Introduction to Artificial Intelligence

timetraveler314
University of Genshin
timetraveler314@outlook.com

Contents

1. Lec 9 机器学习和线性回归 (2024/3/21)

1.1. 参数化模型

最简单的参数化模型：线性模型 (Linear Model).

$$f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b.$$

其中 $\boldsymbol{x} \in \mathbb{R}^d$ 是输入, $\boldsymbol{w} \in \mathbb{R}^d$ 是权重向量, $b \in \mathbb{R}$ 是偏置项.

1.2. 线性回归训练

- 训练目标：最小化损失函数.
- 对于线性回归，损失函数通常是均方误差 (Mean Squared Error, MSE), 又名平方损失 (Squared Loss).

$$L(f(\boldsymbol{x}_i), y_i) = (f(x_i) - y_i)^2.$$

惩罚预测值偏离真实值 (ground truth) 太大的情况.

- 通过在训练集上最小化平均损失函数来优化参数.

$$\underset{w,b}{\operatorname{argmin}} \frac{1}{n} \sum_{i} L(f(\boldsymbol{x}_i), y_i)$$

训练问题转化为求解以上优化问题.

1.3. 梯度下降

分析上述问题，我们将待优化的**目标** (objective) 定义为 $\boldsymbol{w}, b$ 的函数：

$$J(\boldsymbol{w}, b) = \frac{1}{n} \sum_{i} L(f(x_i) - y_i).$$

- 随机选定$\boldsymbol{w}, b$的初始值，逐步调整$\boldsymbol{w}, b$的值以降低$J(\boldsymbol{w}, b)$.
- 思路：每次沿使$J(\boldsymbol{w}, b)$减小最快的方向走一小步.

$$w \leftarrow w - \alpha \cdot \frac{\partial J}{\partial \boldsymbol{w}}, b \leftarrow b - \alpha \cdot \frac{\partial J}{\partial b}.$$

其中事先指定的超参数 α 是**学习率** (learning rate).

- 不断迭代，直至 J 收敛(例如相邻两次迭代J的变化小于某个阈值).

这就是**梯度下降** (Gradient Descent) 算法.

1.3.1. 梯度的细节

$$\boldsymbol{\nabla} J = \begin{pmatrix} \frac{\partial J}{\partial \boldsymbol{w}} \\ \frac{\partial J}{\partial b} \end{pmatrix} \in \mathbb{R}^{d+1}$$

上式定义了J关于$\boldsymbol{w}, b$的梯度. 考察梯度的几何意义：

考虑增量 $\Delta x, \Delta y$.

$$\begin{aligned} \Delta f = \frac{\partial f}{\partial x} \Delta x + \frac{\partial f}{\partial y} \Delta y &= \boldsymbol{\nabla} f \cdot (\Delta x, \Delta y) \\ &= \|\boldsymbol{\nabla} f\| \left\| \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} \right\| \cos(\theta). \end{aligned}$$

Δf 最大时，$\theta = 0$，此时 $\Delta x, \Delta y$ 与 $\boldsymbol{\nabla} f$ 同向.

这说明梯度的几何意义：梯度方向是函数增长最快的方向.

1.4. 线性回归的梯度

- 计算 $J(\boldsymbol{w}, b)$ 关于 $\boldsymbol{w}, b$ 的梯度, 首先将 J 写成矩阵形式：

$$J = \frac{1}{n} \underbrace{(\boldsymbol{w}^T \boldsymbol{X} + \boldsymbol{b} - \boldsymbol{y})}_{\text{行向量}} (\boldsymbol{w}^T \boldsymbol{X} + \boldsymbol{b} - \boldsymbol{y})^T.$$

其中 $\boldsymbol{X} = \begin{pmatrix} \boldsymbol{x}_1 & \boldsymbol{x}_2 & \cdots & \boldsymbol{x}_n \end{pmatrix}$ 是输入数据矩阵, $\boldsymbol{y} = \begin{pmatrix} y_1 & y_2 & \cdots & y_n \end{pmatrix}$ 是标签向量.

