Assignment-1

线性分类

复旦大学计算机系

1. 概要

   项目主要使用 Numpy/Matplotlib 包，完成了正态分布构建和采样、数据集构建和读取、以最小平方误差/感知器/Logistic 算法构建了三个线性判别式模型、以 Logistic 算法构建了一个线性生成式模型，并通过调整数据集大小/规模/分布重叠的手段对于模型进行分析。

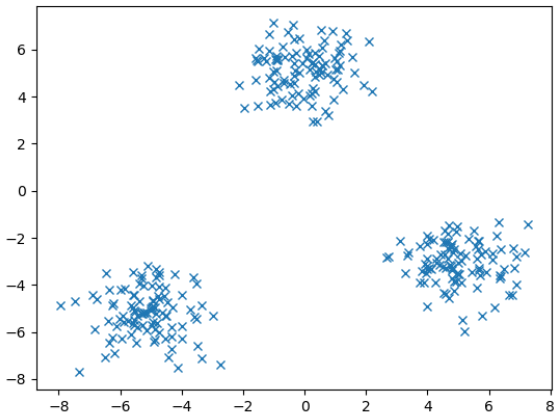
2. 正态分布的构建和采样

   利用舍选法，通过 Numpy 包所带的[0,1]之间的均匀分布生成随机数进行实现。单个正态分布的构建与采样封装在 multivariate_normal 函数之内，同时在函数 normal_distribution_generate 之中进一步封装，使得后者能够处理多个多维正态分布的生成。

   构建方式是：利用 np 的 pi/linalg.det 方法按照正态分布公式计算正态分布下的值。但由于构建主要用于采样，这里为了计算方便只计算了 e 的幂值。

   采样方式是：假设对于正态分布下坐标为参数 mean 的点满足采样率为 100%，即正态分布概率最大值一定采样。先利用均匀分布生成一定范围之内的随机坐标(a, b)，计算参数给定的正态分布下该坐标的概率 P，然后将 P 除以正态分布概率最大值 Pmax 获得接受概率 P'。此时 P'属于[0,1]，即对于该正态分布下的 1 个落在坐标（a,b）的点，采样时有 P'的概率接受，也有 1-P'的概率拒绝，再利用均匀分布生成一个[0,1]之间的随机数 U，如果 U<=P', 则其被接受, 加入列表, 反之 U 不被接受继续上述过程直到样本数量足够。这里由于最大值 Pmax 即为公式 exp 项之前的常系数项，因此可以在计算时不考虑常系数项，直接将 exp 项与 1 的比值作为 P 与 Pmax 的比值（可以证明 P'本身也是正态分布的密度函数）

   以下为二维情况下采样的可视化图：



图表 1-三个正态分布采样

3. 数据集构建方法

数据集的构建以 numpy.ndarray()为主要格式，利用 np.savetxt 与 np.loadtxt 方法作为保存与存储的手段，并且将两者各自封装在函数 save_dataset 与 load_dataset 之内，默认存储的文件名为"data.data"。

4. 最小平方线性判别模型

a) 模型构建

该模型的构建基于平方和误差函数进行优化，对于损失函数进行求导并令导数为 0，可以获得判别函数 y(x)的形式，如下：

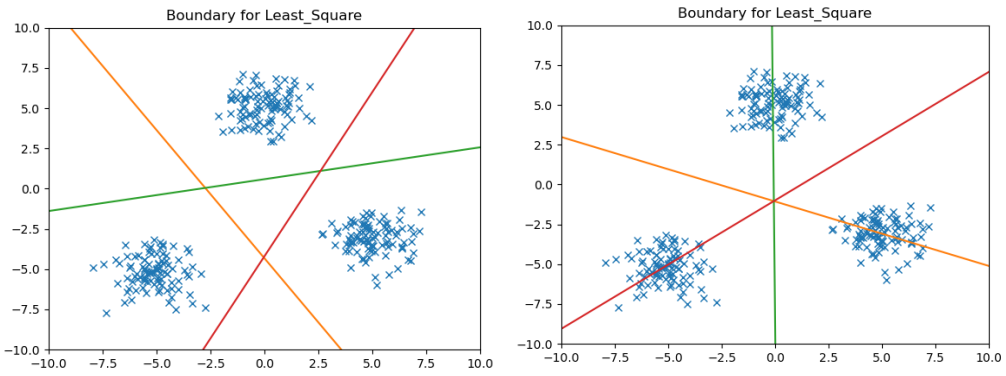
$$y(x) = \widetilde{W}^T \tilde{x} = T^T (\widetilde{X}^\dagger)^T \tilde{x}$$

