Machine Learning Algorithm

BitNene

11.27, 2023

摘要

目录

1 引言

2 线性回归算法

2.1 简介

2.2 定义和数学方法

对于两个自变量的线性回归：

$$Y = X_1\theta_1 + X_2\theta_2$$

拟合的平面：

$$h_\theta(x_1, x_2) = \theta_0 + \theta_1 x_1 + \theta x_2$$

一般性整合：

$$h_\theta(\mathbf{x}) = \sum_{i=1}^{n} \theta_i x_i = \theta^T \mathbf{x}$$

误差（概率论基础）：

- 真实值和预测值之间的差异，一般记作 ε
- 对于每个样本，预测和误差： $y^{(i)} = \theta^T x^{(i)} + \varepsilon^{(i)}$
- 误差服从高斯分布：

$$p(\varepsilon^{(i)}) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(\varepsilon^{(i)})^2}{2\sigma^2}}$$

- 似然函数：

$$L(\theta) = \prod_{i=1}^{m} p(y^{(i)}|x^{(i)};\theta)$$

- 对数似然函数：

$$\log L(\theta)$$

对于对数似然函数展开化简：

$$\sum_{i=1}^{m} \log \frac{1}{\sqrt[1]{2\pi}} \exp(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}) = m \log \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{\sigma^2} \cdot \frac{1}{2} \sum_{i=1}^{m} (y^{(i)} - \theta^T x^{(i)})^2$$

目标函数

$$\begin{aligned} J(\theta) &= \frac{1}{2} \sum_{i=1}^{m} (y^{(i)} - \theta^T x^{(i)})^2 \\ &= \frac{1}{2} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2 = \frac{1}{2} (X\theta - y)^T (X\theta - y) \end{aligned}$$

求相应的参数

$$\theta_{val} = argminJ(\theta)$$

$$
\begin{aligned}
\nabla_\theta J(\theta) &= \nabla_\theta\Big(\frac{1}{2}(X\theta-y)^T(X\theta-y)\Big)\\
&= \nabla_\theta\Big(\frac{1}{2}\Big(\theta^T X^T - y^T\Big)(X\theta-y)\Big)\\
&= \frac{1}{2}\nabla_\theta\Big(\theta^T X^T X\theta - \theta^T X^T y - y^T X\theta + y^T y\Big)\\
&= \frac{1}{2}\Big(2X^T X\theta - X^T y - \big(y^T X\big)^T\Big)\\
&= X^T X\theta - y^T y
\end{aligned}
$$

令导数等于0,　　则有 $\theta = (X^T X)^{-1} X^T y$

2.3 优化算法–梯度下降法

引入：得到一个目标函数后，如何进行求解？

直接求解？（并不一定可解，线性回归是一个有公式的特例）

如何优化参数？

目标函数：

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\Big(h_\theta\big(x^{(i)}\big) - y^{(i)}\Big)^2$$

学习率(LR):对结果会产生巨大的影响，一般需要小一些

批处理数量（batch）：32,64,128都可以

- 批量梯度下降(GD)：

$$\frac{\partial J(\theta)}{\partial\theta_j} = -\frac{1}{m}\sum_{i=1}^{m}(y^i - h_\theta(x^i))x_j^i \qquad \theta_j^{'} = \theta_j + \frac{1}{m}\sum_{i=1}^{m}(y^i - h_\theta(x^i))x_j^i$$

  容易得到最优解，但是速度慢

- 随机梯度下降(SGD)：

$$\theta_j^{'} = \theta_j + (y^i - h_\theta(x^i))x_j^i$$

  迭代速度快，但是不一定每次都朝着收敛的方向

- 小批量梯度下降：

$$\theta_j := \theta_j - \alpha\frac{1}{10}\sum_{k=i}^{i+9}(h_\theta(x^{(k)}) - y^{(k)})x_j^{(k)}$$

每次更新选择一小部分数据来算，比较实用！

2.4 总结