AI Lab2

范翔宇 PB18000006

线性分类算法

迭代方法：

Solving Least Squares Classification

Let

$$\mathbf{X} = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1d} \\ \vdots & & \vdots & \\ 1 & x_{N1} & \cdots & x_{Nd} \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix} b \\ \vdots \\ w_d \end{bmatrix}$$

$$\begin{aligned} \text{Loss} = \min_{\mathbf{w}} (\mathbf{y} - \mathbf{X}\mathbf{w})^2 &= \min_{\mathbf{w}} (\mathbf{X}\mathbf{w} - \mathbf{y})^2 \\ &= \min_{\mathbf{w}} (\mathbf{X}\mathbf{w} - \mathbf{y})^\top (\mathbf{X}\mathbf{w} - \mathbf{y}) \end{aligned}$$

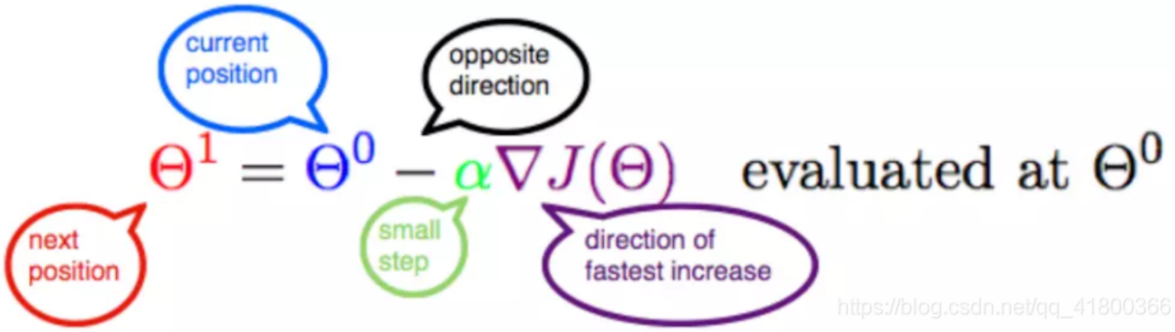
梯度：

Solving for w

$$\begin{aligned} \frac{\partial \text{Loss}}{\partial \mathbf{w}} = 2(\mathbf{X}\mathbf{w} - \mathbf{y})^\top \mathbf{X} = 0 \\ \mathbf{X}^\top \mathbf{X}\mathbf{w} - \mathbf{X}^\top \mathbf{y} = 0 \\ \mathbf{w}^* = \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top \mathbf{y} \end{aligned}$$

其中对应正则项为$2\omega\lambda$

梯度下降法：



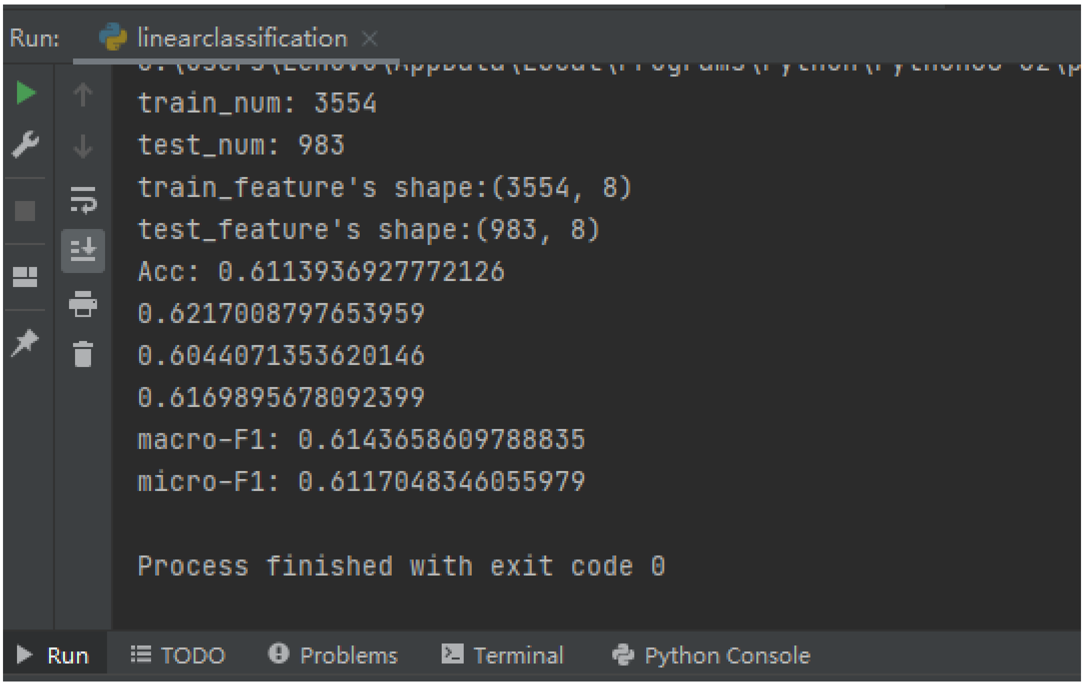
具体参考如下链接：https://blog.csdn.net/anycall201/article/details/111177055和https://blog.csdn.net/qq_41800366/article/details/86583789

