机器学习讲义(L17-B)：生成式贝叶斯分类器与线性判别分析

授课教师：王贝伦 / 助教：张嘉琦，黄旭，谈笑，徐浩卿

1 高斯朴素贝叶斯分类器

注意到我们在上一节所讲的朴素贝叶斯分类器是针对离散型数据的，我们在这一节介绍一种针对连续型数据的生成式分类器。在朴素贝叶斯的假设下，我们进一步假设在同一类中，各特征服从某些均值、方差未知的高斯分布。既然各特征相互独立，那么同一分类的样本向量就服从一个均值未知，协方差矩阵为对角阵的多维正态分布。即，对标签为 C_i 的样本 $(x_1, x_2, \cdots, x_p)|C_i$

$$x_j|C_i \sim \mathcal{N}(\mu_{ij}, \sigma_{ij}^2) \Rightarrow (x_1, x_2, \cdots, x_p)|C_i \sim \mathcal{N}(\mu_i, \Sigma_i) \tag{1}$$

在这种假设下，训练分类器也十分简单，只需要按照最大似然函数的结论，在同一个类的样本点中，对每个特征分量分别计算样本均值 $\hat{\mu}_{ij}$、样本方差 $\hat{\sigma}……2_{ij}$，即得到高斯朴素贝叶斯分类器对该类的估计：

$$\hat{x}_j|C_i \sim \mathcal{N}(\hat{\mu}_{ij}, \hat{\sigma}_{ij}^2) \tag{2}$$

其中 $\hat{\mu}_{ij} = \dfrac{1}{n_i}\sum_{k=1}^{n_i} x_{kj}$，$\hat{\sigma}_{ij}^2 = \dfrac{1}{n_i - 1}\sum_{k=1}^{n_i}(x_{kj} - \hat{\mu}_{ij})^2$，$x_{kj}$ 是训练集中标签为 C_i 的样本点第 j 个特征值，n_i 为训练集中标签为 C_i 的样本数量。或者写成

$$(\hat{x}_1, \hat{x}_2, \cdots, \hat{x}_p)|C_i \sim \mathcal{N}(\hat{\mu}_i, \hat{\Sigma}_i) \tag{3}$$

其中 $\hat{\mu}_i = (\hat{\mu}_{i,1}, \hat{\mu}_{i,2}, \cdots, \hat{\mu}_{i,p})$，$\hat{\Sigma}_i = \mathrm{diag}(\hat{\sigma}_{i,1}^2, \hat{\sigma}_{i,2}^2, \cdots, \hat{\sigma}_{i,p}^2)$。这里 $\mathrm{diag}(\hat{\sigma}_{i,1}^2, \hat{\sigma}_{i,2}^2, \cdots, \hat{\sigma}_{i,p}^2)$ 表示构建一个对角矩阵且对角线上的元素依次为 $\hat{\sigma}_{i,1}^2, \hat{\sigma}_{i,2}^2, \cdots, \hat{\sigma}_{i,p}^2$。与朴素贝叶斯分类器类似，对于先验概率 $P(C = C_i)$，我们直接将类别频率作为估计值

$$P(C = C_i) = \frac{n_i}{n} \tag{4}$$

其中 n 为训练集总样本数。

在计算出 μ_{ij}, σ_{ij}^2 后，对于测试集中的新样本点 x_m，我们对每个类别 C_i 分别计算

$$\begin{aligned} P(C_i|x_m) &= P(C = C_i)\hat{f}(x_m|C_i) \\ &= \frac{n_i}{n}\frac{1}{(2\pi)^{p/2}|\hat{\Sigma}_i|^{1/2}}\exp\left(-\frac{1}{2}(x_m - \hat{\mu}_i)^\top \hat{\Sigma}_i^{-1}(x_m - \hat{\mu}_i)\right) \end{aligned} \tag{5}$$

然后选择一个使得 $P(C_i|x_m)$ 最大的 C_i，表示高斯朴素贝叶斯分类器对该测试样本点的标签预测结果。

1.1 编程实现

我们这里用 Python 实现一个简单的高斯贝叶斯分类器。

```
import numpy as np

data = load_data() # 另外实现的读取数据的函数，读入的数据data应该是一个 n x p + 1 的矩阵，其中最后一列表示样本的标签
x_train, label_train, x_test, label_test = prepare_dataset(data) # 需要另外实现的将数据集划分为训练和测试数据集的函数。其中x的每行是一个样本有p个特征；label表示每个对应样本的类别

c = len(set(label_train)) # 类别数
n, p = x_train.shape # 样本数和特征维数

# Training，训练过程
mu = np.zeros([c,p]) # 样本均值
sigma = np.zeros([c,p]) # 样本方差
count = np.zeros([c,1]) # 每个类的样本数
for i in range(c): # 对每个类
  x_i = x_train[label_train==i] # 选取该类的所有样本
  mu[i] = np.mean(x_i, 0)
  sigma[i] = np.var(x_i, 0, ddof=1)
  count[i] = len(x_i)
pc = count/len(x_train) # 各类的先验概率

# Testing，测试过程
# 测试过程以一个样本x_test[0]为例
p_max = 0.0 # 最大的后验概率
label_predict = -1 # 标签预测值
for i in range(c): # 计算样本对于每一个类的后验概率
  p_i = pc[i] * np.product(sigma[i])**-0.5\
    * np.exp(-0.5*(x_test[0]-mu[i]).dot(
    (x_test[0]-mu[i])/sigma[i]))
  if p_i > p_max:
    p_max = p_i
    label_predict = i
print('The label prediction is', label_predict) # 预测结果
```

