南京理工大学
NANJING UNIVERSITY OF SCIENCE & TECHNOLOGY

模型评估与选择

程煦

xcheng8@njust.edu.cn

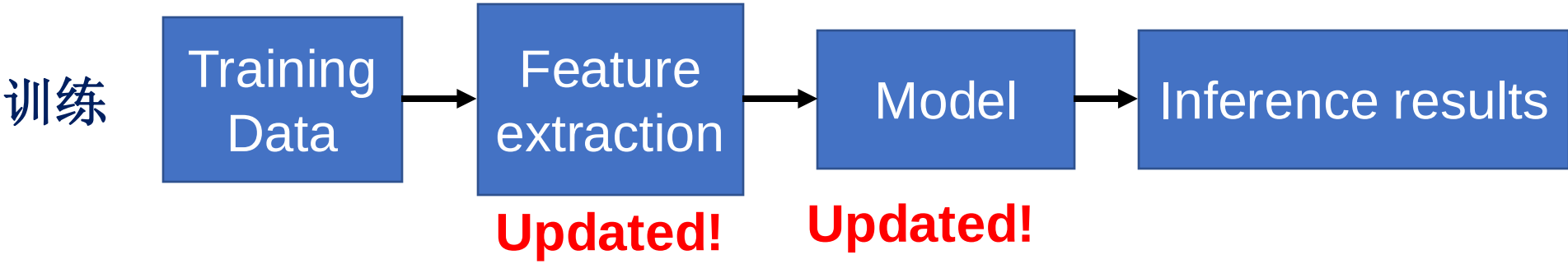
计算机科学与工程学院

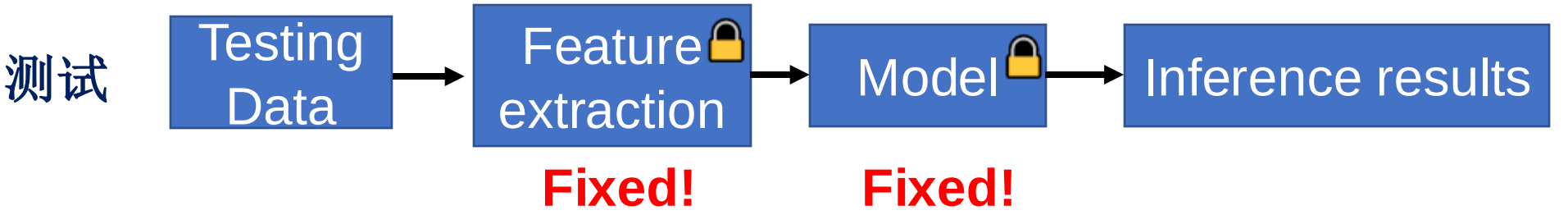
Outline

- 经验误差和泛化误差
- 过拟合与正则化
- 性能评估

机器学习过程

训练数据∩测试数据=∅





泛化能力：指的是**模型在未见数据（测试集或真实环境）上的表现能力，**即它能否正确处理训练数据之外的新数据。

经验误差与泛化误差

- **经验误差（Empirical Error）**：模型在训练集上的误差，亦称“训练误差”
  - 训练数据集的平均损失（如均方误差误差MSE、交叉熵），用来评估模型在训练数据上的拟合程度

$$R_{emp}(g_\theta) = \frac{1}{n}\sum_{i=1}^{n} L(g_\theta(x_i), y_i) = \frac{1}{n}\sum_{i=1}^{n} L(\hat{y}_i, y_i)$$

- **泛化误差（Generalization Error）**：模型在未见数据（测试集或真实环境）上的误差，亦称测试误差
  - 测试数据集的平均损失，用来衡量模型的泛化能力，即模型能否正确预测训练数据之外的新样本。

$$R_{test}(g_\theta) = \frac{1}{n'}\sum_{i=1}^{n'} L\left(g_\theta\left(x_{test,i}\right), y_{test,i}\right) = \frac{1}{n}\sum_{i=1}^{n} L(\hat{y}_{test,i}, y_{test,i})$$

经验误差与泛化误差

- 泛化误差越小越好
- 经验误差是否越小越好？

NO! 因为会出现“过拟合”(overfitting)

