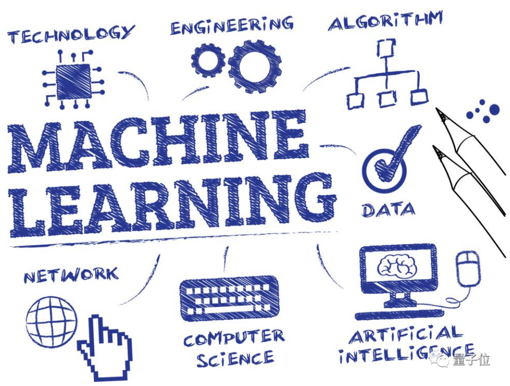
西安交通大学
XI'AN JIAOTONG UNIVERSITY

机器学习大作业



课程名称：　机器学习

姓名：　李峥昊

学院：　管理学院

专业：　大数据管理与应用

学号：　2201111618

2023 年 1 月 1 日

机器学习 **Lecture 1**

线性模型

一、 Introduction

Defination 1 (线性模型)**.**

$$h(\mathbf{x}) = w_1x_1 + w_2x_2 + \cdots + w_nx_n + b$$

Theorem 1. $h(\mathbf{x}) = w_1x_1 + w_2x_2 + \cdots + w_nx_n + b$ 等价于 $h(\mathbf{x}) = \mathbf{w}^\top\mathbf{x}$

证明.

$$\begin{aligned} h(\mathbf{x}) &= b + \sum_{i=1}^{n} w_i x_i \\ &= b \cdot 1 + \sum_{i=1}^{n} w_i x_i \\ &= \sum_{i=0}^{n} w_i x_i, \text{where} \quad \mathbf{x}_0 = 1 \end{aligned}$$

□

二、 感知机

感知机是解决 Binary Classfication 的一种方法，根据数据集是否完全线性可分分为 PLA 和 Pocket 两种算法。这两种算法共用一种模型，只是在停止条件上有所区别。

1. 定义

Defination 2 (Binary Classfication)**.**

$$h(x) = \text{sign}(\mathbf{w}^\top\mathbf{x})$$

$h(\mathbf{x}) = +1$ 时为正类，$h(\mathbf{x}) = -1$ 时为负类

Defination 3 (感知机)**.**

$$\min_{\mathbf{w}} \quad \sum_{i=0}^{m} [y_i \neq \text{sign}(\mathbf{w}^\top\mathbf{x}_i)] \tag{1}$$

等价于

$$\min_{\mathbf{w}} \quad \sum_{\mathbf{x}_i \in \Omega} y_i \mathbf{w}^\top \mathbf{x}_i \tag{2}$$

其中，Ω 表示分类错误的点

两种优化模型对应了两种不同的思路 opt.1优化分类错误的个数，opt.2优化分类错误点到超平面的距离

推导 1. 正确分类等价于 $h(\mathbf{x}) = y$，误分类等价于 $h(\mathbf{x}) \neq y$，因此所有误分类的点的个数等于

$$\sum_{i=1}^{m} [y_i \neq \text{sign}(\mathbf{w}^\top \mathbf{x}_i)]$$

显然上式越小，误分类点个数越少

推导 2. 点到超平面距离

$$d = \frac{|\mathbf{w}^\top \mathbf{x}|}{||\mathbf{w}||}$$

等价于

$$d_i = \begin{cases} y_i \dfrac{\mathbf{w}^\top \mathbf{x}_i}{||\mathbf{w}||}, & \text{分类正确} \\ -y_i \dfrac{\mathbf{w}^\top \mathbf{x}_i}{||\mathbf{w}||}, & \text{分类错误} \end{cases}$$

又因为：$c\mathbf{w}^\top \mathbf{x}$ 不改变超平面，所以令 $||\mathbf{w}|| = 1$，得误分类点到超平面的距离等于：

$$\sum_{\mathbf{x}_i \in \Omega} -y_i \mathbf{w}^\top \mathbf{x}_i$$

2. 迭代方法

由于 opt.1不可微，所以难以进行优化，不过可以对于 opt.2写出对应的梯度下降法和随机梯度下降法

$$\nabla E(\mathbf{w}) = \sum_{\mathbf{x}_i \in \Omega} -y_i \mathbf{x}_i$$

$$\nabla_i E(\mathbf{w}) = \begin{cases} -y_i \mathbf{x}_i, & y_i \mathbf{w}^\top \mathbf{x} \leq 0 \\ 0, & y_i \mathbf{w}^\top \mathbf{x} > 0 \end{cases}$$

