机器学习与深度学习
——K 最近邻（KNN）算法

Personal Website: https://www.miaopeng.info/
Email: miaopeng@stu.scu.edu.cn
Github: https://github.com/MMeowhite
Youtube: https://www.youtube.com/@pengmiao-bmm

目录章节

CONTENTS

01 KNN的概念与原理

02 模型求解与评估

03 模型扩展与改进

04 模型实现与实战

05 总结

▶ 为什么学KNN？

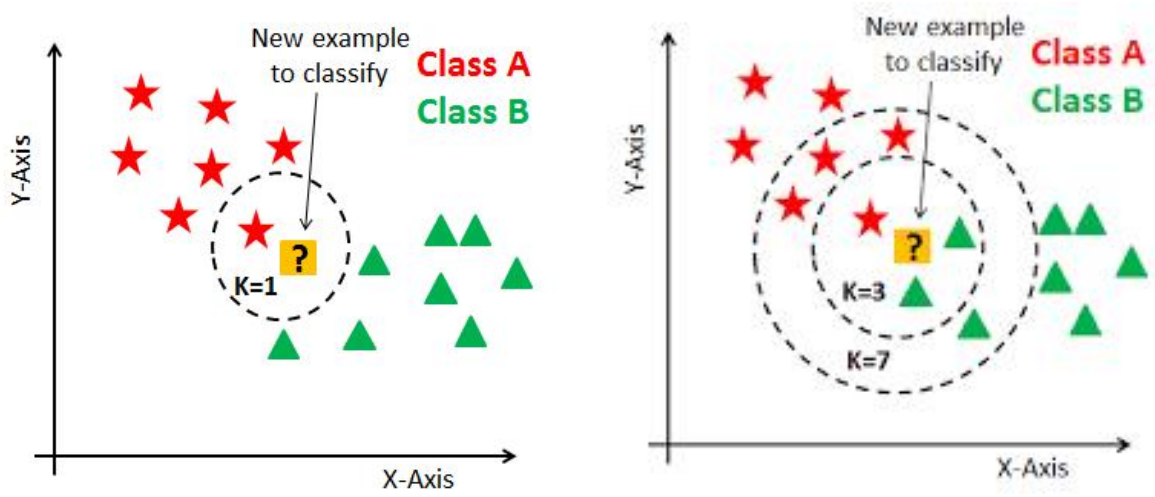
- KNN 的应用场景非常广泛，尤其是在数据规模不算特别大、模型可解释性要求高的任务中常见。可以从**分类、回归、推荐、异常检测**四大类来讲。
  - 分类任务：医学诊断、文本分类、图像识别等。
  - 回归任务：房价预测，股票预测等。
  - 推荐系统：用户兴趣推荐、电商商品推荐等。
  - 异常检测：信用卡欺诈检测、网络入侵检测等。

- KNN 因其**直观、无需显式训练、效果稳健**，可用于多种场景，尤其在**样本量不大**或**特征空间相对规整**时表现良好。

KNN 是机器学习的第三个台阶，代表了从“参数化模型”到“非参数化模型”的过渡。

什么是KNN？

- 定义：KNN（K-最近邻，K-Nearest Neighbors）是一种**基于距离**的**非参数化监督学习**算法，通过计算样本与训练集中样本的**相似度**（通常用欧式距离），**选取最近的 K 个邻居**，根据多数投票（分类，离散值）或平均值（回归，连续值）来进行预测。
- 本质：一种“基于实例”的懒惰学习算法。
- 决策规则：“物以类聚，人以群分，近朱者赤，近墨者黑”，预测靠最近的 K 个邻居。



source: https://github.com/Robots-Vision/knn-examples

- 1）基于实例的学习：KNN 不显式学习模型参数，而是**直接依赖训练数据**本身进行预测。
- 2）通过“距离”度量相似性，对新样本，KNN 计算其与训练集中所有样本的距离，找到最近的K个邻居。
- 3）多数投票 / 平均值预测：分类任务：由邻居中出现最多的类别决定；回归任务：由邻居的数值平均（或加权平均）决定。

逻辑回归的数学表达形式

- KNN 数学定义：假设训练集为：

$$\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}, \quad x_i \in \mathbb{R}^d, y_i \in \mathcal{Y}$$

  - 对于一个新样本x，KNN的预测分为两种情况：
  - 1）分类任务 (majority voting)：
    - 计算与所有训练样本的距离（常用欧式距离）：

$$d(x, x_i) = \sqrt{\sum_{j=1}^{d} (x_j - x_{ij})^2}$$

    - 选取最近的K个邻居：

$$N_k(x_i) = \{(x_i, y_i) \in \mathbb{D} | x_i \text{ is the K nearest neighbors of x}\}$$

    - 预测类别：

$$\hat{y} = \arg\max_{c \in \mathbb{Y}} \sum_{(x_i, y_i) \in N_k(x)} \mathbf{1}(y_i = c)$$

  - 2）回归任务 (average of neighbors)：
    - 首先同样找到最近的K个邻居，然后预测值是邻居的均值：

$$\hat{y} = \frac{1}{K} \sum_{(x_i, y_i) \in N_K(x)} y_i$$

    - 也可以做加权平均（距离越近，权重越大）：

$$\hat{y}(x) = \frac{\sum_{i=1}^{K} w_i y_i}{\sum_{i=1}^{K} w_i}, \quad w_i = \frac{1}{d(x, x_i) + \epsilon}$$