经验误差低 ≠ 泛化误差低

- **过拟合定义：模型在训练数据上表现良好，但在新数据（测试集）上表现较差→经验误差低，泛化误差高！**

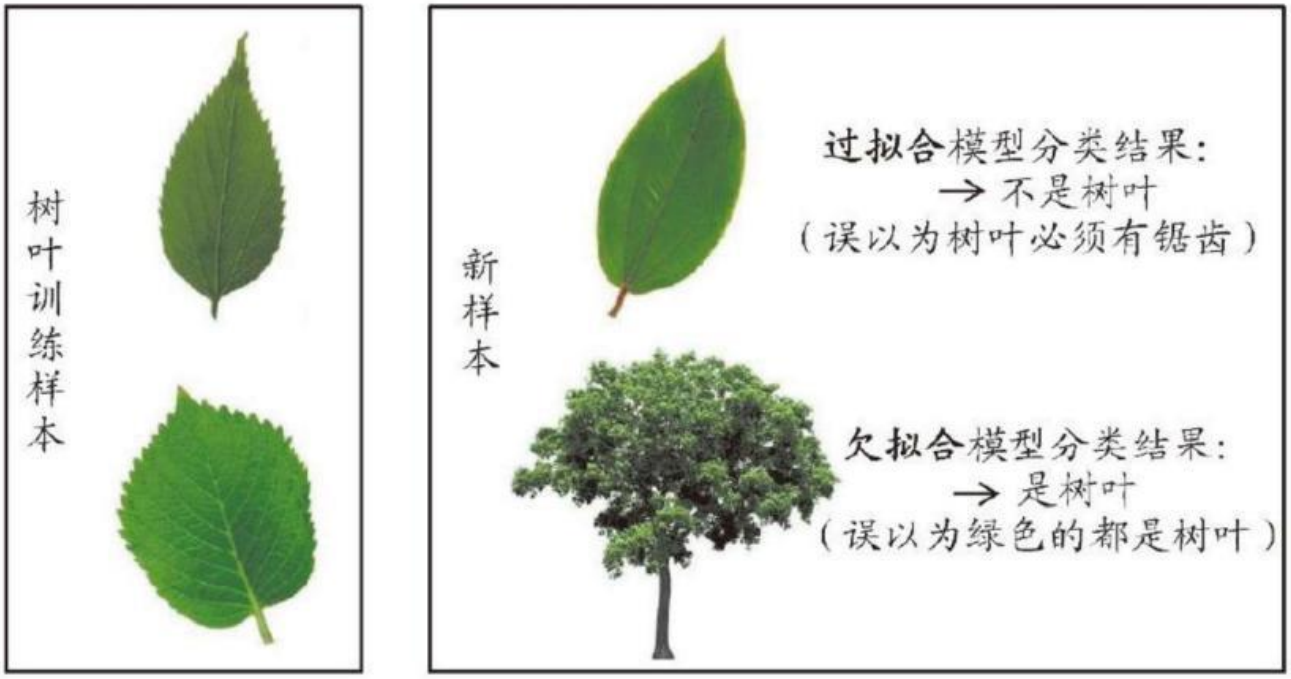
- 过拟合原因：追求提高对训练数据的预测能力，使用参数过多的模型容易拟合训练数据中的噪声。

- 举例：训练一个神经网络分类垃圾邮件，如果过拟合，模型可能会记住训练集中特定邮件的单词组合，而不是学习通用的垃圾邮件特征。

模型选择旨在避免过拟合并提高模型的泛化能力！

过拟合与欠拟合

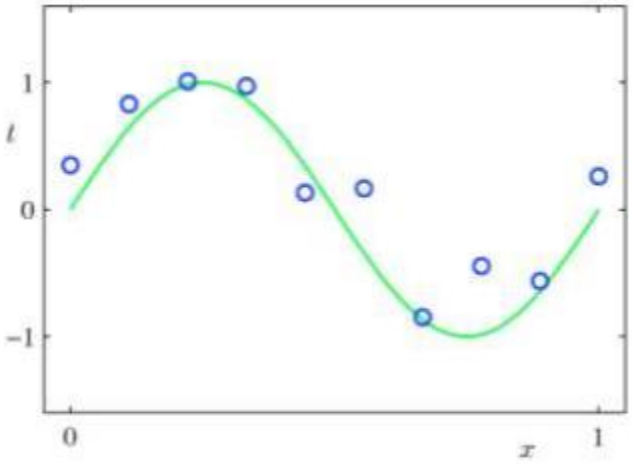
- **过拟合（overfitting）：<u>模型经验误差低，泛化误差高</u>**，即模型过度学习了训练数据（包括噪声），导致泛化能力下降
  - **模型太复杂**，包含过多参数或层数，能拟合训练数据中的每个细节，包括噪声
- **欠拟合（underfitting）：<u>模型经验误差高，泛化误差高。</u>**
  - **模型过于简单**，无法充分且有效地捕捉数据的关键特征



