机器学习算法伪代码记录

中南大学 Junyi Fang

2025 年 1 月

1 引言

本文将记录一些常见的机器学习算法的核心思想和伪代码，以便在上课和上机实验过程中进行参考和回顾。注：其中梯度提升算法和弹性网络算法虽然上课没讲到，但是在上机中需要对算法进行调参评优，故自主学习记录。

2 目录

- 多层感知器 (MLPClassifier)
- 朴素贝叶斯 (GaussianNB)
- AdaBoost 分类 (AdaBoostClassifier)
- 岭回归 (Ridge)
- Lasso 回归 (Lasso)
- 弹性网络 (ElasticNet)*
- 随机森林回归 (RandomForestRegressor)
- 梯度提升回归 (GradientBoostingRegressor)
- 自适应提升回归 (AdaBoostRegressor)
- 支持向量回归 (SVR)
- 决策树回归 (DecisionTreeRegressor)

3 算法伪代码

Algorithm 1: 梯度下降算法（动量 + RMSprop）

Input: 学习率 α，动量系数 β，数值稳定性 ϵ

初始化：momentum $= 0$，squared_grads $= 0$;

定义损失函数：$L(x, y) = \sin(x) \cdot \cos(y)$;

定义梯度：

$$\frac{\partial L}{\partial x} = \cos(x) \cdot \cos(y), \quad \frac{\partial L}{\partial y} = -\sin(x) \cdot \sin(y)$$

训练过程：**for** 每个训练步骤 $t = 1, 2, \ldots, T$ **do**

- 计算梯度 $\nabla L = \left[\frac{\partial L}{\partial x}, \frac{\partial L}{\partial y}\right]$;
- 更新动量：momentum $= \beta \cdot$ momentum $+ (1 - \beta) \cdot \nabla L$;
- 更新平方梯度：
  squared_grads $= \beta \cdot$ squared_grads $+ (1 - \beta) \cdot (\nabla L)^2$;
- 计算调整后的梯度：adjusted_grads $= \frac{\text{momentum}}{\sqrt{\text{squared\_grads}} + \epsilon}$;
- 更新参数：params $=$ params $- \alpha \cdot$ adjusted_grads;

输出：返回最终的参数 params;

Algorithm 2: 逻辑回归算法（通过梯度下降优化）

Input: 学习率 α，迭代次数 T，输入数据 X，标签 y

初始化：weights $= \mathbf{0}$，bias $= 0$;

定义 Sigmoid 函数：

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

训练过程：**for** 每个训练步骤 $t = 1, 2, \ldots, T$ **do**

- 计算线性模型输出：linear_model $= X \cdot$ weights $+$ bias;
- 计算预测值：predictions $= \sigma$(linear_model);
- 计算梯度：$dw = \frac{1}{n} X^T (\text{predictions} - y)$,
  $db = \frac{1}{n} \sum (\text{predictions} - y)$;
- 更新权重：weights $=$ weights $- \alpha \cdot dw$;
- 更新偏置：bias $=$ bias $- \alpha \cdot db$;

输出：返回最终的权重 weights 和偏置 bias;

Algorithm 3: 逻辑回归预测

Input: 输入数据 X，权重 weights，偏置 bias

计算线性模型输出：linear_model $= X \cdot \text{weights} + \text{bias}$;

计算预测概率：predicted_probabilities $= \sigma(\text{linear\_model})$;

根据预测概率判断类别：predicted_class $= \{1 \text{ if } p > 0.5 \text{ else } 0 \mid p \in$ predicted_probabilities$\}$;

输出：返回预测类别 predicted_class;

Algorithm 4: K-Nearest Neighbors（KNN）算法

Input: 输入数据 X_{train}，标签 y_{train}，测试数据 X_{test}，最近邻数目 k

初始化：将训练数据 X_{train} 和标签 y_{train} 存储为模型的属性；**for** 每个测试样本 x 在 X_{test} 中 **do**

  计算每个训练样本与测试样本之间的距离：distances $= [\text{dis}(x, x_{\text{train}})$ for each x_{train} in $X_{\text{train}}]$;

  找到最近的 k 个邻居：k_indices $= \text{argsort}(\text{distances})[:k]$;

  获取这些邻居的标签：k_nearest_labels $= [y_{\text{train}}[i]$ for i in $k\_indices]$;