注：w_i：权重，距离越小，权重越大

ϵ：防止分母为零的微小常数

▶ 扩展：距离度量方式

- 欧式距离（Euclidean Distance）：

$$d(x,y)=\sqrt{\sum(x_i-y_i)^2}$$

  - 直观理解：几何上的“直线距离”，最常见。举例：二维平面上两点之间的直线。

- 曼哈顿距离（Manhattan/L1 Distance）：

$$d(x,y)=\sum|x_i-y_i|$$

  - 直观理解：像走棋盘的格子，不能走斜线。举例：出租车在城市街道上走的路径。

- 切比雪夫距离 (Chebyshev Distance)：

$$d(x,y)=\max_i|x_i-y_i|$$

  - 直观理解：棋盘上“国王走法”的最短步数。

- 切比雪夫距离 (Chebyshev Distance)：

$$d(x,y)=(\sum|x_i-y_i|^p)^{\frac{1}{q}}$$

  - 一个更通用的形式，欧式、曼哈顿、切比雪夫都是它的特例。

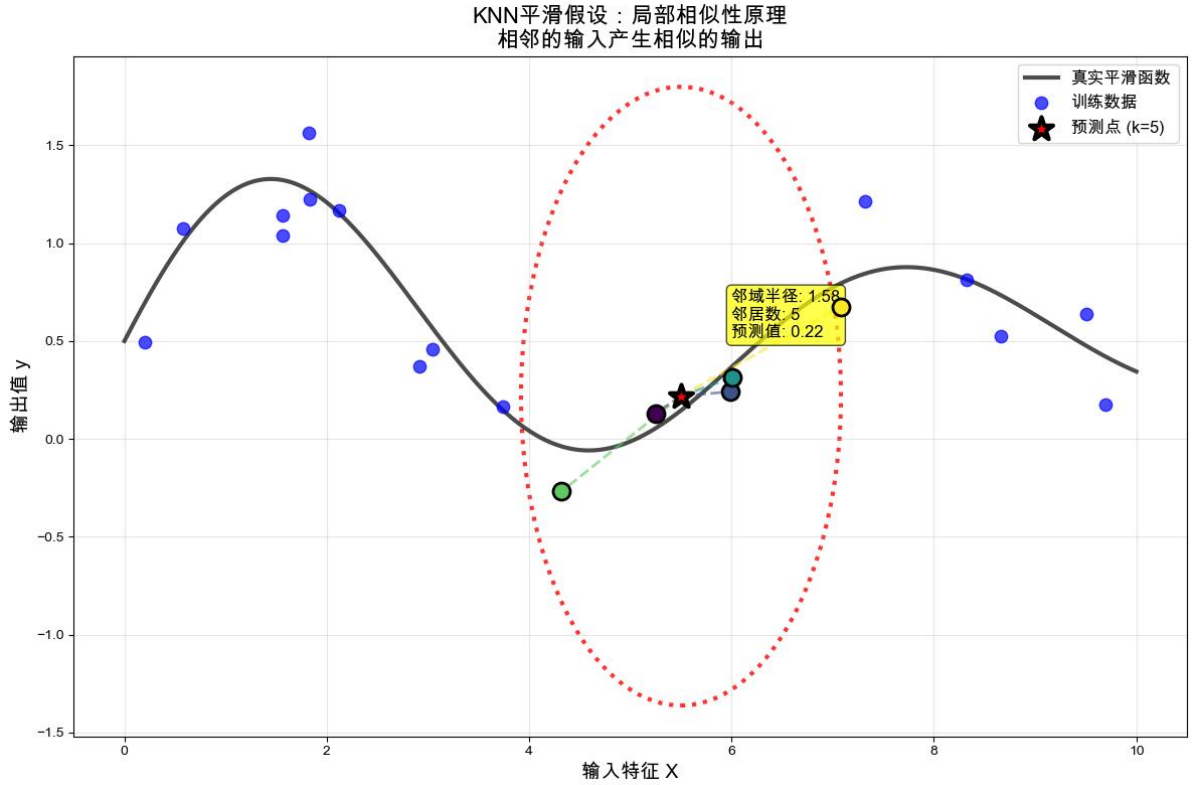
- 余弦相似度 (Cosine Similarity) （常见于文本/高维稀疏数据）：

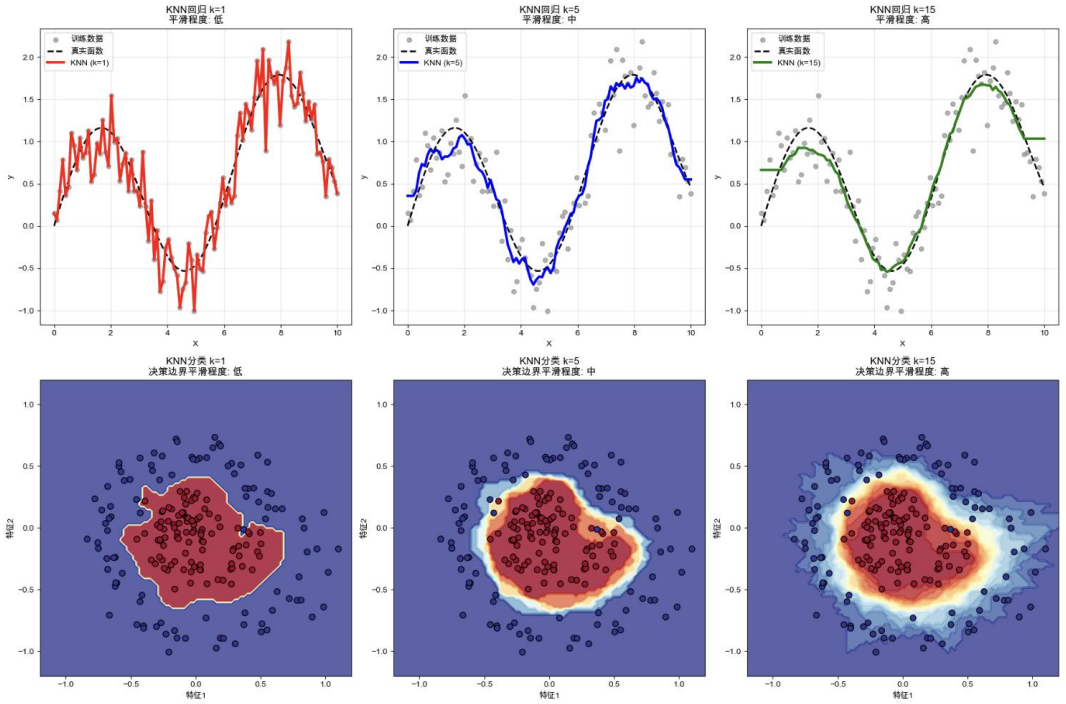
$$\mathrm{CosSim}(x,y)=\frac{x\cdot y}{\|x\|\,\|y\|}=\frac{\sum_{i=1}^n x_iy_i}{\sqrt{\sum_{i=1}^n x_i^2}\,\sqrt{\sum_{i=1}^n y_i^2}}$$

