验证来评估不同超参数组合下的模型性能，并选择最优的参数组合。

(4) 验证集：首先将训练集分为训练集和验证集，然后在验证集上评估模型效果，并使用其来微调超参数。在选择最终的超参数时，可以使用验证集来确认模型的泛化能力。

(5) 经验法则：可以根据经验法则对超参数进行调整。例如，学习率通常应设置为 0.01 或更小，批量大小可以设置为 32 或 64 等。

3.4.4 超参数搜索

在机器学习的实际应用中，我们往往需要选择合适的超参数才能得到一个好的模型。搜索超参数的方法有很多种，下面介绍其中几种主流的方法。

1. 网格搜索

网格搜索是一种最常见、最简单的参数选择方法。它将每个超参数的值域分成若干份，对每个超参数的取值都进行一次穷举搜索，获得不同参数组合的模型效果，最后找到最优超参数组合。

网格搜索步骤如下。

- (1) 定义需要调优的超参数及其取值范围。
- (2) 遍历所有可能的组合，训练并评估模型。
- (3) 选择性能最优的参数组合。

优点：简单直观，适用于参数数量较少的情况。

缺点：计算成本高，适用于参数取值空间较小的情况。

2. 随机搜索

随机搜索与网格搜索类似，都是通过搜索超参数空间来寻找最优的超参数组合。不过它在超参数搜索的空间上采用均匀分布或高斯分布等概率分布进行随机采样。这比网格搜索更加灵活，因为我们可以自己决定搜索空间的分布形式，也可以优化搜索空间的采样密度。

随机搜索步骤如下。

- (1) 定义需要调优的超参数及其取值范围。
- (2) 随机选择一组参数，训练并评估模型。
- (3) 重复多次，选择性能最优的参数组合。

优点：计算成本较低，适用于参数取值空间较大的情况。

缺点：可能无法找到全局最优解。

3. 贝叶斯优化

贝叶斯优化是一个优雅而强大的超参数搜索算法。它可以根据历史结果更新模型，并选

择新的超参数进行评估。与传统的网格搜索、随机搜索相比，贝叶斯优化可以通过高效地利用历史信息来加速超参数搜索。

贝叶斯优化步骤如下。

- (1) 定义需要调优的超参数及其取值范围。
- (2) 基于历史评估结果，构建目标函数的概率分布。
- (3) 选择下一个最有可能提升性能的超参数组合。
- (4) 重复迭代，直到找到最优解。

优点：效率高，适用于参数取值空间较大的情况。

缺点：实现复杂，需要一定的计算资源。

4. 早停

早停是一种通过在验证集性能不再提升时提前终止训练的方法。

早停步骤如下。

- (1) 监控验证集的损失或性能指标。
- (2) 当验证集性能在一定轮次内不再提升时，终止训练。

优点：防止模型过拟合，节省训练时间。

缺点：可能过早终止训练，导致模型未完全收敛。

调整超参数需要注意以下几点。

(1) 选择合适的搜索算法：搜索算法对超参数搜索有着重要的影响。网格搜索和随机搜索是目前应用非常广泛的两种方法，它们各有优缺点，可以根据情况选择。

(2) 搜索尽可能多的超参数：我们无法预先知道哪些超参数会影响模型性能，因此建议搜索尽可能多的超参数，以便找到最优解。

(3) 均衡时间和性能：超参数搜索需要花费大量时间和计算资源，需要在效果和速度之间做出好的平衡。

(4) 多次搜索取平均：由于超参数搜索本质上是一种随机过程，同一组超参数可能会得出不同的结果。因此，建议进行多次搜索并取平均结果。

综上所述，调整超参数是优化模型性能的一个重要步骤。通常使用的搜索方法是网格搜索和随机搜索，其中后者较为灵活。超参数搜索需要注意搜索算法、搜索尽可能多的超参数、均衡时间和性能、多次搜索取平均等问题。

当然，调整超参数只是优化模型性能的一部分，更好的数据质量、特征工程等也会对模型性能有很大的提升。在实际应用中，我们需要综合考虑不同因素，才能得到一个最优的模型。



习题 3

一、单选题

1. 模型训练的第一步通常是 ()。

A. 模型保存

B. 数据准备

- C. 设置超参数
D. 初始化模型参数
2. 下列不属于超参数范畴的是 ()。
- A. 学习率
B. 批量大小
C. 权重系数
D. 网络层数
3. 在损失函数中, 常用于回归任务的是 ()。
- A. 交叉熵损失
B. 均方误差
C. Hinge 损失
D. Softmax 损失
4. 优化算法 Adam 结合了 () 两种算法的思想。
- A. SGD 与 Momentum
B. RMSProp 与 Momentum
C. SGD 与 RMSProp
D. Adagrad 与 Momentum
5. 过拟合的表现之一是 ()。
- A. 训练集精度高, 测试集精度低
B. 训练集精度低, 测试集精度高
C. 训练集和测试集精度都低
D. 模型完全无法收敛
6. 常用的正则化技术不包括 ()。
- A. L1 正则化
B. L2 正则化
C. Dropout
D. 梯度下降
7. 在模型评估中, 分类任务常用的指标是 ()。
- A. RMSE
B. R^2
C. 准确率
D. MAE
8. 网格搜索和随机搜索用于 ()。
- A. 模型训练
B. 模型评估
C. 超参数搜索
D. 数据清洗
9. 在初始化模型参数时, 深度网络常用的方法是 ()。
- A. 全 0 初始化
B. Xavier 初始化
C. 固定常数初始化
D. 数据集均值初始化
10. 模型保存常用的文件格式不包括 ()。
- A. .pth
B. .pt
C. .csv
D. .h5

二、填空题

1. 模型训练流程的最后一步通常是_____与_____。
2. 衡量分类模型性能的常用指标包括准确率、精确率_____、_____、_____与_____。
3. Dropout 是一种_____技术, 用于防止模型过拟合。

4. 在 Adam 优化算法中, 融合了 RMSProp 与_____的思想。
5. 交叉验证常用于评估模型的_____能力。

三、判断题

1. 超参数在训练过程中会自动更新。 ()
2. 交叉熵损失常用于分类问题。 ()
3. L2 正则化通过惩罚权重绝对值的大小来限制模型复杂度。 ()
4. 随机搜索一定比网格搜索更快。 ()
5. 模型保存的主要目的是便于后续加载使用, 而无须重新训练。 ()

四、简答题

1. 简述过拟合产生的原因及两种常见的解决方法。
2. 请列出 AI 模型训练的 6 个主要步骤, 并简要说明每个步骤的作用。