3. 收敛定理

Theorem 2 (收敛定理)**.** 在线性可分的情况下，收敛步数 t 和数据半径 R、到最优超平面最近点距离 ρ 满足以下关系：

$$t \leq \frac{R^2}{\rho^2}$$

证明. 规定最优超平面为 $\mathbf{w}_f$,t 步迭代后的超平面为 $\mathbf{w}_t$

假设数据在半径 r 内

$$\max_{1 \leq i \leq m} ||\mathbf{x}_i||^2 \leq R^2$$

假设存在 $\mathbf{w}_f$ 使得

$$y_i \frac{\mathbf{w}_f^\top \mathbf{x}_i}{||\mathbf{w}||} \geq \rho$$

即

$$\min_{1\leq i\leq m} y_i \frac{\mathbf{w}_f^\top \mathbf{x}_i}{||\mathbf{w}||} = \rho$$

$$\cos\theta = \langle \frac{\mathbf{w}_f}{||\mathbf{w}_f||}, \frac{\mathbf{w}_t}{||\mathbf{w}_t||} \rangle$$

其中

$$\begin{aligned}
\mathbf{w}_f^\top \mathbf{w}_t &= \mathbf{w}_f^\top (\mathbf{w}_{t-1} + y_i \mathbf{x}_i) \\
&\geq \mathbf{w}_f^\top \mathbf{w}_{t-1} + \min_{1\leq i\leq m} y_i \mathbf{w}_f \mathbf{x}_i \\
&= \mathbf{w}_f^\top \mathbf{w}_{t-1} + \gamma
\end{aligned}$$

$$\therefore \mathbf{w}_f^\top \mathbf{w}^t \geq \mathbf{w}_f^\top \mathbf{w}_{t-1} + \gamma \geq \mathbf{w}_f^\top \mathbf{w}_{t-2} + 2\gamma \geq \cdots \geq \mathbf{w}_f^\top \mathbf{w}_0 + t\gamma = t\gamma$$

注意到 $y_i \mathbf{w}_{t-1}^\top \mathbf{x}_i \Leftrightarrow \mathbf{w}$更新

$$\begin{aligned}
||\mathbf{w}_t||^2 &= ||\mathbf{w}_{t-1} + y_i \mathbf{x}_i||^2 \\
&= ||\mathbf{w}_{t-1}||^2 + 2 y_i \mathbf{w}_{t-1}^\top \mathbf{x}_i + y_i^2 ||\mathbf{x}_i||^2 \\
&\leq ||\mathbf{w}_{t-1}||^2 + ||\mathbf{x}_i||^2 \\
&\leq ||\mathbf{w}_{t-1}||^2 + \max_{1\leq i\leq m} ||\mathbf{x}_i||^2
\end{aligned}$$

$$\therefore ||\mathbf{w}_t||^2 \leq ||\mathbf{w}_{t-1}||^2 + R^2 \leq \cdots \leq ||\mathbf{w}_0||^2 + tR^2 = tR^2$$

带入 $\mathbf{w}_f^\top \mathbf{w}_t, \mathbf{w}_t$ 得

$$1 \geq \cos\theta = \langle \frac{\mathbf{w}_f}{||\mathbf{w}_f||}, \frac{\mathbf{w}_t}{||\mathbf{w}_t||} \rangle \geq \frac{t\gamma}{||\mathbf{w}_f||\sqrt{t}R} = \frac{\sqrt{t}\gamma}{||\mathbf{w}_f||R} = \frac{\sqrt{t}\rho}{R}$$

$$\therefore t \leq \frac{R^2}{\rho^2}$$

□

4. 算法

5. 迭代过程

以鸢尾花数据集为例，迭代过程如图1所示。可以看出，PLA 每次的跳跃幅度相对而言会比较大，第二次迭代到第三次迭代期间，PLA 的划分出来的线直接越过了整个区域，而第三次迭代到第四次迭代又一次性分类正确。

Algorithm 1 PLA with SGD

Input: $\mathbf{w}^0, \mathbf{x} \in \mathbb{R}^{m\times n}, \mathbf{y} \in \mathbb{R}^n$

repeat
    if $y_i\mathbf{w}^\top\mathbf{x} \le 0$ **then**
        $\mathbf{w} \leftarrow \mathbf{w} + y_i\mathbf{x}_i$
    end if
until all classified correctly

Algorithm 2 Poket with SGD

Input: $\mathbf{w}^0, \mathbf{w}^*, T, \mathbf{x} \in \mathbb{R}^{m\times n}, \mathbf{y} \in \mathbb{R}^n$