过拟合—多项式曲线拟合

- 给定数据集

$$D=\{(x_1, y_1),(x_2, y_2),\ldots,(x_n, y_n)\}$$



- 假定模型

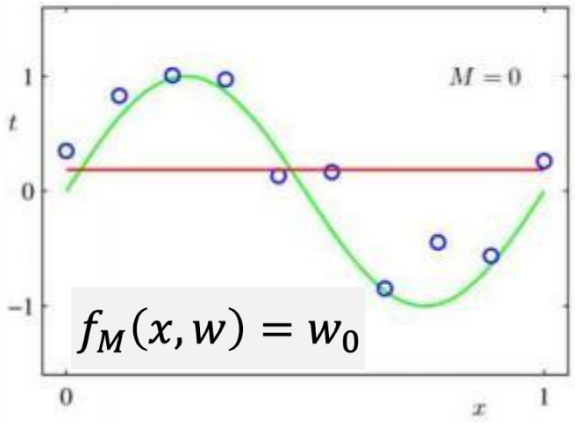
$$f_M(x, w) = w_0 + w_1 x^1 + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

- 最小化经验误差

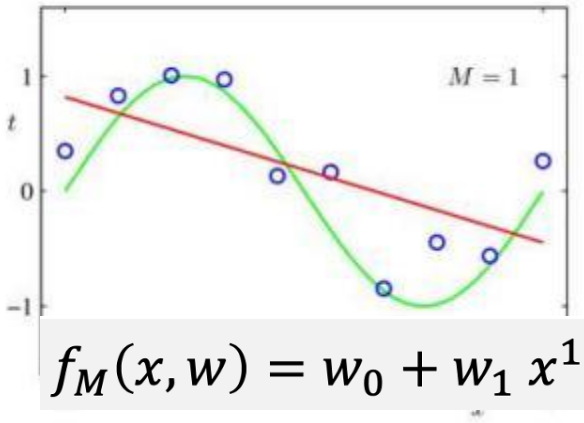
$$min_w \quad \frac{1}{n}\sum_{i=1}^{n} L(f_M(x_i, w), y_i) = \frac{1}{n}\sum_{i=1}^{n} L(\hat{y}_i, y_i)$$

过拟合—多项式曲线拟合

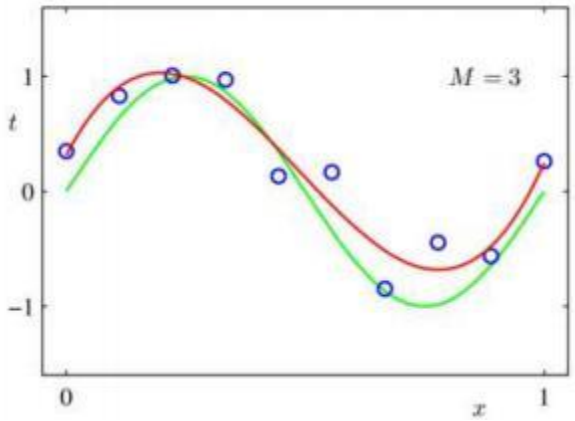
$$f_M(x,w) = w_0 + w_1\,x^1 + w_2\,x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