Theorem 1.4.1

$$\frac{\partial J}{\partial \boldsymbol{w}} = \frac{2}{n}\sum_i(\boldsymbol{w}^T\boldsymbol{x}_i + b - y_i)\boldsymbol{x}_i = \frac{2}{n}\boldsymbol{X}(\boldsymbol{w}^T\boldsymbol{X} + \boldsymbol{b} - \boldsymbol{y})^T,$$
$$\frac{\partial J}{\partial b} = \frac{2}{n}\sum_i(\boldsymbol{w}^T\boldsymbol{x}_i + b - y_i).$$

Note 1.4.1

根据“维度相容原则”可简记如下：

$$\frac{\partial J}{\partial \boldsymbol{w}} = \frac{2}{n}\frac{\partial \boldsymbol{w}^T\boldsymbol{X} + \boldsymbol{b} - \boldsymbol{y}}{\partial \boldsymbol{w}}(\boldsymbol{w}^T\boldsymbol{X} + \boldsymbol{b} - \boldsymbol{y})^T = \frac{2}{n}\boldsymbol{X}(\boldsymbol{w}^T\boldsymbol{X} + \boldsymbol{b} - \boldsymbol{y})^T,$$

下面证明是通过严格计算微分得到的.

Proof: 计算 J 的微分：

$$\mathrm{d}J = \frac{1}{n}\left[\mathrm{d}(\boldsymbol{w}^T\boldsymbol{X} + \boldsymbol{b} - \boldsymbol{y})(\boldsymbol{w}^T\boldsymbol{X} + \boldsymbol{b} - \boldsymbol{y})^T + (\boldsymbol{w}^T\boldsymbol{X} + \boldsymbol{b} - \boldsymbol{y})\mathrm{d}(\boldsymbol{w}^T\boldsymbol{X} + \boldsymbol{b} - \boldsymbol{y})^T\right].$$

这里注意应用内积原则：

Note 1.4.2 矩阵求导常用的内积原则

对列向量 $\boldsymbol{u}, \boldsymbol{v}$,

$$\boldsymbol{u}^T\boldsymbol{v} = \boldsymbol{v}^T\boldsymbol{u}.$$

对行向量 $\boldsymbol{u}, \boldsymbol{v}$,

$$\boldsymbol{u}\boldsymbol{v}^T = \boldsymbol{v}\boldsymbol{u}^T.$$

因为以上的式子都表示内积.

$$\begin{aligned}(\boldsymbol{w}^T\boldsymbol{X} + \boldsymbol{b} - \boldsymbol{y})\mathrm{d}(\boldsymbol{w}^T\boldsymbol{X} + \boldsymbol{b} - \boldsymbol{y})^T &= (\boldsymbol{w}^T\boldsymbol{X} + \boldsymbol{b} - \boldsymbol{y})[\mathrm{d}(\boldsymbol{w}^T\boldsymbol{X} + \boldsymbol{b} - \boldsymbol{y})]^T \\ &= \mathrm{d}(\boldsymbol{w}^T\boldsymbol{X} + \boldsymbol{b} - \boldsymbol{y})(\boldsymbol{w}^T\boldsymbol{X} + \boldsymbol{b} - \boldsymbol{y})^T(\text{与第一项相同})\end{aligned}$$

$$\mathrm{d}J = \frac{2}{n}\mathrm{d}(\boldsymbol{w}^T\boldsymbol{X} + \boldsymbol{b} - \boldsymbol{y})\cdot(\boldsymbol{w}^T\boldsymbol{X} + \boldsymbol{b} - \boldsymbol{y})^T.$$

计算 $\mathrm{d}(\boldsymbol{w}^T\boldsymbol{X} + \boldsymbol{b} - \boldsymbol{y})$:

$$\mathrm{d}(\boldsymbol{w}^T\boldsymbol{X} + \boldsymbol{b} - \boldsymbol{y}) = \mathrm{d}(\boldsymbol{w}^T\boldsymbol{X}) = (\mathrm{d}\boldsymbol{w})^T\boldsymbol{X}.$$