之后就转换成代码实现：

fit部分：初始化时，在X最左侧添加一列(其中元素全为1)，并且设置w中元素全为0.001。之后迭代时，先计算出$(Xw-y)^TX$，再加上对应正则项，之后利用梯度下降法进行更新w。

predict部分：将扩充后的矩阵与w相乘，得到的结果进行四舍五入后取整即可。

预测结果如图所示：



准确率达到：61.13936927772126%

朴素贝叶斯分类器

大体思路参考课程主页讲解

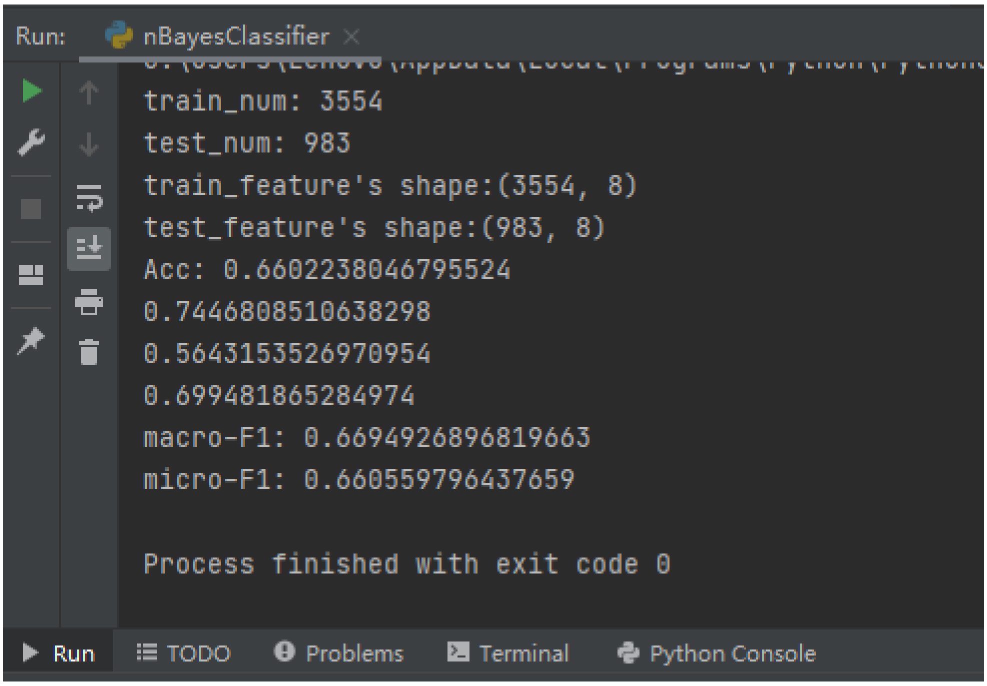
fit部分：采用方法一把连续属性离散化,用相应的离散区间替换连续属性值。计划把每个连续属性都划分为100个区间。为了划分，要先找出每个属性的上界和下界，确定区间即$100*\frac{data-lowerbound}{upperbound-lowerbound}$，并进行四舍五入和取整。之后需要统计$D_c$和$D_{cx}$，直接遍历判断即可。结合下方公式计算先验概率的时候，进行了取对数操作，避免之后相乘为0。计算条件概率进行了同样处理。

$$\hat{P}(c) = \frac{|D_c + 1|}{|D| + N},$$
$$\hat{P}(x_i|c) = \frac{|D_{c,x_i}| + 1}{|D_c + N_i|}$$

predict部分：结合如下公式，由于进行了取对数操作，所以具体实现时是累加。遍历数据累加之后寻找max即可。

$$h_{nb}(x) = \arg\max_{c \in Y} P(c) \prod_{i=1}^{d} P(x_i|c)$$

预测结果如图所示：



SVM

参考PPT

The Optimization Problem

- The dual of this new constrained optimization problem is

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \left(\mathbf{x}_i^\top \mathbf{x}_j \right)$$

$$\text{subject to} \quad \forall i,\ 0 \le \alpha_i \le C,\ \sum_{i=1}^{n} \alpha_i y_i = 0$$

- This is very similar to the optimization problem in the linear separable case, except that there is an upper bound C on α_i now
- Once again, a QP solver can be used to find α_i