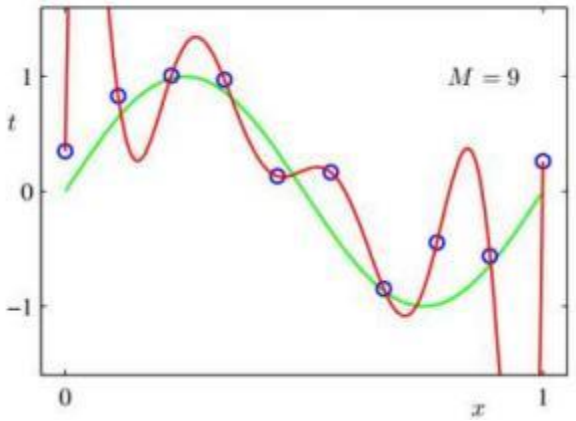


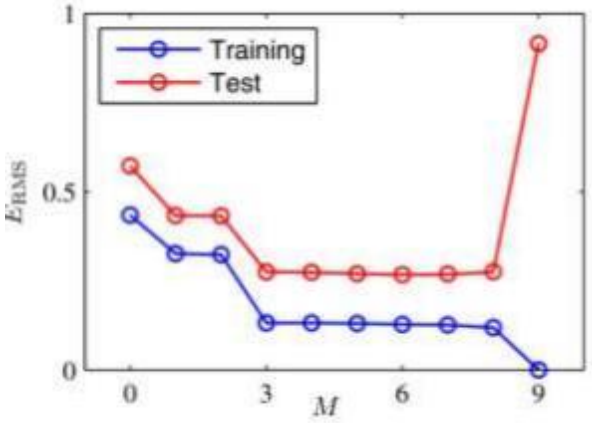
$f_M(x,w) = w_0$



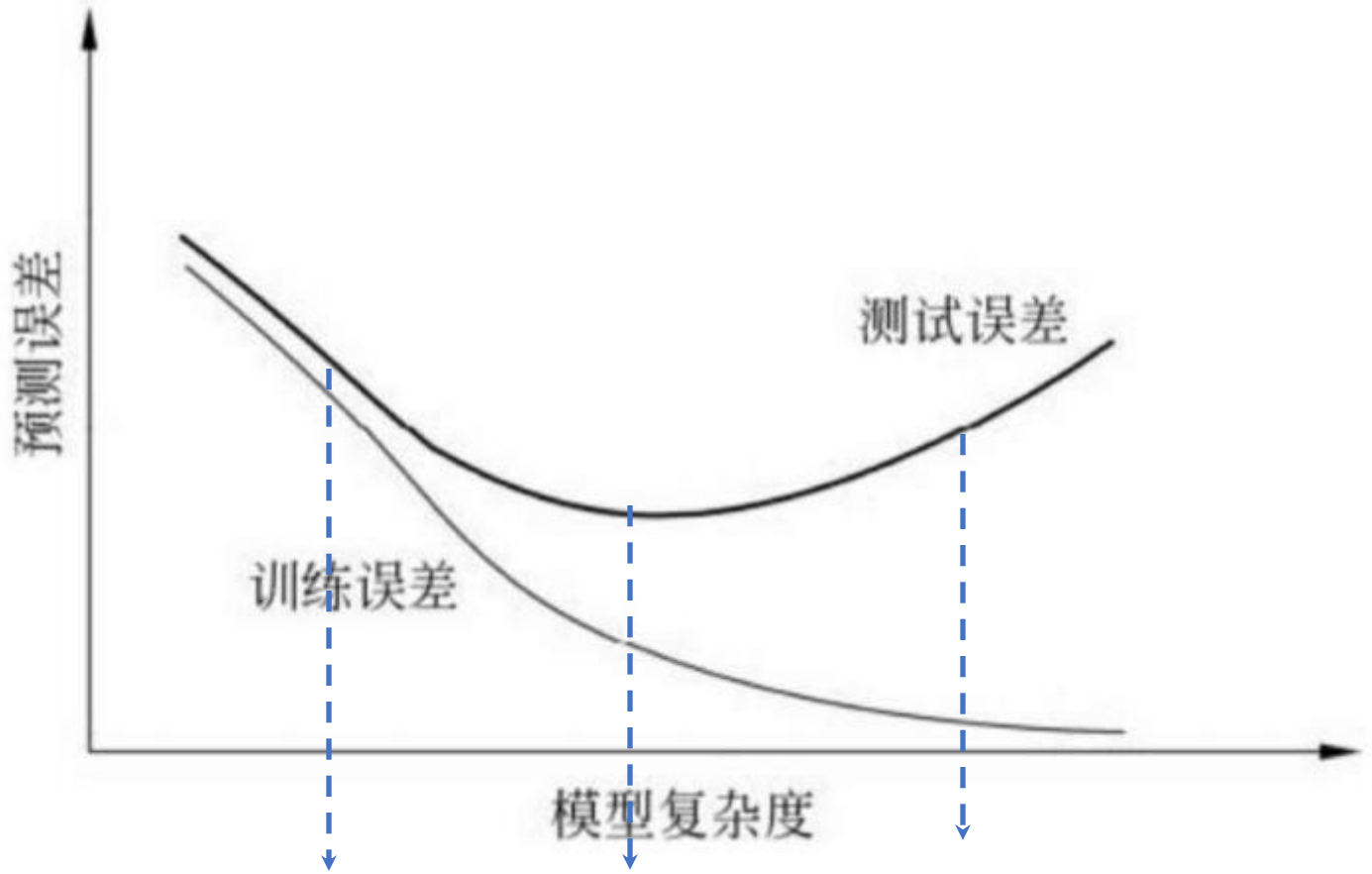
$f_M(x,w) = w_0 + w_1\,x^1$







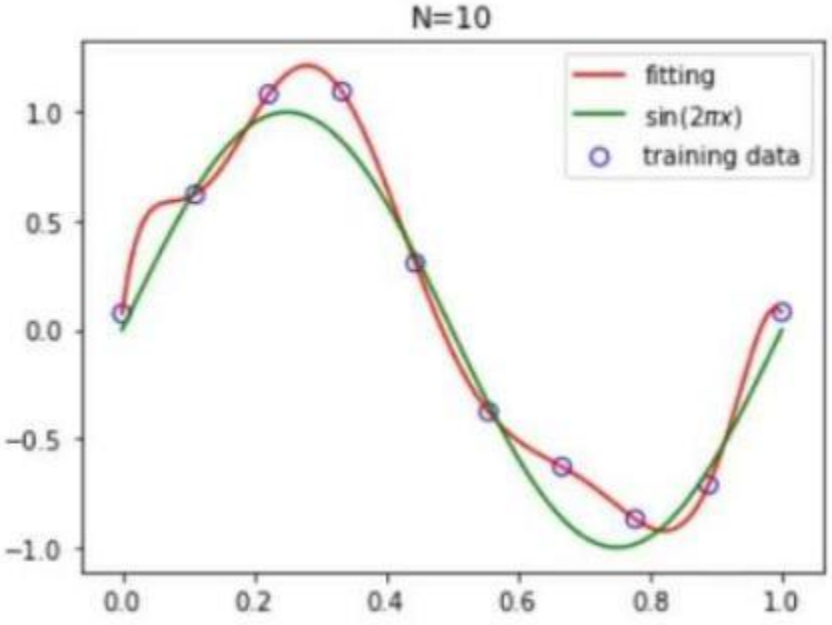
如何选择合适的模型复杂度避免过拟合？

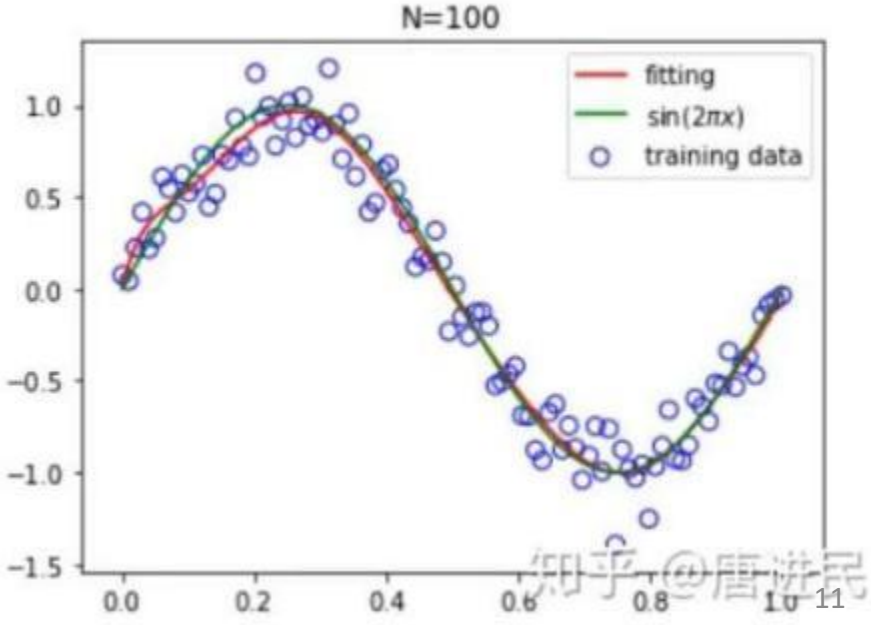


欠拟合　　较为合适的模型复杂度　　过拟合

解决过拟合—增加训练数据

- **数据规模增加**能够有效的减轻模型的过拟合问题。
  - 训练数据太少，模型容易记住训练样本中的特定模式（包括噪声），而不是学习通用特征。
  - 模型接触到更多的样本，能够学习到数据的整体规律，而不是仅仅记住训练集中的个别特征，从而提高泛化能力。





解决过拟合—交叉验证

- 交叉验证：在数据有限的情况下，**通过多次训练和测试，确保模型不会过度依赖某一部分数据**，从而提升泛化能力。
  - 在数据有限的情况下，如果只使用固定的训练集，模型可能会过度学习该数据的特定模式，而无法泛化到新数据。
  - **K 折交叉验证（K-Fold Cross Validation）：**将数据集划分为 K 份，每次用 K-1 份训练，1 份测试，循环 K 次，最终计算所有测试结果的平均值。