其中参数矩阵为扩展的包含向量 w_{k0} 量的矩阵，其表达式扩展的 X^矩阵的伪逆：

$$\widetilde{W} = (\widetilde{X}^T \widetilde{X})^{-1} \widetilde{X}^T T = \widetilde{X}^\dagger T$$

在代码中利用该式能精准确定参数，这是该模型的一大优势。

同时通过在目标矩阵 T 中进行扩展能够将分类性质进行更改，进而模型的线性边界类型进行替换，如单独线性分类器是分出唯一类（一对多），还是只分两类中的一个（多对一）：

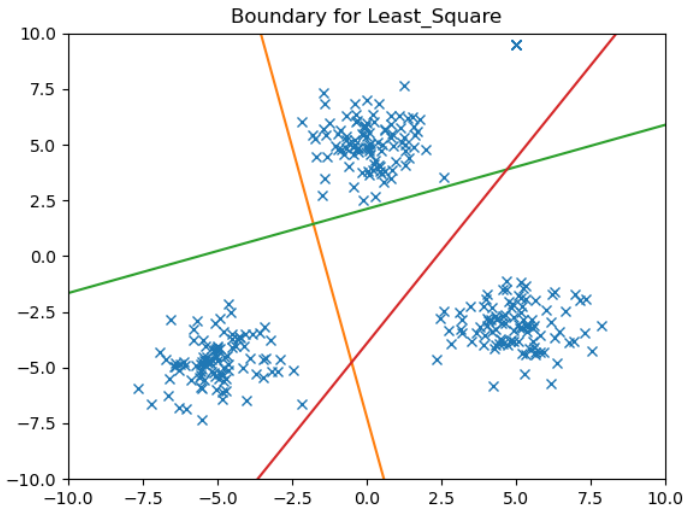


图表 2-一对多 MSE（左）与一对一 MSE（右）

其中左图中一个线性分类器分出一类，右图中两个分类器共同分出一类。

b) 模型分析

i. 鲁棒性：



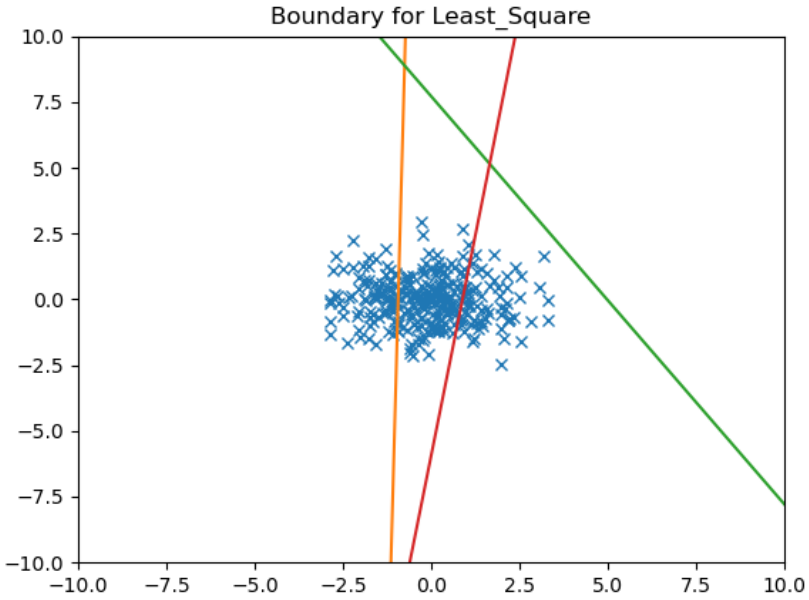
通过设置奇异点可以发现，模型的相对于奇异点的改动有些过分，为了整体的误差总和最小可能会为了少数奇异点去牺牲大部分点的分类正确性。

