PRML Assignment 1

薛振梁　18307130172

1. 数据生成

使用以下命令行生成一个基本数据：

```
./source.py generate -a="-0.5,-0.5,0.15,1" -b="0.4,-0.3,0.2,1" -c="-0.2,0.4,0.15,1" 100000
```

将会输出 10^5 个数据点到文件 `a.data`。可以从这里下载：https://riteme.site/aha.data

2. 模型设置

2.1 生成式模型　目标是要预测出指定点 $\mathbf{x}$ 处每种标签的概率 $\Pr[C_j|\mathbf{x}]$。根据 Bayes 公式：

$$\Pr[C_j|\mathbf{x}] \propto \Pr[\mathbf{x}|C_j]\Pr[C_j]$$

其中先验概率 $\Pr[C_j]$ 用数据集中 C_j 出现的频率作为估计。对于 $\Pr[\mathbf{x}|C_j]$ 假设其满足正态分布：

$$\Pr[\mathbf{x}|C_j;\mathbf{c}_j,\sigma_j] = \frac{1}{2\pi\sigma_j}\exp\left(-\frac{1}{2\sigma_j}\|\mathbf{x}-\mathbf{c}_j\|^2\right)$$

这里参数 $\mathbf{c}_j$ 表示正态分布的中心，σ_j 表示方差。于是整个数据集 $\{\mathbf{x}_i\}$ 的似然函数为：

$$P = \prod_{j=1}^{3}\prod_{i\in C_j}\Pr[\mathbf{x_i}|C_j]$$

考虑最大化 $\ln P$

$$\ln P = -\sum_{j=1}^{3} n_j\ln\sigma_j - \sum_{j=1}^{3}\frac{1}{2\sigma_j}\sum_{i\in C_j}\|\mathbf{x}_i-\mathbf{c}_j\|^2 + \text{const.}$$

因为

$$\frac{\partial(\ln P)}{\partial\mathbf{c}_j} = -\frac{1}{\sigma_j}\sum_{i\in C_j}(\mathbf{x}_i-\mathbf{c}_j)$$
$$\frac{\partial(\ln P)}{\partial\sigma_j} = -\frac{n_j}{\sigma_j} + \frac{1}{2\sigma_j^2}\sum_{i\in C_j}\|\mathbf{x}_i-\mathbf{c}_j\|^2$$

于是可以得到极值点：

$$\mathbf{c}_j = \frac{1}{n_j}\sum_{i\in C_j}\mathbf{x}_i$$
$$\sigma_j = \frac{1}{2n_j}\sum_{i\in C_j}\|\mathbf{x}_i - \mathbf{c}_j\|^2$$

故直接对数据集进行计算即可。预测时选择后验概率最高者作为标签的预测值。严格来说上述模型并不是线性模型。考虑两个标签 C_i 和 C_j 之间的决策边界：

$$\begin{aligned}
&\Pr[\mathbf{x}|C_i] = \Pr[\mathbf{x}|C_j] \\
\Rightarrow\ &\frac{1}{\sigma_i}\exp\left(-\frac{1}{2\sigma_i}\|\mathbf{x}-\mathbf{c}_i\|^2\right) = \frac{1}{\sigma_j}\exp\left(-\frac{1}{2\sigma_j}\|\mathbf{x}-\mathbf{c}_j\|^2\right) \\
\Rightarrow\ &-\frac{1}{2\sigma_i}\|\mathbf{x}\|^2 + \mathbf{a}_i^T\mathbf{x} + b_i = -\frac{1}{2\sigma_j}\|\mathbf{x}\|^2 + \mathbf{a}_j^T\mathbf{x} + b_j
\end{aligned}$$

因此当 $\sigma_i \neq \sigma_j$ 时，决策边界应该是二次曲线。当然，可以令 $\sigma_1 = \sigma_2 = \sigma_3$，这样得到的就会是一个线性模型。生成式模型源代码参见 `GenerativeModel` 类。