  - **留一法（LOOCV, Leave-One-Out Cross Validation）：**每次仅用 1 个样本作为测试集，剩余数据作为训练集，循环进行。

解决过拟合—正则化

- **正则化**：在损失函数中加入惩罚项，控制模型复杂度

$$\min_{\theta} \underbrace{\frac{1}{n}\sum_{i=1}^{n} L(y_i, g_\theta(x_i))}_{\text{经验风险}} + \underbrace{\lambda \cdot J(\theta)}_{\text{正则化}}$$

- **L1正则化（Lasso）**：在损失函数中惩罚所有模型参数绝对值的和，以惩罚大参数值。

$$J(\theta) = \sum_{t=1}^{T} |\theta_t|$$

- 实现压缩模型大小、**特征选择**（将一些不重要的特征的参数压缩为零）。
- 适用于线性回归、逻辑回归模型。

解决过拟合—正则化

- **正则化**：在损失函数中加入惩罚项，控制模型复杂度

$$\min_{\theta} \underbrace{\frac{1}{n}\sum_{i=1}^{n} L(y_i, g_\theta(x_i))}_{\text{经验风险}} + \underbrace{\lambda \cdot J(\theta)}_{\text{正则化}}$$

➢ **L2正则化（Ridge）**：在损失函数中添加所有模型参数的平方和，以惩罚大参数值。

$$J(\theta) = \sum_{t=1}^{T} \theta_t^2$$

- **让模型参数值更小、更均匀**，从而防止某些特征对模型的影响过大

解决过拟合—正则化

- 正则化符合奥卡姆剃刀(Occam's razor)原理。

  - 14世纪英国哲学家威廉·奥卡姆：“如无必要，勿增实体”，即“在所有能够解释现象的假设中，最简单的那个往往是最好的。”

  - **奥卡姆剃刀原理应用于模型选择**：在所有可能选择的模型中，能够很好地解释已知数据并且十分简单才是最好的模型，也就是应该选择的模型。→简单的假设/模型更具泛化能力，更不容易受到噪声和特例的影响。

模型评估方法

- **训练集（Training Set）：**用于训练模型，调整模型的参数，使其能够学习数据的特征。
  - 举例：在图像分类任务中，训练集包括大量带有标签的图片，模型通过这些数据不断优化其内部权重。
- **验证集（Validation Set）：**用于调优超参数和防止过拟合。
  - 在训练过程中，模型会在验证集上进行评估，以选择最优的超参数（如学习率、正则化系数等）、网络结构或算法。
- **测试集（Test Set）：**用于评估模型的最终性能以近似泛化误差。
  - 在训练完成并选择最佳模型后，测试集用于衡量模型的泛化能力，以确保其能在未见过的数据上表现良好。