2 非朴素高斯贝叶斯分类器

我们之前介绍的都是基于朴素贝叶斯假设的分类器。这些分类器存在的问题是特征的条件独立性在实际应用中很难满足。虽然这些朴素贝叶斯模型在一些应用上的表现并不差，但是对于数据比较复杂的应用，我们希望能提高生成型贝叶斯分类器的准确率，因此提出了非朴素贝叶斯模型。非朴素贝叶斯模型并不假设特征 $X_1, \ldots, X_p$ 之间两两相互条件独立，而是假设同一类的样本服从多元高斯分布，不同类别的样本服从不同的高斯分布，那么有

$$\begin{aligned} P(X_1, X_2, \ldots, X_p|C) &= \mathcal{N}(x|\mu, \Sigma) \\ &= \frac{1}{(2\pi)^{p/2}}\frac{1}{|\Sigma|^{1/2}}\exp\Big(-\frac{1}{2}(x - \mu)^T\Sigma^{-1}(x - \mu)\Big) \end{aligned} \tag{6}$$

其中 p 为特征的个数，μ 为该类别样本的均值，Σ 为该类别样本的协方差矩阵。μ 和 Σ 可以通过最大似然估计来计算。

2.1 线性判别分析

线性判别分析假设每个类的样本的协方差矩阵相等，即

$$\forall k,\ \Sigma_k = \Sigma \tag{7}$$

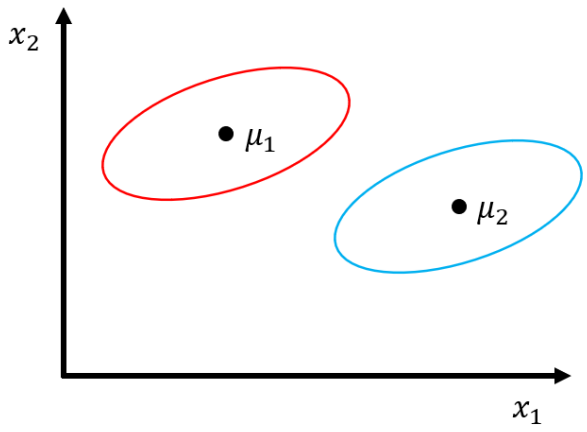

Figure 1: 线性判别分析示意图

图1所示的两个类别分别服从不同的二元高斯分布，两个类别的协方差矩阵相同，均值不同。图2所示的是两个类别的概率密度函数，这两个高斯分布也满足同样的条件。

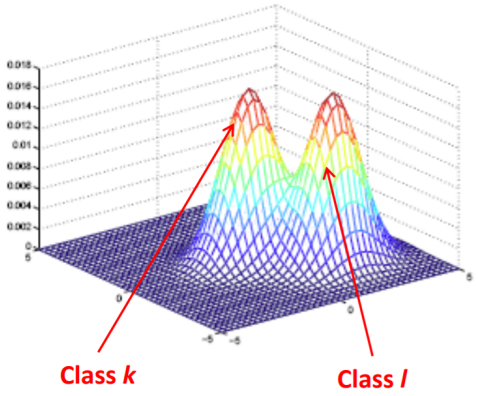

Figure 2: 二元高斯分布的概率密度函数曲面图

与之前的朴素贝叶斯分类器一样，想要预测样本 X 的类别，需要求 $\arg\max_k P(C_k|X)$，也就是目标函数。将式子转化可以得到

$$\begin{aligned}\arg\max_k P(C_k|X) &= \arg\max_k P(X, C_k)\\ &= \arg\max_k P(X|C_k)P(C_k)\\ &= \arg\max_k \log(P(X|C_k)P(C_k))\\ &= \arg\max_k \log P(X|C_k) + \log P(C_k)\end{aligned} \tag{8}$$

接下来我们先分析一下两个类别 k 和 l 的决策边界上的点，决策边界即两个类别的临界面，这样的点满足

$$P(C_k|X) = P(C_l|X) \tag{9}$$

可以得到

$$\log\frac{P(C_k|X)}{P(C_l|X)} = 0 \tag{10}$$

令 $P(C_k) = \pi_k$，$P(C_l) = \pi_l$，该式子实际上表示决策边界上的点连成的曲线方程，继续化简该式子

$$\begin{aligned}\log\frac{P(C_k|X)}{P(C_l|X)} &= \log\frac{P(X|C_k)}{P(X|C_l)} + \log\frac{\pi_k}{\pi_l}\\ &= \log P(X|C_k) - \log P(X|C_l) + \log\frac{\pi_k}{\pi_l}\end{aligned} \tag{11}$$