Summary: Support Vector Machines

- SVM training: build a kernel matrix K using training data
  - Linear: $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^{\mathbf{T}} \mathbf{x}_j$
  - Gaussian (radial-basis function 径向基函数): $K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{\sigma^2}}$
- Solve the following quadratic program :

$$\max_{\alpha} \alpha^\top \mathbf{e} - \frac{1}{2} \alpha^\top \left(\mathbf{y}\mathbf{y}^\top \circ K \right) \alpha$$

$$\text{subject to} \quad \forall i,\ 0 \le \alpha_i \le C,\ \sum_{i=1}^{n} \alpha_i y_i = 0$$

- SVM testing: now with α_i, recover b

$$b = y_i - \sum_{j=1}^{n} \alpha_j y_j K(\mathbf{x}_i, \mathbf{x}_j) \qquad \text{for any } i \text{ that } \alpha_i \neq 0$$

- we can predict new data points by:

$$y^* = \text{sign}\left(\sum_{i \in SV} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}') + b \right)$$

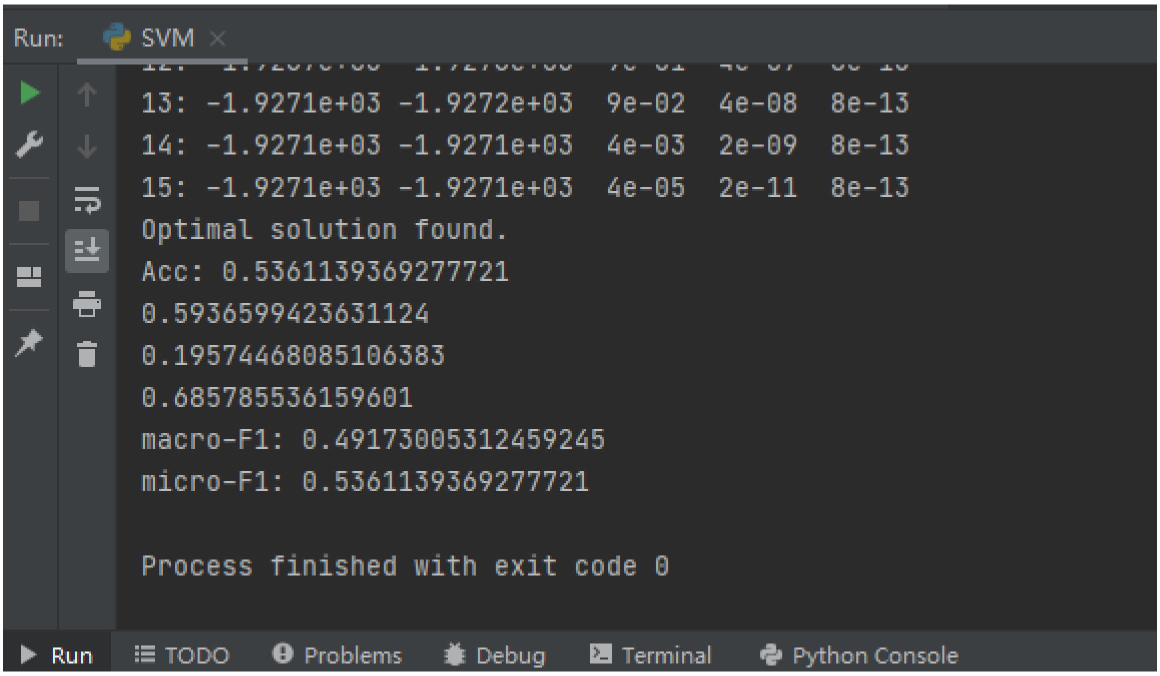
使用cvxopt.solvers.gp求解线性规划具体参考https://blog.csdn.net/qq_45669448/article/details/104678910

其中对应关系为X = $(\alpha_1, \dots \alpha_n)^T$,A = $(y_1, \dots y_n)$,P = $(y_i y_j K(x_i, x_j))$, q = (-1, ..., -1), G = $(I, -I)^T$, h = $(c, \dots, c, 0, \dots 0)^T$， b=0

代码即大致复现

参考链接：https://www.cnblogs.com/massquantity/p/11110397.html和https://blog.csdn.net/viewcode/article/details/12840405

预测结果如图所示：



手写感知机模型并进行反向传播 复现MLP-Mixer

时间不够，放弃了，希望助教可以高抬贵手

实验总结

本次实验带我系统了解了learning部分，让我清晰地认识到不同算法的运行时间与准确率的差距，也让我感受到了AI和python的魅力，但是说实话我还是提不起对AI的兴趣，可能注定无缘。也更加明确了以后的方向，同时也庆祝本科阶段最后一个实验终于结束了()