若在测试集上调整超参数，可能会导致数据泄漏（Data Leakage），影响模型的真实泛化能力！

性能评估

- 不同任务往往采用不同的性能评估方式，
- 不同的性能评估指标往往会导致不同的评判结果

- 回归任务-均方误差：

$$E(f;D)=\frac{1}{m}\sum_{i=1}^{m}(f(\boldsymbol{x}_i)-y_i)^2$$

- 错误率和精度-分类任务
- 错误率

$$E(f;D)=\frac{1}{m}\sum_{i=1}^{m}\mathbb{I}(f(\boldsymbol{x}_i)\neq y_i)$$

- 精度

$$\begin{aligned}\mathrm{acc}(f;D) &= \frac{1}{m}\sum_{i=1}^{m}\mathbb{I}(f(\boldsymbol{x}_i)=y_i)\\ &= 1-E(f;D)\ .\end{aligned}$$

性能评估：查准率和查全率

- 二分类混淆矩阵（多分类为$N \times N$）

	实际为正类 (Positive)	实际为负类 (Negative)
预测为正类 (Positive)	TP（真正例）	FP（假正例）
预测为负类 (Negative)	FN（假负例）	TN（真负例）

- **TP（真正例，True Positive）：** 模型正确分类为正类的样本数。
- **FP（假正例，False Positive）：** 模型错误地将负类分类为正类的样本数（误报）。
- **FN（假负例，False Negative）：** 模型错误地将正类分类为负类的样本数（漏报）。
- **TN（真负例，True Negative）：** 模型正确地将负类分类为负类的样本数。

性能评估：查准率和查全率

- 二分类混淆矩阵

	实际为正类 (Positive)	实际为负类 (Negative)
预测为正类 (Positive)	TP（真正例）	FP（假正例）
预测为负类 (Negative)	FN（假负例）	TN（真负例）

- **查准率（precision）：在所有被模型预测为正类的样本中，实际为正类的比例。**

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

- **TP**：正确预测为正类的样本数；**FP**：被错误预测为正类的负类样本数。
- **查准率**反映了模型的**预测精确度**，即预测为正的样本中，真正为正的比例。
- **示例：** 在垃圾邮件分类中，高查准率意味着所有预测为垃圾邮件的邮件中，大部分确实是垃圾邮件。

性能评估：查准率和查全率

- 二分类混淆矩阵

	实际为正类 (Positive)	实际为负类 (Negative)
预测为正类 (Positive)	TP（真正例）	FP（假正例）
预测为负类 (Negative)	FN（假负例）	TN（真负例）

- **查全率（Recall）：在所有实际为正类的样本中，模型成功预测出的比例。**

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

- **TP**：正确预测为正类的样本数；**FN**：被被错误预测为负类的正类样本数。
- **查全率反映了模型的覆盖能力，即模型能找到多少真正的正类样本。**
- **示例**：在垃圾邮件分类中，高查全率意味着几乎所有垃圾邮件都被成功分类出来。

性能评估：F_1-Score

调和平均数

$$H(A,B)=\frac{2\times A\times B}{A+B}$$

- **F_1-Score：查准率和查全率的调和平均，用于在两者之间取得平衡（若查准率和查全率不平衡，其中一个值非常低，那么F_1-Score也会受到显著影响）。**

$$F1=2\times\frac{\text{Precision}\times\text{Recall}}{\text{Precision}+\text{Recall}}$$

- **当查准率和查全率存在权衡（如提高查全率会降低查准率），F1分数能衡量它们的综合表现。**
- **示例：**在自动驾驶中的行人检测，希望检测到尽可能多的行人（高Recall），使得模型误报路上的物体为行人（低Precision）。低 Recall 可能会导致撞到未检测到的行人，而低 Precision 可能会让汽车频繁误停，影响驾驶体验。使用 F1 分数找到最佳平衡点。

性能评估：F_1-Score

调和平均数

$$H(A,B)=\frac{2\times A\times B}{A+B}$$

- **F_1-Score：查准率和查全率的调和平均，用于在两者之间取得平衡。**

$$F1=2\times\frac{\text{Precision}\times\text{Recall}}{\text{Precision}+\text{Recall}}$$

$$\text{Precision}=\frac{\text{TP}}{\text{TP}+\text{FP}}\qquad\qquad\text{Recall}=\frac{\text{TP}}{\text{TP}+\text{FN}}$$

