逻辑回归与梯度下降原理详解

机器学习教程

目录

1 逻辑回归识别手写数字教程 **1**

1.1 代码概览 1

1.1.1 导入库 1

1.2 MNIST 数据集 1

1.3 数据预处理 2

1.3.1 展平图像 2

1.3.2 归一化到 [0, 1] 2

1.4 逻辑回归模型 2

1.4.1 逻辑回归原理 2

1.4.2 超参数解释 3

1.5 模型评估 3

1.6 逻辑回归核心数学公式 4

1.6.1 二分类情况 4

1.6.2 多分类情况（Softmax 函数） 4

2 逻辑回归数学原理深度解析 **4**

2.1 逻辑回归的目标 4

2.2 线性组合：$w^T x + b$ 4

2.3 偏置项 b 的作用 5

2.4 从线性模型到概率模型 5

2.5 参数 w 和 b 的学习过程 5

2.5.1 训练数据 5

2.5.2 模型预测 5

2.5.3 损失函数 6

1 逻辑回归识别手写数字教程 2

2.5.4 参数优化 6

3 梯度下降原理详解 6

3.1 直觉理解 6
3.2 梯度下降思路 6
3.3 数学解释 6
3.4 学习率 η 7
3.5 梯度下降示例 7
3.6 逻辑回归中的梯度公式 7
3.7 核心概念总结 8

1 逻辑回归识别手写数字教程

1.1 代码概览

这段代码的目标是：训练一个逻辑回归模型，让它识别 0 9 的手写数字。

```
import numpy as np
from sklearn.linear_model import LogisticRegression
from tensorflow.keras.datasets import mnist
```

1.1.1 导入库

- `numpy`：科学计算库，用于操作矩阵、数组
- `sklearn.linear_model`：Scikit-Learn 的机器学习模块
- `tensorflow.keras.datasets`：Keras 自带的数据集

1.2 MNIST 数据集

```
(X_train, y_train), (X_test, y_test) = mnist.load_data()
```

自动下载并加载经典数据集：

- X_{train}：训练图片（60000 张 28×28 像素灰度图）

- y_{train}：对应的标签（0 9）
- X_{test}：测试图片（10000 张）
- y_{test}：测试标签

图片	标签
手写的"3"	3
手写的"7"	7

表 1: MNIST数据集示例

1.3 数据预处理

1.3.1 展平图像

```
X_train = X_train.reshape(-1, 784)
X_test = X_test.reshape(-1, 784)
```

原始图片是 28×28 像素，展平为 784 维向量。

1.3.2 归一化到 [0, 1]

```
X_train = X_train.astype('float32') / 255
X_test = X_test.astype('float32') / 255
```

像素值从 0 255 归一化到 0 1 区间。

1.4 逻辑回归模型

```
clf = LogisticRegression(penalty="l1", solver="saga", tol=0.
clf.fit(X_train, y_train)
```

1.4.1 逻辑回归原理

逻辑回归是分类算法，预测样本属于某类的概率。输入图片向量 x，模型计算：

$$P(y=k \mid x)=\frac{e^{w_k^T x+b_k}}{\sum_{j=0}^{9} e^{w_j^T x+b_j}}$$

这就是 **Softmax 回归**（逻辑回归的多分类版本）。

其中：

- w_k：第 k 个数字的权重向量（784维）
- b_k：偏置
- 输出是每个数字的概率，选概率最大的类别作为预测

1.4.2 超参数解释

- `penalty="l1"`：使用 L1 正则化
- `solver="saga"`：优化算法，适合大数据和 L1
- `tol=0.1`：容忍误差
- `max_iter=100`：最大迭代次数

L1 正则化（稀疏化）：训练时最小化损失函数：

$$L=-\sum_i \log P(y_i|x_i)+\lambda\sum_j |w_j|$$

第二项是正则项，用来防止过拟合。

1.5 模型评估

```
score = clf.score(X_test, y_test)
print("Test score with L1 penalty: %.4f" % score)
```

计算测试集上的准确率：

$$\text{accuracy}=\frac{\text{预测正确的样本数}}{\text{总样本数}}$$

输出约为：`Test score with L1 penalty: 0.9187`

1.6 逻辑回归核心数学公式

1.6.1 二分类情况

$$\hat{y} = \sigma(w^T x + b)$$

其中 Sigmoid 函数：

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

将任意数转换为 0 1 之间的概率。

1.6.2 多分类情况（Softmax 函数）

$$P(y = k|x) = \frac{e^{w_k^T x + b_k}}{\sum_j e^{w_j^T x + b_j}}$$

2 逻辑回归数学原理深度解析

2.1 逻辑回归的目标

逻辑回归要解决的问题是：给定一组特征 x，预测它属于某个类别（比如”是否是数字3”）的概率。

2.2 线性组合：$w^T x + b$

假设每个像素都是一个特征（总共784个），每个特征都有一个权重参数 w_i 表示重要程度。加权求和：

$$z = w_1 x_1 + w_2 x_2 + \cdots + w_{784} x_{784} + b$$

或用矩阵形式：

$$z = w^T x + b$$

- x：输入（784维向量）
- w：模型参数（784个数字）
- b：偏置（bias），相当于”起点调整”