  使用投票机制确定预测标签：most_common $=$ Counter$(k_nearest_labels).most_common(1)$;

  预测标签：predicted_label $=$ most_common[0][0];

输出：返回每个测试样本的预测标签;

Algorithm 5: 神经网络（前向传播与反向传播）

Input: 输入数据 X，标签 y，学习率 η

1 **初始化**：权重：weights_input_hidden，weights_hidden_output，
偏置：bias_hidden，bias_output；

2 **定义 Sigmoid 激活函数**：

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

3 **前向传播过程**：

$$\text{hidden\_layer\_input} = X \cdot \text{weights\_input\_hidden} + \text{bias\_hidden}$$

$$\text{hidden\_layer\_output} = \sigma(\text{hidden\_layer\_input})$$

$$\text{output\_layer\_input} = \text{hidden\_layer\_output} \cdot \text{weights\_hidden\_output} + \text{bias\_output}$$

$$\text{output\_layer\_output} = \sigma(\text{output\_layer\_input})$$

4 **反向传播过程**：计算输出层误差：

$$\text{d\_output} = (\text{output\_layer\_output} - y) \cdot \sigma'(\text{output\_layer\_output})$$

计算隐藏层误差：

$$\text{d\_hidden\_layer} = \text{d\_output} \cdot \text{weights\_hidden\_output}^T \cdot \sigma'(\text{hidden\_layer\_output})$$

5 更新权重和偏置：

$$\text{weights\_hidden\_output} -= \text{hidden\_layer\_output}^T \cdot \text{d\_output} \cdot \eta$$

$$\text{bias\_output} -= \sum \text{d\_output} \cdot \eta$$

$$\text{weights\_input\_hidden} -= X^T \cdot \text{d\_hidden\_layer} \cdot \eta$$

$$\text{bias\_hidden} -= \sum \text{d\_hidden\_layer} \cdot \eta$$

6 **输出**：返回更新后的权重和偏置；

Algorithm 6: K-means clustering 算法

Input: 数据集 $data$，簇的数量 $num\_clusters$，最大迭代次数 max_iter，收敛容忍度 tol

初始化：随机选择 $num\_clusters$ 个数据点作为初始质心 centroids;

for 每次迭代 $i = 1, 2, \ldots, max\_iter$ **do**

  计算每个数据点到所有质心的距离：distances = norm($data[:, None]$ − centroids, $axis = 2$);

  将每个数据点分配给最近的质心：labels = argmin(distances, $axis = 1$);

  更新质心：new_centroids = mean($data$[labels == i], $axis = 0$) for each cluster i;

  如果质心的变化小于容忍度，则停止迭代：if norm(new_centroids - centroids) ≤ tol then break;

  更新质心：centroids = new_centroids;

输出：返回最终的簇标签 labels 和质心 centroids;

Algorithm 7: DBSCAN 算法

Input: 数据集 $data$，距离阈值 ϵ，最小邻居数 min_pts

初始化：标签 labels = −1，访问标记 visited = 0，簇标记 cluster_id = 0;

邻居查找：定义函数 neighbors($point\_idx$) 计算距离小于 ϵ 的点;

扩展簇：定义函数 grow_cluster($idx, neighbors\_list$) 来扩展簇，并对未访问的点进行标记;

for 每个数据点 idx **do**

  如果已经访问过，则跳过；标记为已访问；查找邻居：neighbor_pts = neighbors(idx);

  如果邻居数大于等于 min_pts，则扩展簇：grow_cluster(idx, neighbor_pts);

  增加簇标记 $cluster\_id$;

输出：返回每个点的簇标签 labels;

Algorithm 8: 支持向量机分类 (SVC)

Input: 训练数据 $X \in \mathbb{R}^{n \times p}$，标签 $y \in \mathbb{R}^n$，惩罚系数 $C \in \mathbb{R}$，核函数 K, 容忍度 ϵ

1 **初始化**：选择合适的核函数 K，设置惩罚参数 C，初始化拉格朗日乘子 $\alpha = 0$；**for** 每个训练样本 $i = 1, 2, \ldots, n$ **do**

2 计算核函数：$K(x_i, x_j)$；计算拉格朗日乘子 α_i；通过最大化间隔，求解支持向量机优化问题：

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \quad \text{s.t.} \quad y_i(\mathbf{w}^T x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$$