  - 不关注绝对大小，而关注两个向量的“方向”。特点：适合比较角度，比如用户兴趣、文本主题。

▶ 模型假设（非常重要！）

- 逻辑回归模型的假设类似于线性回归模型的假设，如果不满足以下的假设，那么模型拟合效果肯定不佳：
  - 1）**相似的输入有相似的输出**：目标变量y的**对数几率**与自变量之间存在线性关系：

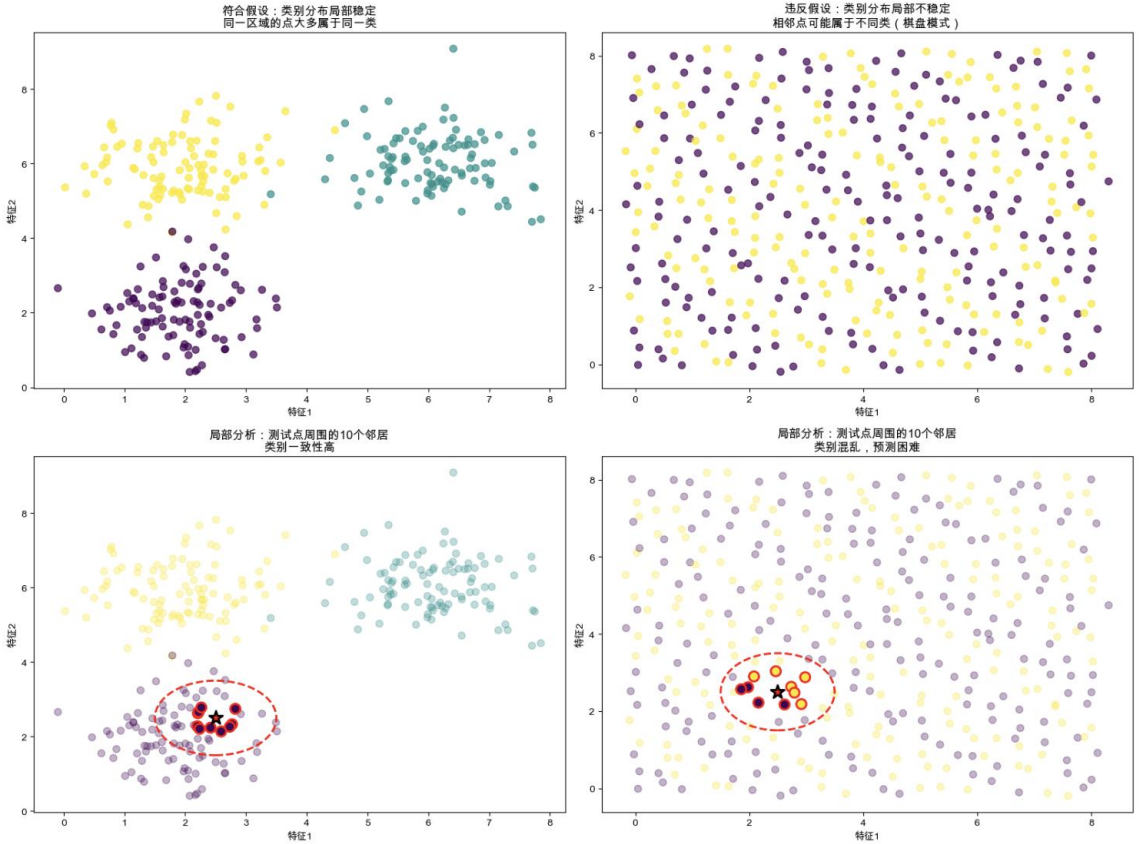




  - 直观说：在特征空间中，**如果两个样本点距离很近，它们的类别/目标值也应该接近**。这是 KNN 的核心平滑假设（local smoothness assumption）。

▶ 模型假设（非常重要！）

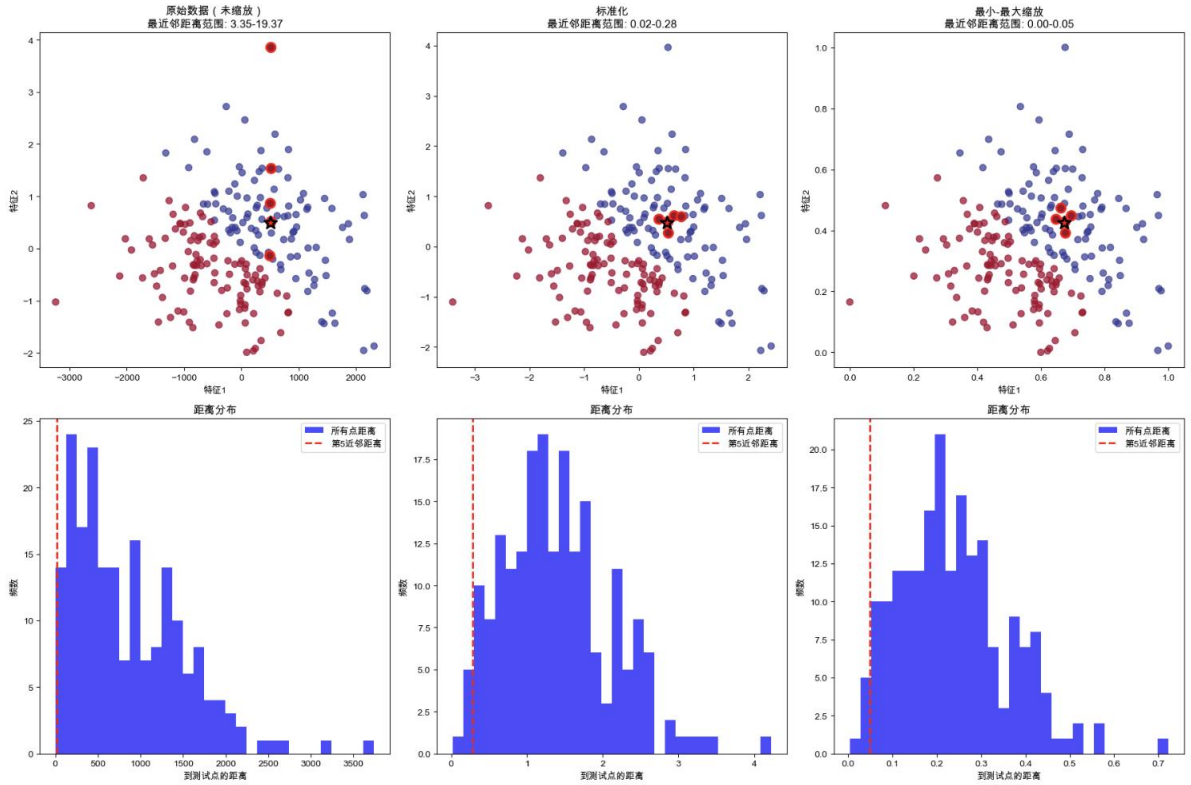
- 逻辑回归模型的假设类似于线性回归模型的假设，如果不满足以下的假设，那么模型拟合效果肯定不佳：
  - 2）**类别分布相对稳定**：类别分布在局部是相对稳定的，一个点的类别可以用它邻居的类别来代表。