已知 $P(X|C_k)$ 和 $P(X|C_l)$ 服从多元高斯分布，将其概率密度函数代入上式，化简可以得到

$$\begin{aligned}\log\frac{P(C_k|X)}{P(C_l|X)} &= \log P(X|C_k) - \log P(X|C_l) + \log\frac{\pi_k}{\pi_l}\\ &= \log\frac{\pi_k}{\pi_l} - \frac{1}{2}(\mu_k+\mu_l)^T\Sigma^{-1}(\mu_k-\mu_l)\\ &\quad + x^T\Sigma^{-1}(\mu_k-\mu_l)\\ &= 0\end{aligned} \tag{12}$$

令 $\log\frac{\pi_k}{\pi_l} - \frac{1}{2}(\mu_k+\mu_l)^T\Sigma^{-1}(\mu_k-\mu_l) = b$，$\Sigma^{-1}(\mu_k-\mu_l) = a$，那么决策边界的曲线方程的最终形式变为

$$x^T a + b = 0 \tag{13}$$

不难看出，两个类别的决策边界是一个线性方程。最后回到我们的目标函数，将 $P(X|C_k)$ 代入式8，可以得到

$$\begin{aligned}&\arg\max_k P(C_k|X) = \arg\max_k P(X|C_k)P(C_k)\\ &= \arg\max_k[-\log((2\pi)^{p/2}|\Sigma|^{1/2})\\ &\quad -\frac{1}{2}(x-\mu_k)^T\Sigma^{-1}(x-\mu) + \log(\pi_k)]\\ &= \arg\max_k[-\frac{1}{2}(x-\mu_k)^T\Sigma^{-1}(x-\mu) + \log(\pi_k)]\end{aligned} \tag{14}$$

也被称为线性判别函数。

2.2 二次判别分析

二次判别分析不需要线性判别分析的假设成立，即每个类别样本的协方差矩阵并不相同，设类别 k 的协方差矩阵为 Σ_k。类似的，二次判别函数为

$$\delta_k(x) = -\frac{1}{2}\log|\Sigma_k| - \frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k) + \log\pi_k \tag{15}$$

那么分类规则为

$$\hat{G}(x) = \arg\max_k \delta_k(x) \tag{16}$$

与线性判别分析类似地分析边界方程，可以得到它的边界为 x 的二次函数。二次判别分析可以更好的拟合数据，但是它需要预估的参数比线性判别分析的更多。

正则判别分析是介于线性判别分析和二次判别分析之间的模型。它定义了正则协方差矩阵

$$\Sigma_k(\alpha) = \alpha\Sigma_k + (1-\alpha)\Sigma \tag{17}$$

那么参数 α 决定了每个类别协方差矩阵的收缩程度。二次判别函数 $\delta_k(x)$ 中的协方差矩阵就用正则协方差矩阵 $\Sigma_k(\alpha)$ 代替。参数 α 同时也控制着模型的复杂度。

透过两种判别分析，我们再回过来看一下朴素贝叶斯分类器的边界。因为朴素贝叶斯方法并没有假设每个类的协方差矩阵相同，所以一般朴素贝叶斯的边界也是二次的。

3 判别模型 vs 生成模型

对比生成模型与判别模型，生成模型是通过 $P(X|C=c_k)$ 和 $P(C=c_k)$ 计算出联合概率分布 $P(X,C)$，然后根据样本和哪个类别的联合概率最高即分为哪类，比如线性判别分析。而判别模型直接计算条件概率 $P(C=c_k|X)$，可以直接通过 X 来得到所属的类别，比如逻辑回归。对比线性判别分析和逻辑回归：
线性判别分析的特点

- 假设同一类的样本服从高斯分布，且每个类的协方差矩阵相同

- 模型参数通过最大化全对数似然函数估计，每个类的参数独立估计，共有 $Kp + \frac{p(p+1)}{2} + (K-1)$ 个参数
- 使用了边缘概率 $P(x)$
- 易于训练，方差较小，如果模型正确的话更高效
- 渐进误差更大，但是收敛的更快

逻辑回归的特点

- 假设同一类的样本服从同一个指数分布
- 模型参数通过最大化条件对数似然函数估计，同时考虑所有其他类，共有 $(K-1)(p+1)$ 个参数
- 没有使用边缘概率 $P(x)$
- 更难训练，但是对数据生成的过程更具鲁棒性
- 渐进误差较小，但是收敛的更慢

总的来说，从实验中来看，生成模型收敛更快，比较适用于小训练集，可以很好的处理丢失的数据，而判别模型有更小的渐进误差，比较适用于大训练集。

引用

[1] Naïve Bayes and Gaussian Bayes Classifier : https://www.cs.toronto.edu/~urtasun/courses/CSC411/tutorial4.pdf

[2] Gaussian Bayes Classifier https://www.cs.cmu.edu/~./awm/tutorials/gaussbc.html

[3] https://www.zhihu.com/question/20446337