2.2 判别式模型　该模型使用一个 3×3 的系数矩阵 $\mathbf{W}$，将输入向量 $\mathbf{x}$ 添加全为 1 的一维，使得 $\tilde{\mathbf{x}} = (\mathbf{x}_1, \mathbf{x}_2, 1)$，并且输出 $\{\mathbf{y}_i\}$ 使用 one-hot 编码。模型尝试直接预测分类

$$\tilde{\mathbf{y}}(\mathbf{x}) = \mathrm{softmax}(\mathbf{W}^T\tilde{\mathbf{x}})$$

使用交叉熵损失函数

$$L = \sum_{i=1}^{n}\mathbf{y}_i^T\tilde{\mathbf{y}}(\mathbf{x}_i)$$

以及使用小批量随机梯度下降法来学习参数 $\mathbf{W}$。迭代方式为：

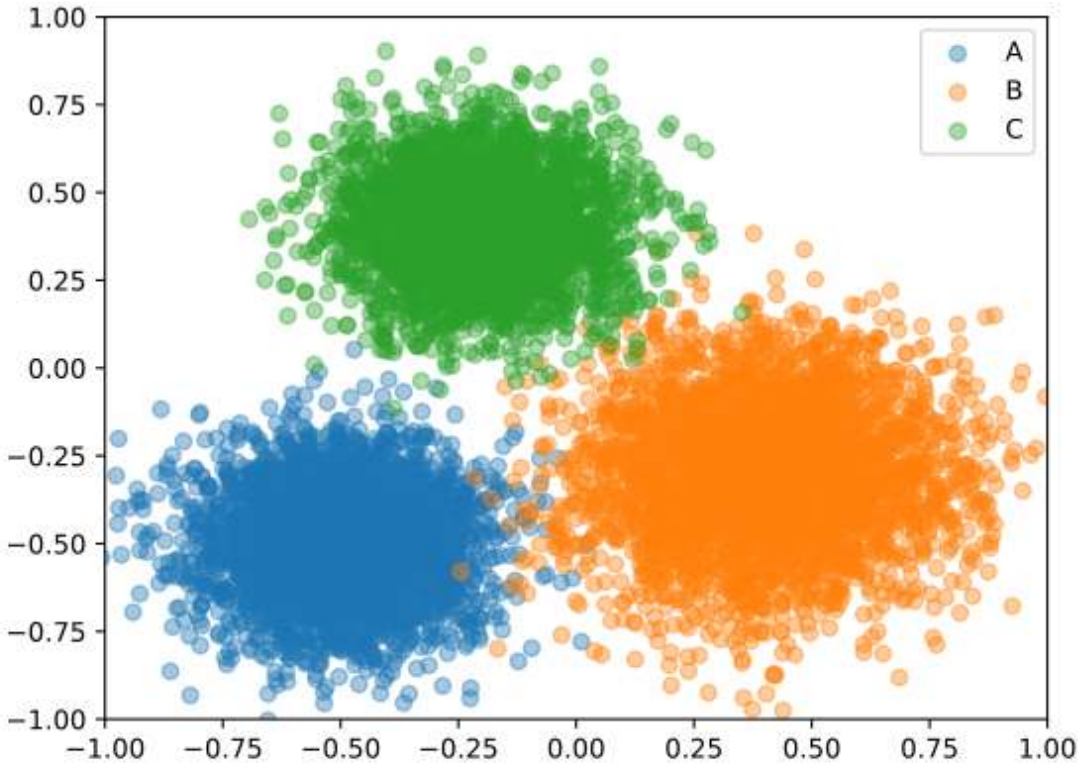
$$\mathbf{W}^{(i+!)} = \mathbf{W}^{(i)} + k\sum_{j=1}^{m}\mathbf{x}_j(\mathbf{y_i} - \tilde{\mathbf{y}}(\mathbf{x}_j))^T$$

其中 m 表示每个批次更新的数据数量，代码中使用的是 64；k 是学习率，是一个单调下降的函数，可通过另外一个参数 d 来控制下降速度。学习时将数据集分为两部分，其中一小部分作为测试集，在学习过程中，如果在测试集上的正确率变化不大时停止学习。最终用学习过程中测试集上正确率最高的参数作为学习结果。判别式模型源代码参见 `DiscriminativeModel` 类。