  - 举例：假如一片区域大部分点是“猫”，那新落在这片区域的点也大概率是“猫”。

▶ 模型假设（非常重要！）

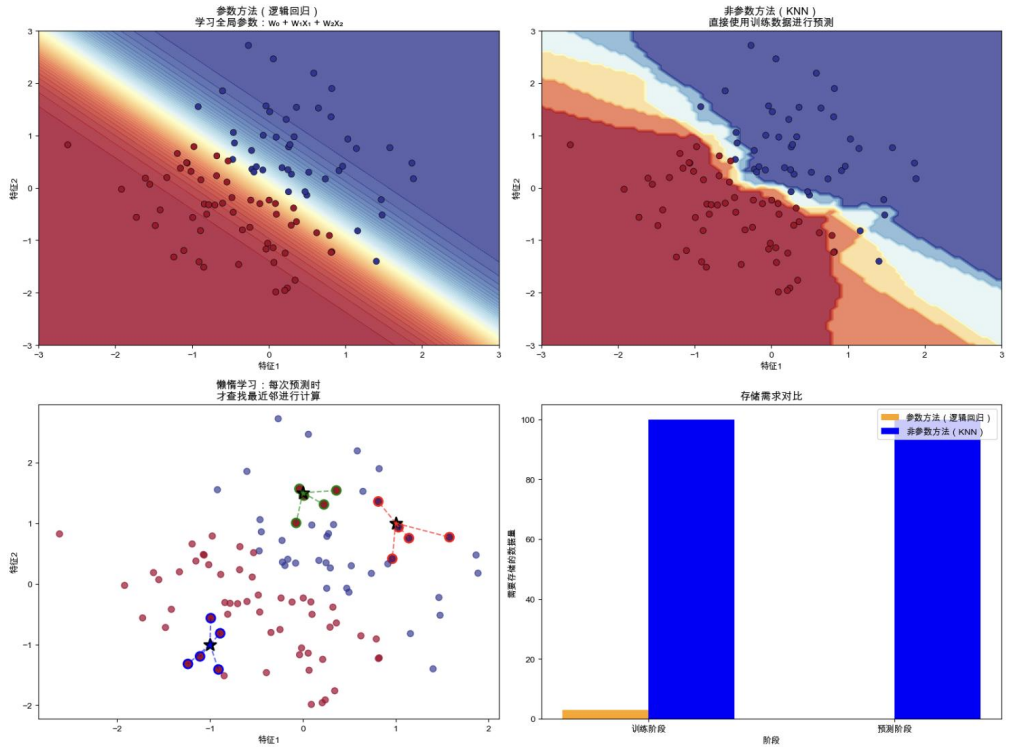
- 逻辑回归模型的假设类似于线性回归模型的假设，如果不满足以下的假设，那么模型拟合效果肯定不佳：
  - 3）**特征空间的度量是有意义的**：使用的距离度量方式（欧式、曼哈顿、余弦等）能够合理反映样本之间的相似性。



  - 特征必须经过合适的预处理（缩放、标准化），否则“近”就不一定真的有意义。

▶ 模型假设（非常重要！）

- KNN的模型假设类似于线性回归模型的假设，如果不满足以下的假设，那么模型拟合效果肯定不佳：
  - 4）**隐含的非参数假设**：KNN不去学习一个显式的参数化函数，而是假设数据分布可以通过**局部邻域直接近似**而非全局的特征。



  - 属于“懒惰学习”（lazy learning）方法，不提前建模，只在预测时利用训练数据。同时也能节省训练时使用内存

▶ 小结

- 为什么需要学习KNN（K-近邻算法）？
  - 学习 KNN 有助于理解基于实例的非参数方法，掌握简单直观的分类与回归思路，并为处理高维、非线性或无模型假设的数据提供基础工具。
- 什么是KNN（K-近邻算法）？
  - KNN（K-近邻算法）是一种基于实例的非参数算法，通过计算待预测样本与训练样本的距离找到最近的 K 个邻居，并根据邻居标签投票（分类）或平均（回归）进行预测。
  - 简单直观，不依赖模型假设，但计算量大且对特征尺度敏感。
- 模型假设：
  - 相似的输入有相似的输出。
  - 特征空间的度量方式具有意义。
  - 类别分布相对稳定。
  - 隐含非参数假设。

学习 KNN 可以掌握一种基于实例的非参数算法，它通过寻找最近邻居进行分类或回归预测，不依赖线性或分布假设，仅假设相似样本具有相似输出且特征空间可度量距离。

目录章节

CONTENTS

01 KNN的概念和原理

02 模型求解与评估

03 模型扩展与改进

04 模型实现与实战

05 总结

▶ 模型求解：分类预测（Classification）

- 假设：
  - 记训练集$\{(x_i,y_i)\}$，i=1,...,n，$y_i \in \{1,2,...,C\}$。新样本为：x_{new}。
- 求解步骤：
  - 1）计算距离：

$$d(x_{new}, x_i) \quad \forall i = 1, 2, \cdots, n$$

  - 2）选取最近的K个邻居：

$$N_K(x_{new}) = \text{indices of the k closet points to } x_{new}$$

  - 3）投票预测类别：

$$\hat{y} = \arg \max_{c \in \{1,\cdots,C\}} \sum_{i \in N_K(x_{new})} \mathbf{1}(y_i = c)$$