同时最小方差以高斯分布作为假设，如果换成其他分布效果可能差很多。

同时也无法生成数据，只能用于分类问题。

但优势是参数数量小非常，只需要 #类数*（#变量维度+1）个参数，相对于生成式模型而言很便宜。

当数据重叠时效果不好：



5. 感知器模型

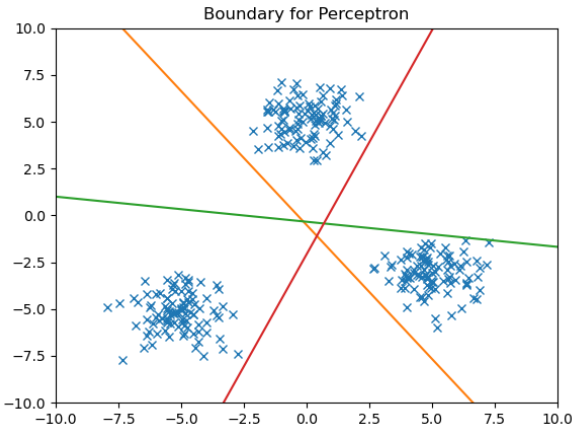
a) 模型构建

感知器的构建基于错误驱动以及被动式更新，其损失函数基于错误的样本构成，主要形式如下：

$$E_P(w) = -\sum_{n \in M} w^T \phi_n t_n$$

其中 t_n 为模型标签，Φn 为非线性变换获得的特征向量，w^T 为当前权重矩阵。
感知器的本质是超平面法向量不断加上错误的点的投影向量，并且由于有训练集线性可分条件，感知器能在有限步内收敛的结论，感知器在特征提取之后的适用性很强。

代码实现时也利用了该点性质，对于数据的每个样本与其标签进行迭代，并在循环过程中判断线性分类器是否正确的划分好每个标签，如果错误则分情况对于分类器的权重矩阵进行加或减的更新，在循环将权重矩阵的更新前后的状态进行对比。
以下是数据集线性可分时的可视化图像：



图表 3-感知器分类

b) 模型分析
对线性不可分模型而言很容易陷入死循环，无法生成数据，而且只能用于良好的分类问题，在特征抽取不到位的情况下效果很差
虽然方法十分边界，参数数量也很小。

6. Logistic 线性判别式模型
a) 模型构建
模型基于 Logistic 函数在多分类任务下的扩展形式：

$$p(C_k|x) = \frac{p(x|C_k p(C_k))}{\sum_j p(x|C_j)p(C_j)} = \frac{exp(a_k)}{\sum_j exp(a_j)}$$

其中有：

$$a_k = ln\ (p(x|C_k)p(C_k))$$

a_k 作为模型先验和似然乘积的 ln 值，在先验假设为正态分布的形式下，其常常被假设为 x 线性变换后的结果。

$$a_k(x) = w_k^T x + w_{k0}$$

注意到这一简化导致了参数量的大幅减少，模型不能生成数据，但是往往在决策边界判断时迭代会更快。
程序中使用了梯度下降方法来对参数矩阵 W 和偏差值 b 进行优化，其中对于 W 与 b 的梯度是程序内显式定义的，有如下的 Loss 函数：

对于标量损失函数 l 而言，**y** 为 one-hot 形式的 m*1 列向量，**W** 是 m*n 矩阵，**x** 是 n*1 列向量，softmax(a) = $\frac{\exp(\boldsymbol{a})}{\mathbf{1}^T \exp(\boldsymbol{a})}$，其中 **1** 代表全 1 向量，exp(**a**)表示逐元素求指数。

$$l = -\boldsymbol{y}^T\left(\log(\exp(\boldsymbol{Wx})) - \mathbf{1}log(\mathbf{1}^T \exp(\boldsymbol{Wx}))\right)$$

$$= -\boldsymbol{y}^T\boldsymbol{Wx} + \log(\mathbf{1}^T \exp(\boldsymbol{Wx}))$$

$$dl = tr\left(-\boldsymbol{y}^T d\boldsymbol{Wx} + \frac{\mathbf{1}^T\left((\exp(\boldsymbol{Wx}) * (d\boldsymbol{Wx})\right)}{\mathbf{1}^T \exp(\boldsymbol{Wx})}\right)$$

$$= tr(-\boldsymbol{y}^T d\boldsymbol{Wx} + softmax(\boldsymbol{Wx})^T d\boldsymbol{Wx})$$

$$= tr(\boldsymbol{x}(softmax(\boldsymbol{Wx}) - \boldsymbol{y})^T d\boldsymbol{W})$$

