Assignment 1

2020 年 7 月

目 录

1 Part 1

代码中 `getNorm()` 函数用来生成给定参数的正态分布数据， `getDataset()` 调用前者生成三个不同规模的二维正态分布。 `plot()` 函数将给定的数据和标签画出散点图（如果提供 `file` 参数，则会将图保存到对应名称的文件中）。

初始正态分布数据的中心为分别为 (0,0),(0,2),(0,4)。x,y 方向的标准差为 1，相关系数为 0。生成的数量分别为 100,200,150。

2 Part 2

`GenerativeModel` 和 `DiscriminativeModel` 这两个类分别对应两种模型。

`GenerativeModel` 根据概率论的最大似然估计，对于某个分类 k 的正态分布预测，其均值 $\mu = \frac{1}{n_k}\sum_{i=1}^{n_k} x_i$，而协方差矩阵 $\Sigma_k = \frac{1}{n_k}\sum_{i=1}^{n_k}(x_i-\mu)(x_i-\mu)^T$。由此得到每个分类的正态分布。最后根据数据中各个分类的数目可以计算出各个分类出现的概率 $P(k) = \frac{n_k}{N}$。在预测的时候，利用贝叶斯公式可以计算出给定 x，属于各个分类的概率 $P(k|x) = P(x|k)P(k)$，取概率最大的类别为预测分类。

`DiscriminativeModel` 使用 Softmax 回归，使用交叉熵作为损失函数。将权重矩阵 W 的维度加一来把偏置 bias 合并到 W 中。在计算 Wx 时，将原有向量 x 扩增一个 1 后再计算 Wx。预测各分类概率 $y' = softmax(W^T x)$，取概率最大的类别为预测分类，权重矩阵 W 更新的公式为 $W \leftarrow W + \alpha\sum_{n=1}^{N} x_i(y_i - y_i')^T$。

训练时，按照 70% 训练集，30% 测试集进行数据的划分，测试数据集真实分类如Figure 1，可以看到不同分类数据之间有所混合。 `GenerativeModel` 和 `DiscriminativeModel` 两种模型的