  - 可加权投票（距离越近权重越大）

$$\hat{y} = \arg\max_c \sum_{i \in N_k(x_{new})} w_i \mathbf{1}(y_i = c), \quad w_i = \frac{q}{d(x_{new}, x_i)}$$

▶ 模型求解：回归预测（Regression）

➢ 假设：

- 记训练集$\{(x_i,y_i)\}$，i=1,...,n，$y_i \in R$。新样本为：x_{new}。

➢ 求解步骤：

- 1）计算距离：

$$d(x_{new}, x_i) \quad \forall i = 1, 2, \cdots, n$$

- 2）选取最近的K个邻居：

$$N_K(x_{new}) = \text{indices of the k closet points to } x_{new}$$

- 3）投票预测类别：

$$\hat{y} = \frac{1}{k} \sum_{i \in N_K(x_{new})} y_i$$

- 加权平均：

$$\hat{y} = \frac{\sum_{i \in N_k(x_{new})} w_i y_i}{\sum_{i \in N_k(x_{new})} w_i}, \quad w_i = \frac{q}{d(x_{new}, x_i)}$$

模型评估

- 由于KNN模型可以同时处理分类问题和回归问题，以及唯一的超参数为k，所以指标也就是分类问题的评估和回归问题的评估。
  - 回归问题的评估：均方误差（MSE）、均方根误差（RMSE）、平均绝对误差（MAE）。

指标名称	缩写	公式	含义说明
均方误差	MSE	$MSE = \frac{1}{m}\sum_{i=1}^{m}(\hat{y}_i - y_i)^2$	平均预测误差的平方，衡量整体误差
均方根误差	RMSE	$RMSE = \sqrt{MSE}$	MSE 的平方根，单位与原数据一致，更直观
决定系数	R^2	$R^2 = 1 - \frac{\sum(\hat{y}_i - y_i)^2}{\sum(\hat{y}_i - \bar{y})^2}$	表示模型对结果方差的解释能力，越接近1越好

KNN 的评估核心是验证其预测准确性与局部邻居拟合效果，常用指标包括分类问题的准确率、精确率、召回率、F1、AUC，以及回归问题的均方误差（MSE）、均方根误差（RMSE）、平均绝对误差（MAE）等。

模型评估

- 由于KNN模型可以同时处理分类问题和回归问题，以及唯一的超参数为k，所以指标也就是分类问题的评估和回归问题的评估。
  - 分类问题的评估：准确率、精准率、召回率、F1分数、ROC-AUC

类别	指标名称	公式	说明
分类性能指标	准确率	$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$	正确预测样本数 / 总样本数，整体正确性。
	精确率	$Precision = \frac{TP}{TP + FP}$	在预测为正类的样本中，真正为正类的比例。
	召回率/灵敏度	$Recall = \frac{TP}{TP + FN}$	在实际正类样本中，被正确预测为正类的比例。
	F1分数	$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$	精确率与召回率的调和平均，综合考量模型性能。
	ROC-AUC	$\mathrm{ROC} : t \mapsto (\mathrm{FPR}(t), \mathrm{TPR}(t)), \quad t \in [0, 1]$	反映模型区分正负样本的能力，曲线下的面积越大越好。

KNN 的评估核心是验证其预测准确性与局部邻居拟合效果，常用指标包括分类问题的准确率、精确率、召回率、F1、AUC，以及回归问题的均方误差（MSE）、均方根误差（RMSE）、平均绝对误差（MAE）等。

▶ 小结

- 逻辑回归模型的求解主要包含三种方法：梯度下降法、牛顿法、拟牛顿法。
  - 梯度下降法：通过迭代沿着目标函数负梯度方向更新参数，逐步逼近最优解，计算简单但收敛速度较慢。

$$\beta \leftarrow \beta - \lambda \sum_{i \in \mathcal{B}} (p_i - y_i) x_i$$

  - 牛顿法：利用一阶和二阶导数信息直接迭代更新参数，收敛速度快，但需要计算 Hessian 矩阵，代价较高。

$$\beta^{(t+1)} = \beta^{(t)} - (\mathbf{X}^T \mathbf{W} \mathbf{X})^{-1} \mathbf{X}^T (\mathbf{p} - \mathbf{y})$$

  - 拟牛顿法：在牛顿法基础上用近似的 Hessian 替代真实二阶导数，兼顾计算效率与收敛速度。

$$H_{k+1} = \left(I - \frac{s_k y_k^T}{y_k^T s_k} \right) H_k \left(I - \frac{y_k s_k^T}{y_k^T s_k} \right) + \frac{s_k s_k^T}{y_k^T s_k}$$

线性回归可用正规方程法或梯度下降法求解，模型好坏可用 MSE、RMSE 衡量误差大小，用R^2衡量解释能力。

小结

- 逻辑回归模型的评估主要包含两类指标：分类性能指标和概率质量指标。

类别	指标名称	公式	说明
分类性能指标	准确率	$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$	正确预测样本数 / 总样本数，整体正确性。
	精确率	$Precision = \frac{TP}{TP + FP}$	在预测为正类的样本中，真正为正类的比例。
	召回率/灵敏度	$Recall = \frac{TP}{TP + FN}$	在实际正类样本中，被正确预测为正类的比例。
	F1分数	$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$	精确率与召回率的调和平均，综合考量模型性能。
	ROC-AUC	$\mathrm{ROC} : t \mapsto (\mathrm{FPR}(t), \mathrm{TPR}(t)), \quad t \in [0, 1]$	反映模型区分正负样本的能力，曲线下的面积越大越好。
概率质量指标	对数似然 (Log-Loss)	$\mathrm{LogLoss} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(p_i) + (1 - y_i)\log(1 - p_i)]$	度量预测概率与真实标签的差异，值越小说明概率预测越准确。
	Brier Score	$\text{Brier Score} = \frac{1}{N}\sum_{i=1}^{N}(p_i - y_i)^2$	预测概率与真实标签之间的均方误差，越小越好。