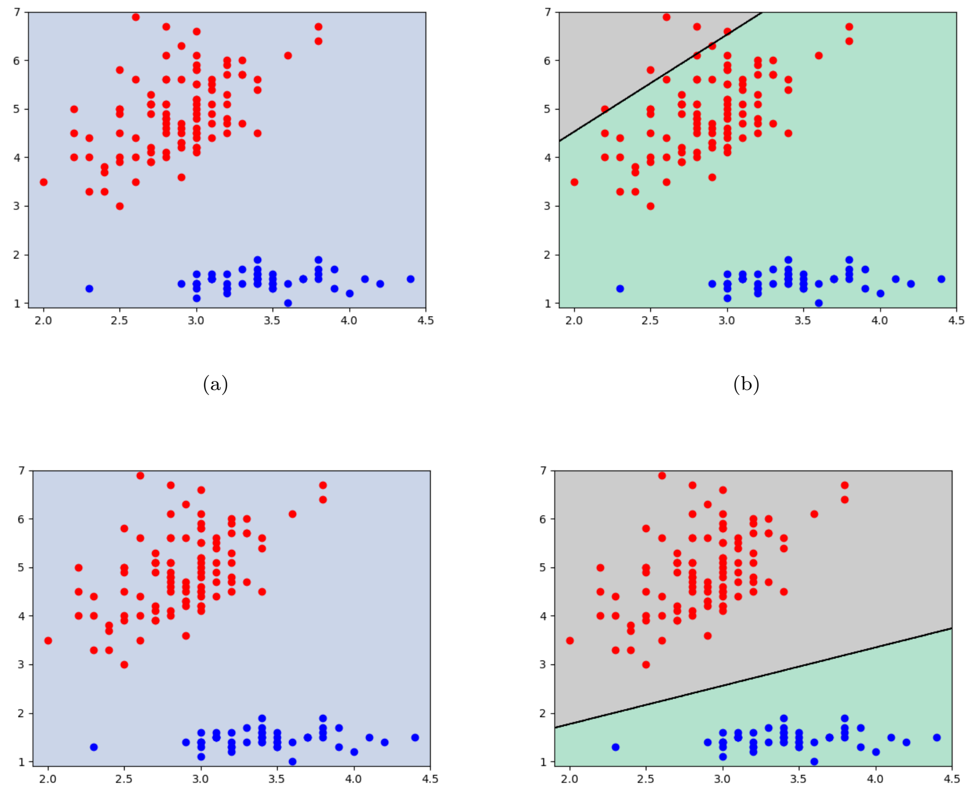
repeat
    if $y_i\mathbf{w}^\top\mathbf{x} \le 0$ **then**
        $\mathbf{w} \leftarrow \mathbf{w} + y_i\mathbf{x}_i$
    end if
    if $\sum_{i=1}^m[y_i \ne \mathrm{sign}(\mathbf{w}^\top\mathbf{x})] < \sum_{i=1}^m[y_i \ne \mathrm{sign}((\mathbf{w}^*)^\top\mathbf{x})]$ **then**
        $\mathbf{w}^* \leftarrow \mathbf{w}$
    end if
until $t > T$ or $\sum_{i=1}^m[y_i \ne \mathrm{sign}(\mathbf{w}^\top\mathbf{x})] = 0$

6. 代码

```
class PLA:
    def __init__(self, X, y):
        '''
        :param X: shape ---> (m,n) m个数据 n个参数
        :param y: shape ---> m
        '''
        self.X = X
        self.y = y
        self.m = np.shape(X)[0]
        self.n = np.shape(X)[1]
        self.w = np.zeros(self.n)
        self.b = 0

    def loss(self):
        '''
        以分类错误的点的个数作为loss
        :return:
        '''
        res = self.y*(np.sign(self.X.dot(self.w) + self.b))
        mistake = np.where(res <= 0)[0]
        loss = len(mistake)/self.m
        return loss

    def dloss(self):
        '''
```



(a) (b) (c) (d)

图 1: PLA 迭代过程

```
        以分类错误的点到超平面的距离作为loss
        :return:
        '''
        res = self.y * (np.sign(self.X.dot(self.w) + self.b))
        mistake = np.where(res <= 0)[0]
        temp = self.X.dot(self.w)+self.b
        for i in mistake:
            temp[i] = y[i]*temp[i]
        loss = -np.sum(temp[mistake])
        loss = loss/(len(mistake)+1e-7)
        return loss

    def train(self):
        while self.loss() > 0:
            res = self.y * (np.sign(self.X.dot(self.w) + self.b))
            mistake = np.where(res <= 0)[0]
            temp = self.y[mistake[0]] * self.X[mistake[0]]
            self.w += temp
            self.b += self.y[mistake[0]]
            print(f'loss1:{self.loss()}, loss:2{self.dloss()}, w:{self.w}, b:{self.b}')
```

```python
class Pocket:
    def __init__(self, X, y):
        '''
        :param X: shape ---> (m,n) m个数据 n个参数
        :param y: shape ---> m
        '''
        self.X = X
        self.y = y
        self.m = np.shape(X)[0]
        self.n = np.shape(X)[1]
        self.w = np.zeros(self.n)
        self.b = 0
        self.best_loss = float('inf')
        self.step = 0

    def loss(self):
        res = self.y*(np.sign(self.X.dot(self.w) + self.b))
        mistake = np.where(res <= 0)[0]
        loss = len(mistake)/self.m
        return loss

    def train(self):
        while self.loss() > 0:
            res = self.y * (np.sign(self.X.dot(self.w) + self.b))
            mistake = np.where(res <= 0)[0]
            temp = self.y[mistake[0]] * self.X[mistake[0]]
            self.w += temp
            self.b += self.y[mistake[0]]
            curr_loss = self.loss()
            self.step += 1
            if curr_loss < self.best_loss:
                self.best_loss = curr_loss
                print(self.loss(), self.w, self.b)
            if self.step > 1000:
                break
```