- $w^T x$：内积（dot product）

这一步的本质是：把一个高维输入 x，通过加权求和压缩成一个实数 z。

2.3 偏置项 b 的作用

如果没有 b，函数就一定过原点 $(0,0)$，灵活性太低。加上 b 就相当于给整个函数"上下平移"的自由度。

2.4 从线性模型到概率模型

$z = w^T x + b$ 的结果可能是任意实数，但我们希望输出是概率（0 1之间）。引入 **Sigmoid 函数**：

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

这样：

- 当 z 很大时，$\hat{y} \to 1$
- 当 z 很小时，$\hat{y} \to 0$

所以：

$$\hat{y} = P(y = 1|x) = \sigma(w^T x + b)$$

2.5 参数 w 和 b 的学习过程

参数不是人工设定的，而是通过训练数据"学出来"的。

2.5.1 训练数据

$$(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$$

2.5.2 模型预测

对每个样本：

$$\hat{y_i} = \sigma(w^T x_i + b)$$

2.5.3 损失函数

逻辑回归使用交叉熵损失：

$$L(w,b) = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)]$$

2.5.4 参数优化

通过梯度下降最小化损失函数：

$$w := w - \eta\frac{\partial L}{\partial w} \tag{1}$$

$$b := b - \eta\frac{\partial L}{\partial b} \tag{2}$$

3 梯度下降原理详解

3.1 直觉理解

损失函数 $L(w,b)$ 告诉我们模型预测的误差。目标是最小化这个误差，就像在山地地形图上找最低点。

3.2 梯度下降思路

站在山上某一点（当前 w,b），要走到谷底（最小点），需要沿着最陡的下坡方向走，即梯度的反方向。

3.3 数学解释

梯度是损失函数对参数的偏导数组：

$$\nabla L = \left[\frac{\partial L}{\partial w}, \frac{\partial L}{\partial b}\right]$$

梯度下降更新公式：

$$w_{\text{new}} = w_{\text{old}} - \eta\frac{\partial L}{\partial w} \tag{3}$$

$$b_{\text{new}} = b_{\text{old}} - \eta\frac{\partial L}{\partial b} \tag{4}$$

3.4 学习率 η

学习率控制每次"下山"的步长：

- 太小：收敛慢，训练时间长
- 太大：可能震荡或不收敛

3.5 梯度下降示例

以简单二次函数为例：

$$L(w) = (w-3)^2$$

求导：

$$\frac{dL}{dw} = 2(w-3)$$

梯度下降：

$$w := w - \eta \times 2(w-3)$$

迭代	当前 w	梯度 $2(w-3)$	新的 w ($\eta = 0.1$)
0	0.0	-6.0	$0 - 0.1 \times (-6) = 0.6$
1	0.6	-4.8	$0.6 - 0.1 \times (-4.8) = 1.08$
2	1.08	-3.84	$1.08 - 0.1 \times (-3.84) = 1.464$
3	1.464	-3.072	$1.464 - 0.1 \times (-3.072) = 1.7712$
...	...	...	最终逼近 $w = 3$

表 2: 梯度下降迭代过程

3.6 逻辑回归中的梯度公式

逻辑回归损失函数：

$$L(w,b) = -\frac{1}{N}\sum_i [y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)]$$

其中：

$$\hat{y}_i = \sigma(w^T x_i + b)$$

求导结果：

$$\frac{\partial L}{\partial w} = \frac{1}{N}\sum_i (\hat{y}_i - y_i)x_i \tag{5}$$

$$\frac{\partial L}{\partial b} = \frac{1}{N}\sum_i (\hat{y}_i - y_i) \tag{6}$$

3.7 核心概念总结

概念	含义
$L(w,b)$	损失函数，越小表示模型越准
$\frac{\partial L}{\partial w}$	告诉我们 w 应该往哪个方向改才能让损失变小
η	学习率，控制每次改多少
更新公式	$w := w - \eta\frac{\partial L}{\partial w}$
结果	经过多次迭代，最终找到让损失最小的 w, b

表 3: 梯度下降核心概念