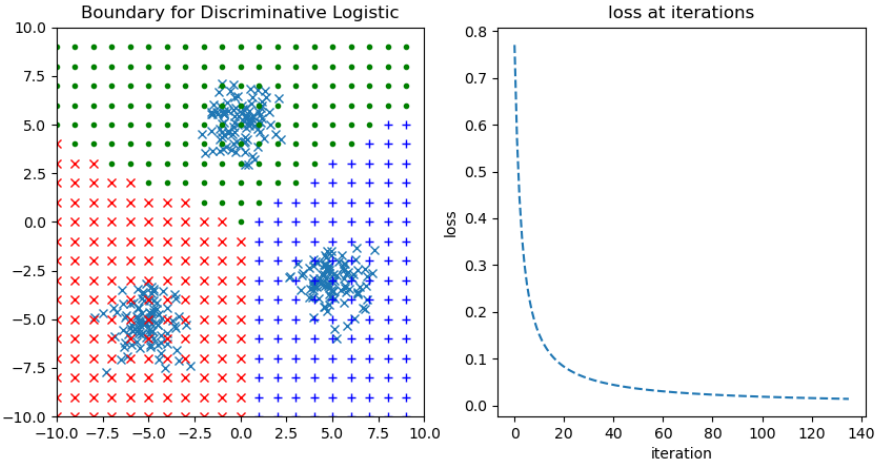
有：$\frac{\partial \boldsymbol{l}}{\partial \boldsymbol{w}} = (softmax(\boldsymbol{Wx}) - \boldsymbol{y})\boldsymbol{x}^T$

同理可以有：

$$\frac{\partial \boldsymbol{l}}{\partial \boldsymbol{b}} = (softmax(\boldsymbol{Wx}) - \boldsymbol{y})$$

在代码实现时，定义了 softmax/cross_entropy/gradient 分别用于求不同的部分值，其中会与公式有一些区别，如：softmax 中可能出现 exp(a)过大而导致溢出的情况，因此会将 a 先行减去平均值将数值减小。

以下为判别式模型的决策边界示意图和梯度下降时 Loss 函数的变化趋势：

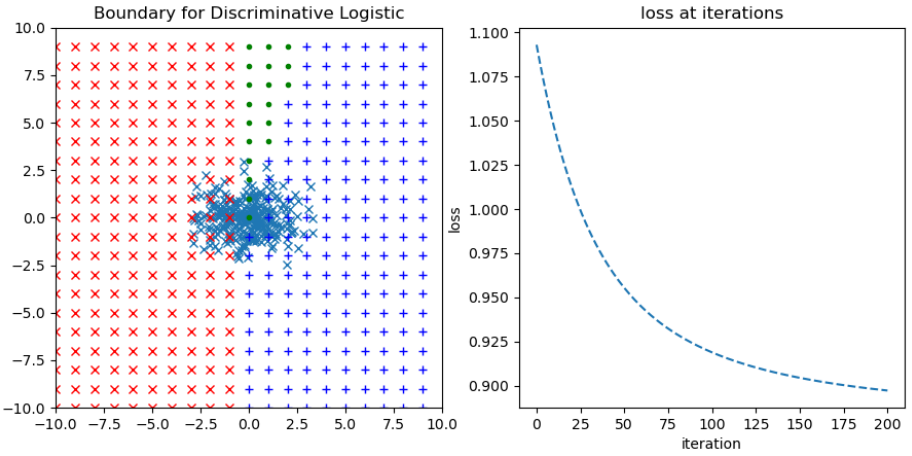


可以看到此时不同类数据之间的决策边界保持线性。

b) 模型分析

与生成式模型相比拥有更少的参数，能更快的学习和迭代，抗奇异值能力强。

相比前两个模型在数据交杂重叠较多时效果不好：



7. Logistic 线性生成式模型

a) 模型构建

Logistic 生成式模型的构建与判别式模型的构建最大不同点在于：不采取 ak 作为 x 线性变换的捷径，而是直接将损失函数的梯度与正态分布的平均值向量和协方差矩阵的梯度进行关联：

定义

$$\boldsymbol{a} = \sum_{k=1}^{K} \boldsymbol{B}_k \left((\sum^{-1}\mu_k)^T x - \frac{1}{2}{\mu_k}^T \sum^{-1}\mu_k \; + \; ln\, p(\mu_k) \right)$$