预测结果如Figure 2的左右两个图。这两个模型的预测结果十分相似，训练出来的准确率分别为 0.756 和 0.778。

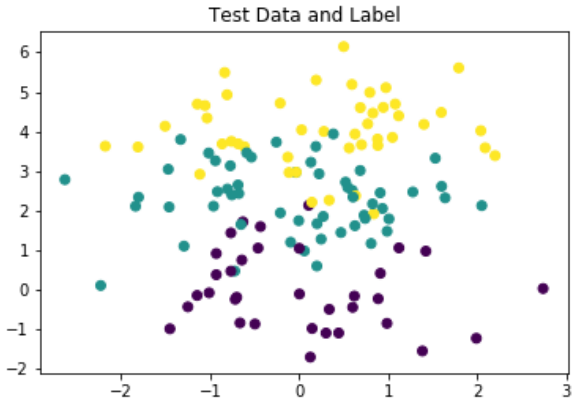

Figure 1: 测试数据的真实分类散点图

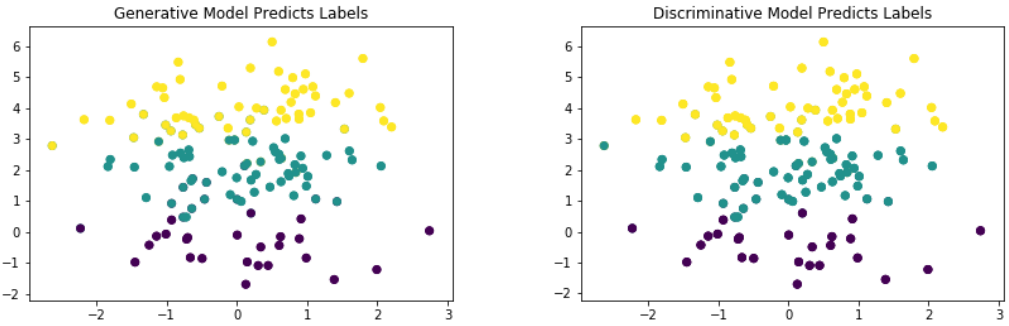

Figure 2: 两种模型的预测分类散点图

两个模型的区别在于，Generative Model 需要预先给定数据的分布概率模型，对数据分布有假设条件。而 Discriminative Model 没有对数据分布进行假设，通过多轮的训练调整权重矩阵逐渐找到一种正确率更高的分类方式，多轮训练也导致这种模型需要更多的时间进行训练。

3 Part 3

当把正态分布的中心设为 (-1,0),(1,0),(0,1.5) 时，分类结果如Figure 3，可以看到两个模型的预测结果都是每个颜色分类边界都是非常明显的，但是实际的数据在中间却是各种颜色交错的，这两个模型都无法正确地对中间交错的数据进行准确分类。此时 Generative Model 和 Discriminative Model 的准确率都为 0.7185。

在此基础上增大方差到 2，结果如Figure 4，数据更加混杂，两个模型的准确率也下降到 0.540 和 0.548。

在以上改变基础上再改变不同数据的规模大小，中心分别为 (1,0),(-1,0),(0,1.5) 的三个正态分布原数据规模为 100,200,150，改变为 100,600,300 后，结果如Figure 5，此时两个模型的准确率都是 0.683。可以看到，左下角以 (-1,0) 为中心的分类范围明显大于其它两个分类，无论是和Figure 4对比还是在这三个图中对比。这说明某个分类的数据量越多，在分类交界处其范围也会相对更大，对于两种模型都存在这个现象。