线性回归可用正规方程法或梯度下降法求解，模型好坏可用 MSE、RMSE 衡量误差大小，用R^2衡量解释能力。

目录章节

CONTENTS

01 KNN的概念和原理

02 模型求解与评估

03 模型扩展与改进

04 模型实现与实战

05 总结

▶ 模型扩展与改进——距离度量优化

- 1）**自适应距离**
  - 概念：不同特征对结果的重要性不同，直接用欧氏距离可能不合理。自适应距离会给重要特征更高权重。

$$d(x_i, x_j) = \sqrt{\sum_{k=1}^{n}(x_{ik} - x_{jk})^2}$$

  - 意义：权重与特征的重要性有关，加强特征的作用。
- 2）**学习距离度量**
  - 概念：直接训练一个模型，使得“同类样本距离近，不同类样本距离远”。
  - 方法：LMNN（Large Margin Nearest Neighbor）：通过优化使得最近邻属于同类，异类尽量远；Siamese Network：神经网络学习特征映射，将欧氏距离映射到更合适的空间。
- 3）**非欧式距离**
  - 余弦相似度（对文本或高维稀疏向量非常有效）：

$$\cos(x_i, x_j) = \frac{x_i \cdot x_j}{||x_i|| ||x_j||}$$

  - 马氏距离（考虑特征之间的协方差，适合相关性强的特征）等：

$$d_M(x_i, x_j) = \sqrt{(x_i - x_j)^T S^{-1}(x_i - x_j)}$$

▶ 模型扩展与改进——权重策略改进

➢ 1）距离加权 KNN

- 距离越近权重越大：

$$\hat{y} = \frac{\sum_{i \in N_k(x_{new})} w_i y_i}{\sum_{i \in N_k(x_{new})} w_i}, \quad w_i = \frac{q}{d(x_{new}, x_i)}$$

- 意义：距离越小，权重越大。
- 应用场景：KNN 回归时，邻居距离差异较大；距离近的样本更可靠，因此用距离加权可以让预测更准确。

➢ 2）核函数加权

- 使用高斯核等函数平滑邻居影响：

$$w_i = \exp(-\frac{d(x_{new}, x_i)^2}{2\sigma^2})$$

- 优点：避免距离过远的邻居产生噪声，同时权重变化平滑。
- 应用场景：核函数可以平滑邻居的影响，避免某个异常近邻过度影响结果；当数据密度不均匀时，核函数权重可自动调节邻居影响，保留局部结构；核权重能输出平滑的概率分布，而不仅仅是“硬投票”，适合概率预测场景。

▶ 模型扩展与改进——高维与特征处理

- **1）特征缩放/标准化**
  - 避免不同量纲影响距离：
  - 方法：a.Min-Max标准化：$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$　　b.Z-score标准化：$x' = \frac{x - \mu}{\sigma}$
- **2）先降维，后KNN**
  - 动机：高维空间中样本间距离趋于相似，距离失去区分性 → KNN 精度下降，同时由于KNN依赖距离度量，高维计算距离开销大 → 速度慢。因此先使用PCA、t-SNE、UMAP等减少特征维度，提高效果和速度。
  - 注意：信息损失：降维可能丢掉一些细节，KNN精度可能稍微下降，需要交叉验证调参；标准化重要性：PCA 对量纲敏感，KNN 对距离敏感，标准化是必须步骤；降维维度选择：维度过低 → 丢失重要信息；维度过高 → 高维问题仍存在。非线性降维适用性：t-SNE 更适合可视化，如果用于 KNN 特征，需要验证邻居关系是否保持。
- **3）特征选择**
  - 标准：统计指标、模型驱动、信息理论。
  - 核心目的：保留最有用的特征、去掉冗余或噪声特征，提高模型性能、降低计算复杂度。

▶ 模型扩展与改进——算法效率优化（了解）

- 1）数据结构优化
  - KD 树：适合低维数据，分割空间，加速邻居搜索。
  - Ball 树：适合中高维数据，使用球形分区减少搜索。
- 2）**近似 KNN**
  - 主要针对于高维数据，计算开销较大的场景。
  - 近似KNN算法：LSH（Locality Sensitive Hashing），Annoy / Faiss等。
- 3）**并行计算：**
  - 在大数据集上，距离计算可以分布式或多线程并行处理，显著提升速度。

▶ 小结

➢ KNN 的主要通过优化距离度量、邻居权重和特征处理来提升预测准确性与鲁棒性，同时借助降维、近似搜索和并行计算等方法提高算法效率，从而适应高维与大规模数据场景。

类别	方法	说明	典型应用场景
距离度量优化	自适应距离	特征赋予不同权重	不同量纲、特征重要性差异大
	学习距离度量（LMNN、Siamese）	训练得到更合适的度量方式	图像识别、度量学习任务
	非欧氏距离（余弦/马氏）	余弦 → 文本/稀疏数据；马氏 → 特征相关性强	文本挖掘、相关性强的特征集
权重策略改进	距离加权KNN	距离越近，权重越大，远邻居影响小	回归（房价、气象）、模糊边界分类
	核函数加权 KNN	高斯核平滑邻居影响，降低噪声	概率估计、推荐系统、医疗预测
高维与特征处理	特征缩放/标准化	避免不同量纲影响距离	混合特征（收入 vs 面积等）
	降维（PCA、t-SNE、UMAP）	去除冗余特征，缓解维度灾难	高维数据、可视化、加速计算
	特征选择	去掉低方差、低相关、低重要性特征	数据清洗、提升泛化能力
算法效率优化	数据结构（KD 树、Ball 树）	空间分区，加速邻居搜索	低维/中维数据集
	近似 KNN（LSH、Faiss、Annoy）	牺牲少量精度换取大幅加速	高维向量检索、推荐系统
	并行计算	多线程/分布式计算距离	大规模数据集