用标量矩阵函数求导的核心：

$$\mathrm{d}f = \mathrm{tr}\left(\frac{\partial f}{\partial \boldsymbol{X}}^T\mathrm{d}\boldsymbol{X}\right).$$

对比

$$\mathrm{d}J = \frac{2}{n}(\mathrm{d}\boldsymbol{w})^T X(\boldsymbol{w}^T\boldsymbol{X}+\boldsymbol{b}-\boldsymbol{y})^T = \frac{2}{n}(\boldsymbol{w}^T\boldsymbol{X}+\boldsymbol{b}-\boldsymbol{y})\boldsymbol{X}^T\,\mathrm{d}\boldsymbol{w},$$

得到

$$\begin{aligned}\frac{\partial J}{\partial \boldsymbol{w}} &= \frac{2}{n}\boldsymbol{X}(\boldsymbol{w}^T\boldsymbol{X}+\boldsymbol{b}-\boldsymbol{y})^T \\ &= \frac{2}{n}\sum_i(\boldsymbol{w}^T\boldsymbol{x}_i+b-y_i)\boldsymbol{x}_i.\end{aligned}$$

对 b 求导是显然的. □

1.5. 凹凸性与优化

- 凹函数 (Convex Function) 的定义：对于任意 x, y 和 $0 \le \lambda \le 1$,

$$f(\lambda x+(1-\lambda)y) \le \lambda f(x)+(1-\lambda)f(y).$$

- 凹函数的性质：

Theorem 1.5.1 凹函数的性质

凹函数的局部最小值是全局最小值.

Proof: 反证.

- 假设 x^* 是局部最小点，x' 是全局最小点，且 $f(x') < f(x^*)$.
- 则取 $x = \lambda x^* + (1-\lambda)x'$，有 $f(x) \le \lambda f(x^*) + (1-\lambda)f(x') < f(x^*)$.
- 取 $\lambda \to 1$附近，知 x^* 不是局部最小点，矛盾.

□

因此我们可以考虑线性回归问题：

Theorem 1.5.2

线性回归的损失函数 $J(\boldsymbol{w}, b)$ 是凸函数.

Proof: 考虑 Hessian 矩阵 H:

$$\begin{aligned}H &= \begin{pmatrix}\frac{\partial^2 J}{\partial \boldsymbol{w}^2} & \frac{\partial^2 J}{\partial \boldsymbol{w}\partial b} \\ \frac{\partial^2 J}{\partial b\partial \boldsymbol{w}} & \frac{\partial^2 J}{\partial b^2}\end{pmatrix} = \begin{pmatrix}\frac{2}{n}\boldsymbol{X}\boldsymbol{X}^T & \frac{2}{n}\boldsymbol{X}\boldsymbol{1}^T \\ \frac{2}{n}\boldsymbol{X}^T\boldsymbol{1} & \frac{2}{n}n\end{pmatrix} \\ &= \frac{2}{n}\begin{pmatrix}\boldsymbol{X} \\ \boldsymbol{1}\end{pmatrix}\begin{pmatrix}\boldsymbol{X} \\ \boldsymbol{1}\end{pmatrix}^T.\end{aligned}$$

因此 H 是半正定矩阵，J 是凸函数. □

1.5.1. 学习率的调整

- 学习率过大可能导致算法不收敛，过小可能导致收敛速度慢.
- 要选择适中的学习率，可以考虑自适应学习率算法，逐步减小.

2. Lec 10. 逻辑回归、多分类和正则化 (2024/3/25)

2.1. 回顾：经验风险最小化框架 (ERM)

大部分监督学习都遵循基本框架：经验风险最小化 (Empirical Risk Minimization, ERM)，区别仅在于选择的具体损失函数.

- ERM 框架：在训练集上最小化损失函数.

2.2. 逻辑回归

逻辑回归处理二分类问题. 仍然使用线性模型，但采用交叉熵损失函数 (Cross Entropy Loss).