其中 ξ_i 是松弛变量，用于允许误差；

3 **输出**：返回支持向量 α，权重向量 $\mathbf{w}$，偏置项 b；

Algorithm 9: 决策树分类 (Decision Tree Classifier)

Input: 训练数据 $X \in \mathbb{R}^{n \times p}$，标签 $y \in \mathbb{R}^n$，最大深度 $D \in \mathbb{N}$，最小样本分割数 $min\_samples\_split \in \mathbb{N}$

1 **初始化**：根节点为空；**while** 未满足停止条件 **do**

2 选择最佳特征 f，并计算特征 f 的划分点 θ；计算信息增益或基尼指数：

$$\text{Information Gain} = \text{Entropy}(S) - \sum_{v \in \text{Values}(f)} \frac{|S_v|}{|S|} \cdot \text{Entropy}(S_v)$$

或者使用基尼指数：

$$Gini = 1 - \sum_{i=1}^{k} p_i^2$$

将数据集按特征 f 和划分点 θ 划分为左右子集；递归构建左子树和右子树；

3 **输出**：返回构建的决策树；

Algorithm 10: 随机森林分类 (Random Forest Classifier)

Input: 训练数据 $X \in \mathbb{R}^{n \times p}$，标签 $y \in \mathbb{R}^n$，决策树数量 $n\_trees \in \mathbb{N}$，最大深度 $D \in \mathbb{N}$

for 每棵树 $t = 1, 2, \ldots, n\_trees$ **do**

  随机抽取数据集 $X_t \subset X$ 和 $y_t \subset y$，训练决策树 t；树的构建过程同决策树分类；

输出：通过所有树的投票机制，返回最终预测标签：

$$\hat{y} = \arg\max_{y} \sum_{t=1}^{n\_trees} I(y_t = y)$$

其中 I 是指示函数，表示类别 y 出现的次数；

Algorithm 11: 极限梯度提升*(XGBClassifier)

Input: 训练数据 $X \in \mathbb{R}^{n \times p}$，标签 $y \in \mathbb{R}^n$，学习率 $\eta \in \mathbb{R}$，树的数量 $n\_estimators \in \mathbb{N}$

初始化：初始模型为零函数 $f_0(x) = 0$；**for** 每棵树 $t = 1, 2, \ldots, n\_estimators$ **do**

  计算负梯度：

$$g_t = -\frac{\partial L}{\partial f_t(x)}$$

  用 g_t 训练一棵回归树，获得树的预测 $h_t(x)$；更新模型：

$$f_{t+1}(x) = f_t(x) + \eta \cdot h_t(x)$$

  其中 L 是损失函数，通常为对数损失或平方误差；

输出：返回最终的模型 $f_T(x)$；

Algorithm 12: K 近邻分类 (KNeighbors Classifier)

Input: 训练数据 $X_{\text{train}} \in \mathbb{R}^{n_{\text{train}} \times p}$，标签 $y_{\text{train}} \in \mathbb{R}^{n_{\text{train}}}$，测试数据 $X_{\text{test}} \in \mathbb{R}^{n_{\text{test}} \times p}$，最近邻数量 $k \in \mathbb{N}$

for 每个测试样本 $x \in X_{test}$ **do**

计算每个训练样本与测试样本之间的距离：

$$\text{distances} = \|X_{\text{train}} - x\|$$

找到最近的 k 个邻居：

$$\text{k\_indices} = \text{argsort}(\text{distances})[:k]$$

预测标签：$\hat{y} = \text{majority\_vote}(y_{\text{train}}[\text{k\_indices}])$；

输出：返回每个测试样本的预测标签；

Algorithm 13: 多层感知器 (MLPClassifier)

Input: 输入数据 $X \in \mathbb{R}^{n \times p}$，标签 $y \in \mathbb{R}^{n}$，学习率 $\eta \in \mathbb{R}$，隐藏层数量和神经元数量 $h_1, h_2, \ldots, h_L$

初始化：随机初始化权重 $W_{\text{input}}, W_{\text{hidden}}, \ldots$ 和偏置 $b_{\text{input}}, b_{\text{hidden}}, \ldots$；**前向传播**：

$$\text{hidden\_layer} = \sigma(X \cdot W_{\text{input}} + b_{\text{input}}), \quad \text{output\_layer} = \sigma(\text{hidden\_layer} \cdot W_{\text{hidden}} + b_{\text{hidden}})$$