KNN 的精髓在于“找对距离、选好邻居、简化特征、加速计算”，这样才能既准又快。

目录章节

CONTENTS

01 KNN的概念和原理

02 模型求解与评估

03 模型扩展与改进

04 模型实现与实战

05 总结

▶ 算法实现：numpy实现

➢ 实现：MyKNN类，第一步：先定义类及初始化方法及内部（私有_）方法。

```python
class MyKNN:
    def __init__(self, n_neighbors=5, metric="euclidean",
                 weights="uniform", sigma=1.0):
        self.n_neighbors = n_neighbors
        self.metric = metric
        self.weights = weights
        self.sigma = sigma

    def _distance(self, x1, x2):
        if self.metric == "euclidean":
            return np.sqrt(np.sum((x1 - x2) ** 2))
        elif self.metric == "manhattan":
            return np.sum(np.abs(x1 - x2))
        elif self.metric == "cosine":
            num = np.dot(x1, x2)
            den = (np.linalg.norm(x1) * np.linalg.norm(x2)) + 1e-8
            return 1 - num / den
        else:
            raise ValueError("Unsupported metric.")

    def _get_weights(self, distances):
        if self.weights == "uniform":
            return np.ones_like(distances)
        elif self.weights == "distance":
            return 1 / (distances + 1e-8)
        elif self.weights == "kernel":
            return np.exp(- (distances ** 2) / (2 * self.sigma ** 2))
        else:
            raise ValueError("Unsupported weight type.")
```

- 类定义及初始化：可选参数包括邻居数量、距离度量方式、权重策略、高斯核带宽。
- _distance方法（_表示私有）：根据距离度量方式选择实现距离测量方式，包括'euclidean'（欧氏）、'manhattan'（曼哈顿）、'cosine'（余弦）等。
- _get_weights方法（_表示私有）：根据权重策略实现邻居权重，包括'uniform' 全等权，'distance' 距离越近权重越大，'kernel' 用高斯核平滑距离。

算法实现：numpy实现

➢ 实现：MyKNN类，第二步：实现核心的模型训练和预测方法：fit(X, y)，predict(X)。

```python
def fit(self, X, y):
    """ 注：KNN是惰性学习，不训练，只保存训练数据 """
    self.X_train = np.array(X)
    self.y_train = np.array(y)

def predict(self, X):
    X = np.array(X)
    y_pred = []

    for x in X:
        # 计算与所有训练样本的距离
        distances = np.array([self._distance(x, x_train) for x_train in self.X_train])
        # 选出前k个邻居
        idx = np.argsort(distances)[:self.n_neighbors]
        neighbor_labels = self.y_train[idx]
        neighbor_distances = distances[idx]

        # 计算权重
        weights = self._get_weights(neighbor_distances)

        # 分类任务：加权投票
        label_weight = {}
        for label, w in zip(neighbor_labels, weights):
            label_weight[label] = label_weight.get(label, 0) + w # 默认权重为0

        # 选择权重和最大的类别
        y_pred.append(max(label_weight, key=label_weight.get))

    return np.array(y_pred)

def score(self, X, y):
    """ 计算准确率 """
    y_pred = self.predict(X)
    return np.mean(y_pred == y)
```

- KNN 是惰性学习（Lazy Learning），不需要像逻辑回归或神经网络那样拟合参数。
- 预测时直接用这些训练样本计算距离和寻找邻居。
- KNN预测流程：遍历每个测试样本->计算与所有训练样本的距离->选取前 k 个最近邻->根据权重策略计算邻居贡献->投票或加权投票决定预测类别->输出所有预测结果

▶ 算法实现：numpy实现

➢ 实现：MyLogisticRegression类，第三步：实现评估方法。

```
def score(self, X, y):
    """ 计算准确率 """
    y_pred = self.predict(X)
    return np.mean(y_pred == y)
```

- 这里只实现了分类的作用，所以只实现分类的准确率指标。

算法实现：pytorch实现（练习）

利用Pytorch实现关键步骤：

```python
# PyTorch 实现KNN

import torch

class MyKNN_PyTorch:
    def __init__(self, n_neighbors=5, metric="euclidean",
                 weights="uniform", sigma=1.0):
        self.n_neighbors = n_neighbors
        self.metric = metric
        self.weights = weights
        self.sigma = sigma
        self.device = 'cpu' if not torch.cuda.is_available() else 'cuda'

    def _distance(self, x1, x2):
        if self.metric == "euclidean":
            pass
        elif self.metric == "manhattan":
            pass
        elif self.metric == "cosine":
            pass
        else:
            raise ValueError("Unsupported metric.")

    def _get_weights(self, distances):
        if self.weights == "uniform":
            pass
        elif self.weights == "distance":
            pass
        elif self.weights == "kernel":
            pass
        else:
            raise ValueError("Unsupported weight type.")

    def fit(self, X, y):
        """ 注: KNN是惰性学习, 不训练, 只保存训练数据 """
        self.X_train = torch.tensor(X, dtype=torch.float32, device=self.device)
        self.y_train = torch.tensor(y, dtype=torch.int64, device=self.device)

    def predict(self, X):
        X = torch.tensor(X, dtype=torch.float32, device=self.device)
        y_pred = []

        for x in X:
            pass

        return torch.tensor(y_pred, dtype=torch.int64, device=self.device)

    def score(self, X, y):
        """ 计算准确率 """
        y_pred = self.predict(X)
        return (y_pred == torch.tensor(y, dtype=torch.int64, device=self.device)).float().mean().
```

- 任务1：完成距离度量方法，包括包括'euclidean'（欧氏）、'manhattan'（曼哈顿）、'cosine'（余弦）三种方法。
- 任务2：完成权重调整方法，包括'uniform' 全等权，'distance' 距离越近权重越大，'kernel' 用高斯核平滑距离。
- 任务3：完成核心的KNN算法，具体可以参考Numpy实现。

▶ 算法实战：手写数字识别

➢ 如之前一样，首先第一步就是导入数据，并查看数据的情况，决定下一步该干什么。

```
from sklearn.datasets import fetch_openml

# 下载MNIST数据集
mnist = fetch_openml('mnist_784', version=1, as_frame=False, data_home=".")
X, y = mnist.data, mnist.target.astype(int)

print("数据集大小:", X.shape, y.shape) # 28 x 28 = 784个像素点
```

```
数据集大小: (70000, 784) (70000,)
```

- MNIST（Modified National Institute of Standards and Technology）是一个经典的手写数字图像数据集，用于机器学习和计算机视觉的基础实验。
- 内容：包含 70,000 张灰度图像（28×28=784 像素），数字范围为 0－9。每张图像已归一化到固定大小，且中心对齐，便于算法训练。