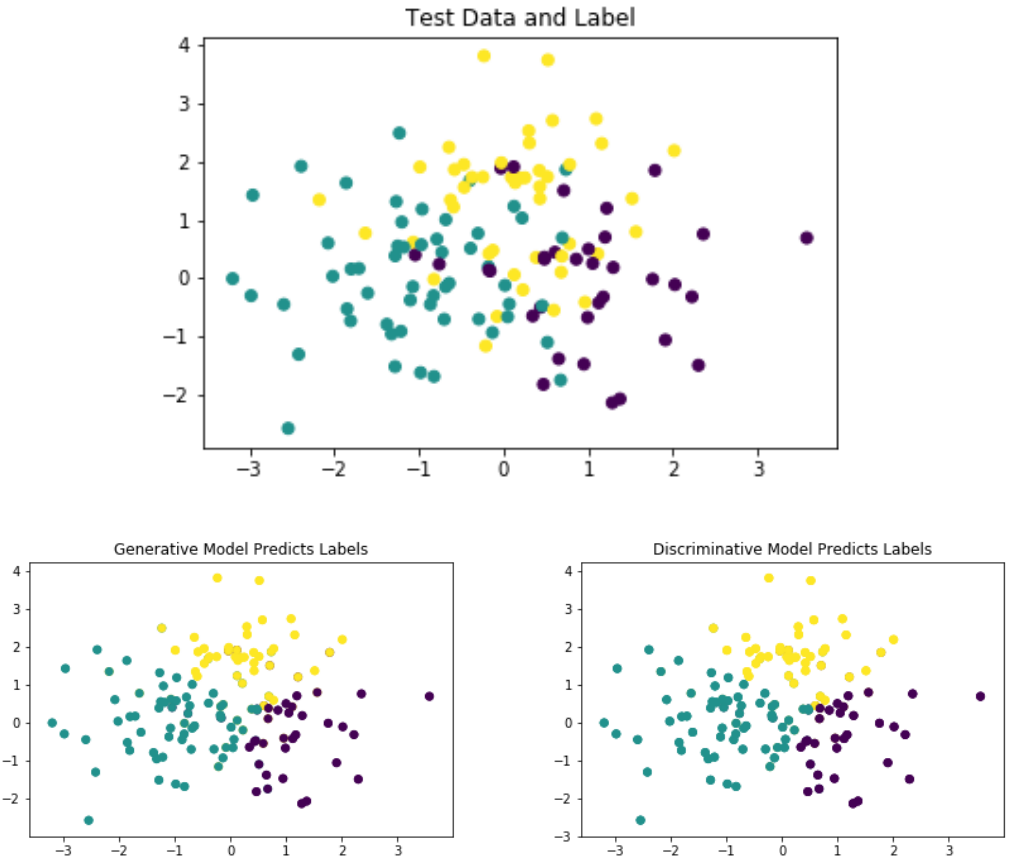

Figure 3: 改变正态分布中心

4 运行代码

使用命令 `python source.py` 就能运行代码,会在当前目录下生成三个图片：`test-dataset.png` , `GenerativeModel.png` , `DiscriminativeModel.png` 分别为测试数据集和两个模型预测结果的分类散点图。

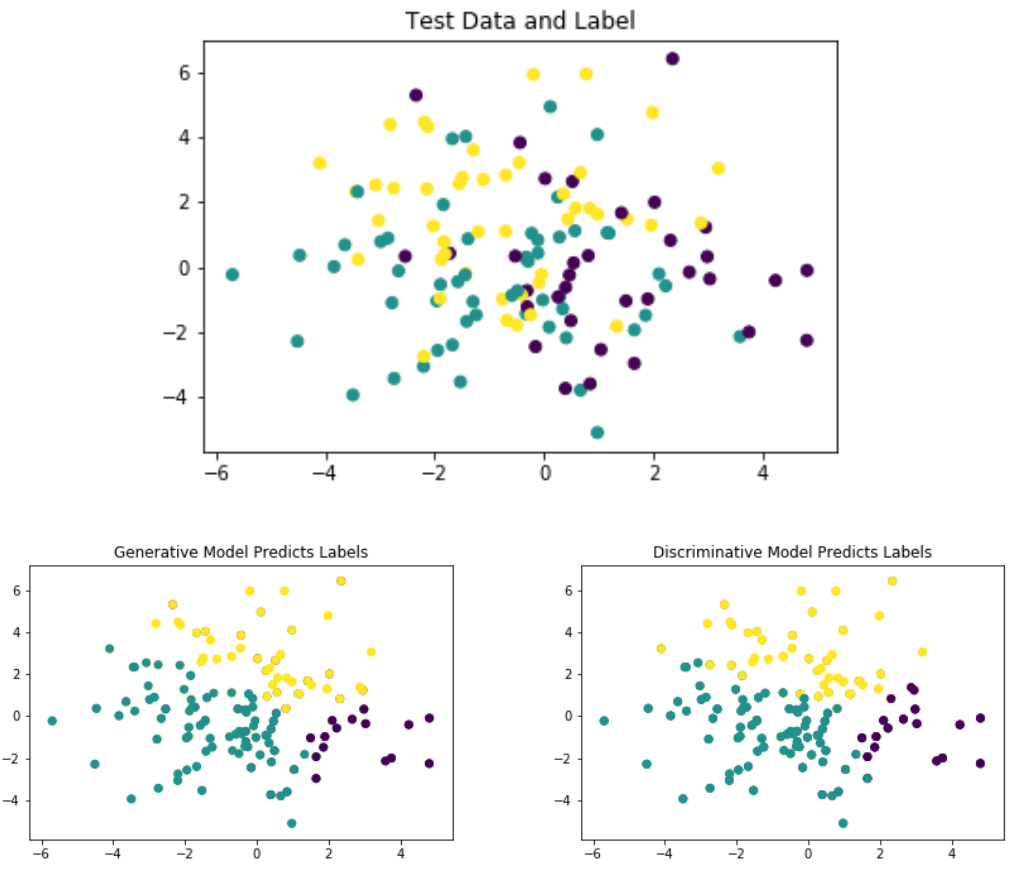

Figure 4: 改变正态分布方差

Figure 5: 改变不同分类数据规模