其中 σ 是激活函数，例如 ReLU 或 Sigmoid；**反向传播**：

$$\text{error\_output} = \text{output\_layer} - y, \quad \text{error\_hidden} = \text{error\_output} \cdot W_{\text{hidden}}^T \cdot \sigma'(\text{hidden\_layer})$$

更新权重和偏置：

$$W_{\text{hidden}} -= \eta \cdot \text{error\_output} \cdot \text{hidden\_layer}^T, \quad W_{\text{input}} -= \eta \cdot \text{error\_hidden} \cdot X^T$$

输出：返回训练好的模型；

Algorithm 14: 朴素贝叶斯分类 (Gaussian Naive Bayes)

Input: 训练数据 $X \in \mathbb{R}^{n \times p}$，标签 $y \in \mathbb{R}^n$

初始化：根据训练数据 X 和标签 y 计算每个类别的先验概率：

$$P(y = c) = \frac{\sum_{i=1}^{n} I(y_i = c)}{n}$$

和每个特征的条件概率：

$$P(x_j | y = c) = \frac{\sum_{i=1}^{n} I(y_i = c) \cdot x_{ij}}{\sum_{i=1}^{n} I(y_i = c)}$$

for 每个测试样本 x **do**

计算后验概率：

$$P(y|X) = P(y) \prod_{j=1}^{p} P(x_j|y)$$

选择最大后验概率的类别作为预测标签：

$$\hat{y} = \arg\max_{c} P(y = c|X)$$

输出：返回每个测试样本的预测标签；

Algorithm 15: AdaBoost 分类器 (AdaBoostClassifier)

Input: 训练数据 $X \in \mathbb{R}^{n \times p}$，标签 $y \in \mathbb{R}^n$，弱分类器数量 $n\_estimators \in \mathbb{N}$

初始化：样本权重 $w_i = \frac{1}{n}$；**for** 每个弱分类器 $t = 1, 2, \ldots, n\_estimators$ **do**

训练一个弱分类器 h_t；计算加权错误率：

$$\text{error}_t = \frac{\sum_{i=1}^{n} w_i I(h_t(x_i) \neq y_i)}{\sum_{i=1}^{n} w_i}$$

更新分类器权重：

$$\alpha_t = \frac{1}{2} \ln \frac{1 - \text{error}_t}{\text{error}_t}$$

更新样本权重：

$$w_i = w_i \cdot e^{-\alpha_t y_i h_t(x_i)}$$

输出：返回最终加权分类器模型；

Algorithm 16: 岭回归 (Ridge Regression)

Input: 训练数据 $X \in \mathbb{R}^{n \times p}$，标签 $y \in \mathbb{R}^n$，正则化系数 $\lambda \in \mathbb{R}$

初始化：计算正规方程：

$$\hat{\beta} = (X^T X + \lambda I)^{-1} X^T y$$

输出：回归系数 $\hat{\beta} \in \mathbb{R}^p$

Algorithm 17: Lasso 回归 (Lasso Regression)

Input: 训练数据 $X \in \mathbb{R}^{n \times p}$，标签 $y \in \mathbb{R}^n$，正则化系数 $\lambda \in \mathbb{R}$

初始化：回归系数 $\beta = 0$；**for** 每个特征 $j = 1, 2, \ldots, p$ **do**

计算坐标下降步长：$r_j = \frac{1}{n} \sum_{i=1}^{n} x_{ij}(y_i - \hat{y}_i + \beta_j x_{ij})$；更新回归系数：$\beta_j = \text{soft\_threshold}(r_j, \lambda)$

输出：回归系数 $\beta \in \mathbb{R}^p$；

Algorithm 18: 弹性网回归*(ElasticNet)

Input: 训练数据 $X \in \mathbb{R}^{n \times p}$，标签 $y \in \mathbb{R}^n$，正则化系数 $\lambda_1, \lambda_2 \in \mathbb{R}$

初始化：回归系数 $\beta = 0$；**for** 每个特征 $j = 1, 2, \ldots, p$ **do**

  计算步长：$r_j = \frac{1}{n} \sum_{i=1}^{n} x_{ij}(y_i - \hat{y}_i + \beta_j x_{ij})$；更新回归系数：$\beta_j = \text{soft\_threshold}(r_j, \lambda_1 + \lambda_2)$；