▶ 算法实战：房价预测

➢ 根据上面的分析，进行数据预处理，并进行数据集划分（训练集、测试集）。

```
from sklearn.model_selection import train_test_split
from sklearn.decomposition import PCA

# 划分数据集
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 数据标准化
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# 降维 (PCA -> 保留 95% 方差)
pca = PCA(0.95)
X_train = pca.fit_transform(X_train)
X_test = pca.transform(X_test)
print(f"降维后维度: {X_train.shape[1]}")
```

```
降维后维度: 330
```

- 由于784个像素点维度稍微有点高，计算代价有点大，所以我们先用PCA进行降维。可见使用PCA降维之后784维的数据变成了330维。

▶ 算法实战：房价预测

➢ 定义模型进行训练、预测，最后通过**准确率、召回率、F1分数等**指标评估效果。

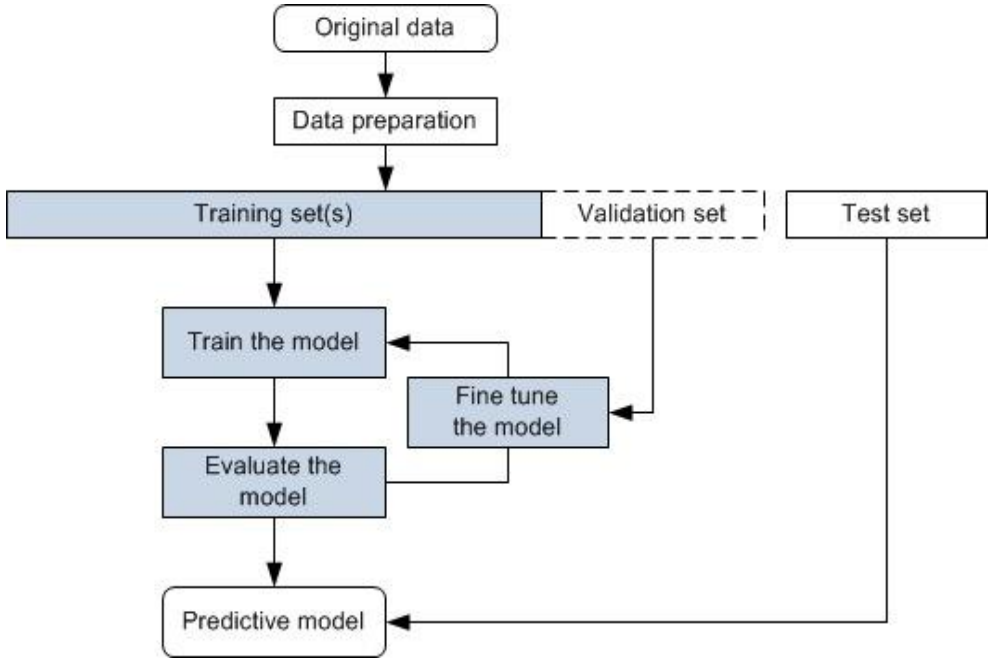
```python
from sklearn.neighbors import KNeighborsClassifier

# 建立 KNN 模型
knn_sklearn = KNeighborsClassifier(
    n_neighbors=5,        # 选 5 个邻居
    metric="euclidean",   # 距离度量方式（可选 manhattan, minkowski 等）
    weights="distance"    # 权重策略（uniform=等权, distance=距离加权）
)

# 训练
knn_sklearn.fit(X_train, y_train)

# 评估准确率
acc = knn_sklearn.score(X_test, y_test)
print(f"Sklearn KNN 手写数字识别准确率: {acc:.4f}")
```

```
Sklearn KNN 手写数字识别准确率: 0.9509
```

● 准确率为0.95，效果较好。

▶ 小结

- 模型实现：
  - 主要的方法：初始化参数（__init__），拟合方法（fit），预测方法（predict），评估指标（score），实现这些方法的私有方法（_xxx）。
- 模型实战：
  - 数据准备：导入数据、清洗缺失值、特征选择/编码。
  - 数据划分：训练集 / 测试集（常用70%/30%或80%/20%）。
  - 特征预处理：标准化/归一化/去NULL值/独热编码等。
  - 模型训练：KNN 是惰性学习，训练只需保存样本，预测时通过距离找邻居并投票或加权输出。
  - 模型评估：通过准确率，精准率，召回率，F1分数等进行评估。



KNN 的实现核心是保存训练样本和标签，在预测时通过计算距离找到最近邻，并根据权重策略投票或平均输出预测结果，同时通过评价指标评估模型性能。

目录章节

CONTENTS

01 KNN的概念和原理

02 模型求解与评估

03 模型扩展与改进

04 模型实现与实战

05 总结

总结

- **KNN（K-近邻算法）的概念与原理**
  - ✓ KNN 是一种基于实例的监督学习算法，通过测量样本之间的距离来进行分类或回归。KNN 简单直观，不需要训练模型，但计算量大且对特征尺度敏感。
  - ✓ 对于待预测样本，算法找到训练集中与其最相似的 K 个邻居，并根据这些邻居的标签进行投票（分类）或平均（回归）。
  - ✓ 模型假设：1）相似性假设；2）距离可度量；3）特征尺度一致；4）隐含的非参数假设。
- **KNN（K-近邻算法）的求解与评估**
  - ✓ KNN 通过计算待预测样本与训练样本的距离，找出最近的 K 个邻居，并根据邻居标签投票或平均来进行预测；由于 KNN 可以同时用于分类和回归任务，其评估指标也分为两类：分类问题常用准确率、精确率、召回率和 F1 值，回归问题常用均方误差（MSE）、均方根误差（RMSE）和平均绝对误差（MAE）。
- **KNN（K-近邻算法）的扩展与改进**
  - ✓ 线性回归的扩展与改进主要有：1）距离度量优化；2）权重策略改进；3）高维与特征处理；4）算法效率优化。

KNN 通过计算待预测样本与训练样本的距离找到最近的 K 个邻居，并根据邻居标签投票或平均实现分类或回归预测，可通过分类与回归指标评估性能，同时可通过选择距离度量、K 值和加权策略扩展以增强适用性。

感谢聆听

Personal Website: https://www.miaopeng.info/

Email: miaopeng@stu.scu.edu.cn

Github: https://github.com/MMeowhite

Youtube: https://www.youtube.com/@pengmiao-bmm