3. 模型测试

3.1 “三角形”式数据　这是一个简单的测试。使用以下命令行生成数据：

```
./source.py generate -a="-0.5,-0.5,0.15,1" -b="0.4,-0.3,0.2,1" -c="-0.
2,0.4,0.15,1" 10000
```



运行模型代码:

```
./source.py run a.data
```

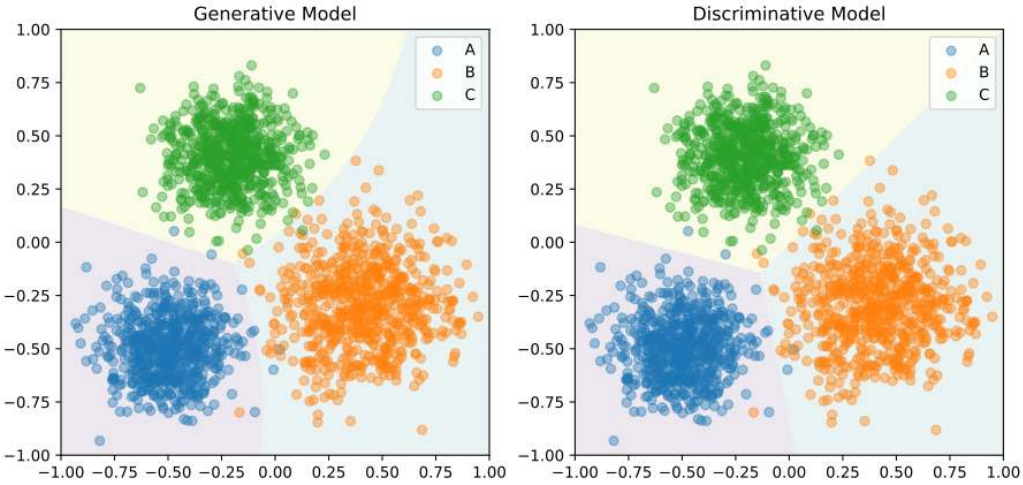
运行时会将原始数据中的 20% 作为测试数据，在训练时不使用它们。在终端界面会输出一些模型相关的内容:

```
shared testset size: 2000

# Generative Model:
    dataset size: 8000
    center: [[-0.50262024 -0.50089275]
 [ 0.39997598 -0.30294113]
 [-0.20477597  0.40510652]]
    variance: [0.02267505 0.03920124 0.02245489]
    count: [2691 2682 2627]
Accuracy: 99.60%

# Discriminative Model:
    dataset size: 6400
    testset size: 1600
    weight: [[-9.16845621 12.90910327 -3.74064706]
 [-8.36459021 -4.10747919 12.4720694 ]
 [-2.44088488  1.15345601  1.28742887]]
Accuracy: 99.50%
```

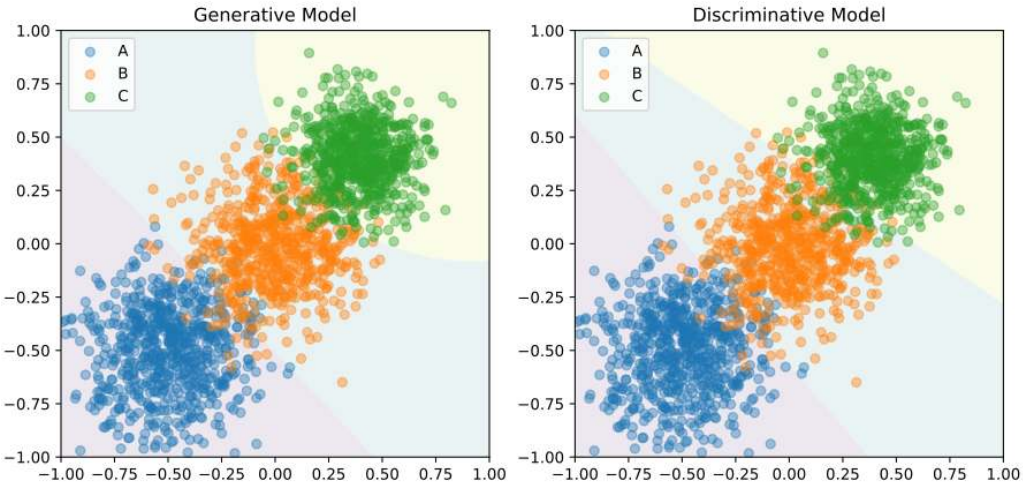
上面表明测试集大小为 2000，并且生成式模型和判别式模型在本次运行中的正确率分别为 99.60% 和 99.50%。可以看出两者差距不大。最终程序会展示两个模型的可视化结果:



这里可以看出由于不同类别的分布的方差不同的原因，生成式模型的决策边界并不是线性的。虽然二者都是包含 9 个参数的模型，但是生成式模型在这个问题上的能力会比线性的判别式模型更强一些。

3.2 “串行”式数据 让三个类别的分布中心都处在一、三象限的对角线上，此时应用两条直线作为决策边界。用下面的命令行生成数据：

```
./source.py generate -a="-0.5,-0.5,0.2,1" -b="0,0,0.2,1" -c="0.4,0.4,
0.15,1" 10000
```



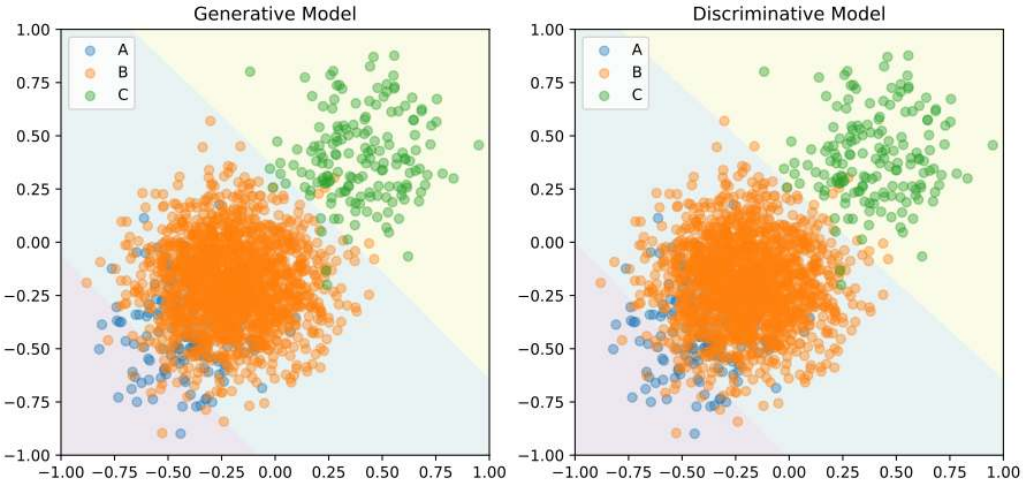
可以看到，在生成式模型中，A、B 之间的决策边界是一条直线，这是因为 A、B 两个分布的方差是一样的。在这次运行中，生成式模型的正确率为 94.95%，而判别式模型的正确率为 93.85%，这有可能是生成式模型中 B、C 间弯曲的决策边界的优势。

3.3 大面积重叠的数据 将 A、B 两个数据的分布中心贴近，并且增加 B 数据在数据集中的比重（A:B:C 为 1:10:1）：

```
./source.py generate -a="-0.4,-0.4,0.2,1" -b="-0.2,-0.2,0.2,10" -c="0.
4,0.4,0.2,1" 10000
```

这种时候判别式模型中的梯度下降速度减慢，默认的学习参数没有得到正确的结果，需要调整一下学习参数：

```
./source.py run a.data -tr=0.99 -d=1e-3
```



此时可以看到由于 B 的比重过大，导致 A、B 之间的分类实际上没有任何意义。而另一方面，B、C 之间的决策边界基本位于二者分布中心的中垂线上，并没有因为 B 的比重而受到影响，比较符合分类的要求。

4. 总结

生成式模型比较复杂，但是最后可以得到最优参数的封闭解，因此学习起来非常简单，比较可靠。判别式模型比较简单，计算起来也非常高效、方便，但是需要使用优化算法来进行学习。在上面的测试中出现了需要手动调整学习参数的情况。学习过程没有生成式模型那么高效。