2.2.1. 二分类问题

- 标签只有两种. e.g. $y \in \{-1, 1\}$.
- 一般不直接让 $f(x)$ 拟合 y，而是使用 $\mathrm{sign}(f)$ 将实数输出转化为二分类的类别输出. 因而转化为了一个回归问题.
- 损失函数的选择：

朴素想法：0-1 损失函数：

$$L(f(x), y) = \begin{cases} 0, & f(x)y \geq 0 \\ 1, & \text{otherwise} \end{cases} \Leftrightarrow L(f(x), y) = \begin{cases} 0, & \text{if} \, \mathrm{sign}(f(x)) = y \\ 1, & \text{otherwise} \end{cases}.$$

这是最直接的目标，但是不连续，不易优化.

2.2.2. 最大似然框架 (Maximum Likelihood)

- 最大似然估计的原则：

(1) 对观测数据进行(条件)概率建模： 每个观测数据即为一个训练样本.

对于判别式模型，只建模 $p(y|x;\theta), \theta$ 是模型参数.

(2) 通过最大化观测数据在给定模型下的**似然**（例如，把训练样本预测正确的概率），来调整模型参数.

独立同分布假设下，MLE: $\theta^* = \operatorname*{argmax}_\theta \prod_i p(y = y_i | x = x_i; \theta)$. 简写为

$$\theta^* = \operatorname*{argmax}_\theta \prod_i p(y_i | x_i; \theta).$$

但是大量乘法会带来数值精度问题，因此通常转化为对数似然最大化：

$$\theta^* = \operatorname*{argmax}_\theta \sum_i \log p(y_i | x_i; \theta).$$

2.2.3. 逻辑回归的最大似然估计

- 建模：

已有线性模型 $f(x) = \boldsymbol{w}^T \boldsymbol{x} + b$. 只需将其转化为正类的概率.

采用 Sigmoid 函数：

Note 2.2.3.1 Sigmoid 函数

$$\sigma(z) = \frac{1}{1 + \exp(-z)}.$$

$$1 - \sigma(z) = \sigma(-z).$$

将 $\mathbb{R}$ 映射到 $[0, 1]$ 区间，可以看作是概率值.

则

$$p(y = 1 | x; \boldsymbol{w}, b) = \sigma(f(x)) = \sigma(y \cdot f(x)),$$

$$p(y = -1 | x; \boldsymbol{w}, b) = 1 - \sigma(f(x)) = \sigma(y \cdot f(x)).$$

二者形式恰统一.

- 优化对数似然：

这里最优化问题为

$$\begin{aligned} & \underset{\boldsymbol{w}, b}{\operatorname{argmax}} \sum_i \log \sigma(y \cdot f(x)) \\ = & -\underset{\boldsymbol{w}, b}{\operatorname{argmax}} \sum_i \log(1 + \exp(-y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b))). \end{aligned}$$

写成最小化形式，这就是逻辑回归的 **ERM** 形式.

$$\underset{\boldsymbol{w}, b}{\operatorname{argmin}} \sum_i \log(1 + \exp(-y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b))).$$

其中提取出损失函数：Logistic Loss / Log Loss.

2.2.4. 另一视角：替代损失函数

以上从最大似然估计的角度推导了逻辑回归的损失函数，但也可以从另一角度看待:

- 交叉熵损失函数是$0 - 1$损失的上界，且是凸函数.
- 合页损失 (Hinge Loss)：$L = \max(0, 1 - y_i f(x_i))$ 同样是$0 - 1$损失的上界.

2.3. 多分类问题: Softmax 回归

2.3.1. K 分类问题

- 标签有 K 个类别，$y \in \{1, 2, ..., K\}$.
- 共同训练 K 个模型 $f_1(x), f_2(x), ..., f_{K(x)}$，每个模型输出属于该类别的概率.
- 概率归一化：Softmax 函数

Definition 2.3.1.1 Softmax 函数

$$\operatorname{softmax}(z) = \frac{[\exp(z_1), \exp(z_2), ..., \exp(z_K)]}{\sum_i \exp(z_i)}.$$

3. Lec 12. 神经网络与反向传播

3.1.