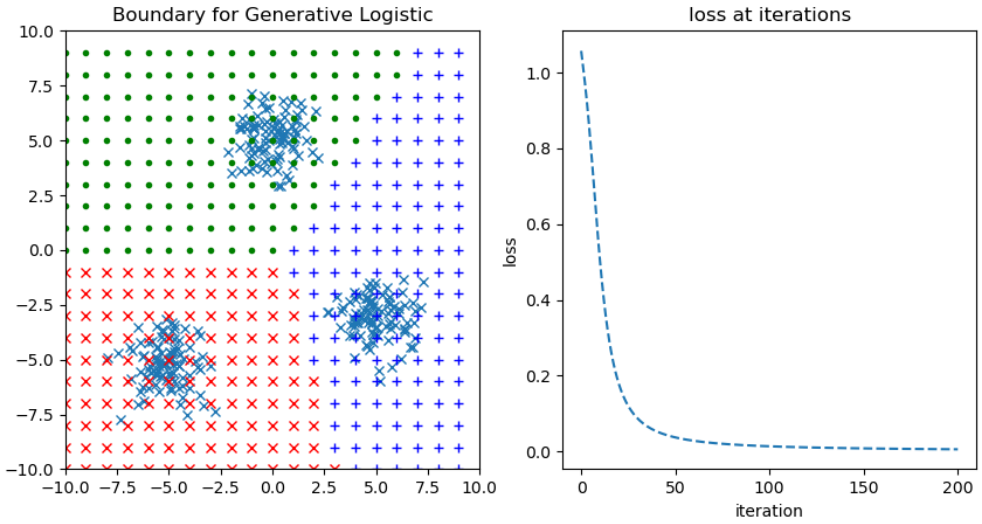
其中 Bk 为第 k 行为 1 其余为 0 的 c*1 列向量，c 表示类的个数

这里导出的公式为：

$$\frac{\partial \boldsymbol{l}}{\partial \sum^{-1}} = (\sum_{k=1}^{K} \; (\mu_k {B_k}^T \frac{\partial \boldsymbol{l}}{\partial \boldsymbol{a}} \boldsymbol{x}^T - \frac{1}{2}(\mu_k \frac{\partial \boldsymbol{l}^T}{\partial \boldsymbol{a}} \; B_k {\mu_k}^T)^T$$

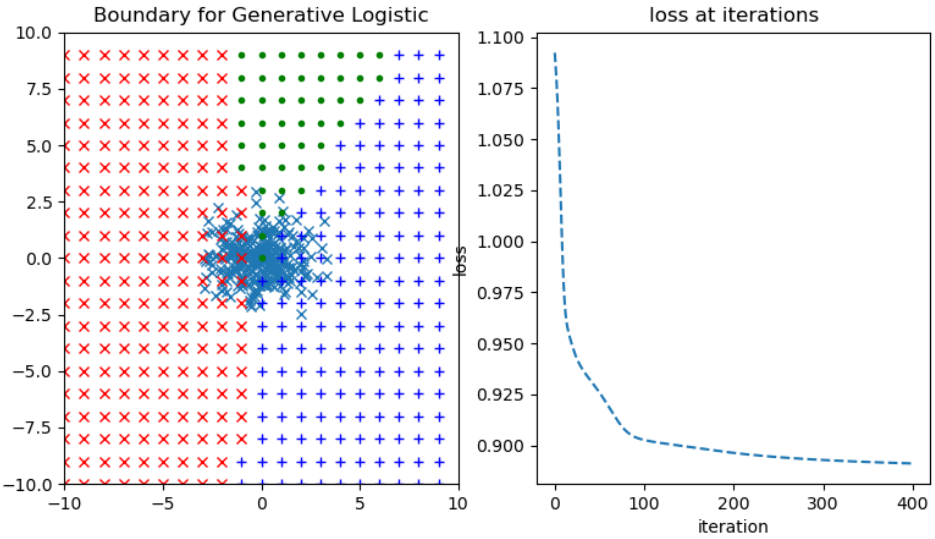
$$\frac{\partial \boldsymbol{l}}{\partial \mu_k} \; = \; (\sum_{k=1}^{K} \; (B_k{}^T \frac{\partial \boldsymbol{l}}{\partial \boldsymbol{a}} \boldsymbol{x}^T \sum^{-1} - \frac{1}{2}(B_k{}^T \frac{\partial \boldsymbol{l}}{\partial \boldsymbol{a}} {\mu_k}^T \sum^{-1^T}) - \frac{1}{2}\frac{\partial \boldsymbol{l}^T}{\partial \boldsymbol{a}} \; B_k {\mu_k}^T \sum^{-1})^T$$

以下为生成式模型的决策边界示意图和梯度下降时 Loss 函数的变化趋势



b) 模型分析

生成式模型的效果也不够好:



虽然生成式模型能够生成数据，但是这里的问题在于数据本身并非是回归式的而是分类式的，生成式模型并不能还原产生数据的高斯模型，logistic 生成模型最后 loss 接近 0 后返回的协方差矩阵和均值矩阵都和源数据相差很大。我认为这是由于数据本身的问题，数据本身带有的标签信息是类的信息，这导致模型再好也只能在分类任务中学习分类边界，从这点角度而言，分类的好坏并不意味着生成数据的好坏，而且生成模型需要更多的参数，同时需要更好的假设，从这一点上来看，判别式模型在分类问题上有它自身的优势。