➢ **当查准率和查全率存在权衡（如提高查全率会降低查准率），F1分数能衡量它们的综合表现。**

为什么不用普通平均数（$\frac{Precision+Recall}{2}$）？

如果直接使用普通平均数，Precision 和 Recall 之间的极端不平衡情况可能会被掩盖。例如：

- Precision = 0.99, Recall = 0.1，普通平均值是 0.545，而 F1 分数是 0.18，反映出 Recall 低的影响。
- 这种情况下，F1 分数能更公平地反映 Precision 和 Recall 之间的折中。

性能评估：F_1-Score

- **F_1-Score：查准率和查全率的调和平均，用于在两者之间取得平衡（若查准率和查全率不平衡，其中一个值非常低，那么F_1-Score也会受到显著影响）。**

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

- **较小的 F_1-Score 可能表示模型的查准率或查全率较低，模型在查准率和查全率之间的平衡较差→需要进一步优化模型**
- **较大的 F_1-Score通常表示模型在查准率和查全率之间有较好的平衡，查全率和查准率都相对较高。**

性能评估：F_β-Score

- **F_β-Score：** **F1 分数的推广形式，它引入了一个权重因子 β，用于调整 Precision（查准率）和 Recall（查全率）之间的相对重要性。**

$$F_\beta = \frac{(1+\beta^2) \times Precision \times Recall}{\beta^2 \times Precision + Recall}$$

- **β**: 控制 **Recall** 的重要性，即**Recall**相对于 **Precision** 的权重。
  - **$\beta > 1$: 更关注recall（在分母recall的权重比precision小），适用于需要尽可能召回所有正例的任务，**如医学诊断（避免漏诊）。
  - **$\beta < 1$: 更关注Precision（在分母precision的权重比recall小），适用于需要降低误报率的任务，**如垃圾邮件分类（减少误判为垃圾邮件的正常邮件）。
  - **$\beta = 1$： F_β= F1，Precision和Recall同等重要。**

性能评估：偏差与方差

- **偏差（bias）：**给定一个样本x，偏差度量了模型预测的系统误差，它衡量的是模型的预测值$\hat{y} = f(x)$与真实值y之间的差异。

$$\mathbf{Bias} = \mathbb{E}[\hat{y}] - y$$

  - $\mathbb{E}[\hat{y}]$ ：即由于模型的预测可能会受到噪声和不同训练集的影响，因此对于同一个输入样本，模型的预测值通常是不同的。即，模型的预测会在不同的训练过程或不同的训练集上有所不同。$\mathbb{E}[\hat{y}]$ 考虑到所有可能的训练集后，模型在样本 x 上的平均预测。
  - **高偏差：模型的拟合能力差，比如模型过于简单、泛化能力差等。**
  - **低偏差：模型的预测值与真实值之间的差距较小，说明模型的拟合能力较强，能够较好地捕捉到数据中的模式或趋势。**

性能评估：偏差与方差

- **方差（variance）:模型在不同训练集上预测结果的波动程度，反映了模型的稳定性。**

$$\mathbf{Var} = \mathbb{E}[(\mathbb{E}[\hat{y}] - y)^2]$$

- $\mathbb{E}[\hat{y}]$:在所有训练集上对输入x的预测值的期望。
- $\mathbb{E}[(\mathbb{E}[\hat{y}] - y)^2]$：计算了所有不同训练集上的预测值与平均预测值之间的差异的平方。
- 当我们训练相同的模型，但使用不同的训练数据时，方差越大，意味着模型的预测结果在不同的数据集上差异越大，模型不稳定，容易受到训练集的影响。

性能评估：偏差与方差

- **方差（variance）:模型在不同训练集上预测结果的波动程度，反映了模型的稳定性。**

$$\mathbf{Var} = \mathbb{E}[(\mathbb{E}[\hat{y}] - y)^2]$$

- **高方差**：表示模型的预测结果对不同的训练数据集非常敏感。即在一个训练集上表现很好，但在另一个训练集上可能表现很差→在某个训练集上“过拟合”（Overfitting），学习到了数据中的噪声，而不是数据的真实规律。
- **低方差**： 模型的预测结果在不同训练集上变化不大，模型对训练数据的敏感性较低→模型的稳定性较好，能够在不同的训练集上保持相似的预测效果。


Thank You!