输出：回归系数 $\beta \in \mathbb{R}^p$；

Algorithm 19: 随机森林回归 (Random Forest Regressor)

Input: 训练数据 $X \in \mathbb{R}^{n \times p}$，标签 $y \in \mathbb{R}^n$，树的数量 $n\_trees \in \mathbb{N}$，最大深度 $D \in \mathbb{N}$

初始化：随机森林模型 $\mathcal{F} = \{T_1, T_2, \ldots, T_{n\_trees}\}$；**for** 每棵树 $t = 1, 2, \ldots, n\_trees$ **do**

  随机选择数据子集 $X_t \subset X$，标签子集 $y_t \subset y$；构建回归树 T_t，最大深度为 D；

输出：预测值 $\hat{y} = \frac{1}{n\_trees} \sum_{t=1}^{n\_trees} T_t(X)$；

Algorithm 20: 梯度提升回归 (Gradient Boosting Regressor)

Input: 训练数据 $X \in \mathbb{R}^{n \times p}$，标签 $y \in \mathbb{R}^n$，树的数量 $n\_estimators \in \mathbb{N}$，学习率 $\eta \in \mathbb{R}$

初始化：初始模型 $\hat{y}_0 = \frac{1}{n} \sum_{i=1}^{n} y_i$；**for** 每棵树 $t = 1, 2, \ldots, n\_estimators$ **do**

  计算残差：$\text{residuals}_t = y - \hat{y}_{t-1}$；训练回归树 T_t，拟合残差；更新模型：$\hat{y}_t = \hat{y}_{t-1} + \eta \cdot T_t(X)$；

输出：最终预测值 $\hat{y}_{final} = \hat{y}_{n\_estimators}$；

Algorithm 21: 自适应提升回归 (AdaBoost Regressor)

Input: 训练数据 $X \in \mathbb{R}^{n \times p}$，标签 $y \in \mathbb{R}^n$，弱回归器数量 $n\_estimators \in \mathbb{N}$

初始化：样本权重 $w_i = \frac{1}{n}$，初始预测模型为常数模型 $\hat{y}_0 = \frac{1}{n}\sum_{i=1}^{n} y_i$；**for** 每个弱回归器 $t = 1, 2, \ldots, n\_estimators$ **do**

  训练回归器 h_t，并计算加权误差率：$\text{error}_t = \frac{\sum_{i=1}^{n} w_i \cdot (y_i - h_t(x_i))^2}{\sum_{i=1}^{n} w_i}$；

  计算回归器权重：$\alpha_t = \frac{1}{2} \ln \frac{1-\text{error}_t}{\text{error}_t}$；更新样本权重：$w_i = w_i \cdot e^{-\alpha_t (y_i - h_t(x_i))^2}$；

输出：最终加权回归器模型 $\hat{y} = \sum_{t=1}^{n\_estimators} \alpha_t h_t(x)$；

Algorithm 22: 支持向量回归 (SVR)

Input: 训练数据 $X \in \mathbb{R}^{n \times p}$，标签 $y \in \mathbb{R}^n$，惩罚系数 $C \in \mathbb{R}$，核函数 K，容忍度 $\epsilon \in \mathbb{R}$

初始化：选择合适的核函数 K，设置惩罚参数 C，容忍度 ϵ；**for** 每个训练样本 $i = 1, 2, \ldots, n$ **do**

  求解拉格朗日乘子，最大化间隔，最小化以下目标函数：

$$\mathcal{L}(\beta) = \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{n} \max(0, |y_i - \hat{y}_i| - \epsilon)$$

输出：支持向量回归模型 $\hat{y} = f(X)$；

Algorithm 23: 决策树回归 (Decision Tree Regressor)

Input: 训练数据 $X \in \mathbb{R}^{n \times p}$，标签 $y \in \mathbb{R}^n$，最大深度 $D \in \mathbb{N}$，最小样本分割数 $min\_samples\_split \in \mathbb{N}$

初始化：根节点为空；**while** 未满足停止条件 **do**

  计算每个特征的最佳切分点：

$$\text{BestSplit} = \arg\min_{\text{feature,threshold}} \text{MSE}_{\text{split}}$$

  将数据集按最佳切分点切分为左右子集；对每个子集递归调用决策树构建算法；

输出：决策树回归模型；