# Abstract

本期我们学习线性分类-软输出-概率生成模型的一种算法：高斯判别分析($GDA$)。

# Idea

在前一期我们学习的逻辑回归算法属于概率判别模型，判别模型与生成模型的区别是：

- 判别模型是直接对概率 $p(y|x)$ 进行建模，求出其真实的概率值
- 生成模型是则是对 $p(y|x)$ 使用贝叶斯定理，转化为 $\frac{p(x|y)p(y)}{p(x)}$，因为 $p(x)$ 与 $y$ 无关，因此可以忽略，最终得到：

$$p(y|x) \propto p(x|y)p(y) = p(x;y) \tag{15}$$

因此我们关注的是 $(x, y)$ 这个联合分布，最后预测时只需比较 $p(y=0|x), p(y=1|x)$ 哪个大即可。

# Algorithm

首先，我们对模型做出一些假设：

$$y \in \{0,1\} \quad y \sim Bernuolli(\phi) \quad p(y) = \phi^y(1-\phi)^{1-y}$$

$$\begin{cases} x|y=1 & \sim & N(\mu_1, \Sigma) \\ x|y=0 & \sim & N(\mu_2, \Sigma) \end{cases} \Longrightarrow p(x|y) = N(\mu_1, \Sigma)^y N(\mu_2, \Sigma)^{1-y}$$

因此模型的所有参数 $\theta$ 为：

$$\theta = (\phi, \mu_1, \mu_2, \Sigma) \tag{16}$$

现在给出模型的损失函数：

$$\begin{aligned} J(\theta) = log(p(Y|X)) &= log(\prod_{i=1}^{n} p(y_i|x_i)) \\ &= \sum_{i=1}^{n} log(p(y_i|x_i)) \end{aligned} \tag{17}$$

因此：

$$\begin{aligned} \hat{\theta} = argmax(J(\theta)) &= argmax(\sum_{i=1}^{n} log(\frac{p(x_i|y_i)p(y_i)}{p(x_i)})) \\ &= argmax(\sum_{i=1}^{n} log(p(x_i|y_i)p(y_i))) \\ &= argmax(\sum_{i=1}^{n} y_i\, log(N(\mu_1, \Sigma)) + (1-y_i)\, log(N(\mu_2, \Sigma)) + log(\phi^{y_i}(1-\phi)^{1-y_i})) \end{aligned} \tag{18}$$

## $\phi$的求解

对 $\phi$ 求偏导：

$$\sum_{i=1}^{N} \frac{y_i}{\phi} + \frac{y_i - 1}{1-\phi} = 0 \Longrightarrow \phi = \frac{\sum_{i=1}^{N} y_i}{N} = \frac{N_1}{N} \tag{19}$$

其中，$N, N_1, N_2$ 分别为总样本的个数，正例与反例的个数

## $\mu$ 的求解

然后对 $\mu_1$ 进行求解：

$$
\begin{aligned}
\hat{\mu_1} &= \underset{\mu_1}{argmax} \sum_{i=1}^{N} y_i \log N\left(\mu_1, \Sigma\right) \\
&= \underset{\mu_1}{argmax} \sum_{i=1}^{N} y_i \log\left(\frac{1}{(2\pi)^{\frac{p}{2}}|\Sigma|^{\frac{1}{2}}} exp(-\frac{1}{2}(x_i-\mu_1\right)^T (\Sigma)^{-1}(x_i-\mu_1))) \\
&= \underset{\mu_1}{argmin} \sum_{i=1}^{N} y_i (x_i-\mu_1)^T \Sigma^{-1}\left(x_i-\mu_1\right)
\end{aligned} \tag{20}
$$

上述推导中用到了多元高斯分布的概率密度函数：

$$
p(x)=\frac{1}{(2\pi)^{\frac{p}{2}}|\Sigma|^{\frac{1}{2}}} exp(-\frac{1}{2}(x_i-\mu_1)^T(\Sigma)^{-1}(x_i-\mu_1)) \tag{21}
$$

其中，$p$ 为随机变量的个数，读者可以根据一元高斯分布的概率密度函数进行连乘，并辅以线代的知识，就可以推出多元的公式。

下面对式子进行微分：

$$
\begin{aligned}
\frac{\partial \Delta}{\partial \mu_1} = \sum_{i=1}^{N} -2y_i(\Sigma)^{-1}(x_i-\mu_1)=0 \\
\Longrightarrow \mu_1 = \frac{\sum_{i=1}^{N} y_i x_i}{\sum_{i=1}^{N} y_i} = \frac{\sum_{i=1}^{N} y_i x_i}{N_1}
\end{aligned} \tag{22}
$$

而由于正例与反例是对称的，因此：

$$
\mu_2 = \frac{\sum_{i=1}^{N}(1-y_i)x_i}{N_2} \tag{23}
$$

## $\Sigma$ 的求解

我们观察式子的前两项：

$$
\hat{\theta} = argmax(\sum_{i=1}^{n} y_i\ log(N(\mu_1,\Sigma)) + (1-y_i)\ log(N(\mu_2,\Sigma)) + log(\phi^{y_i}(1-\phi)^{1-y_i})) \tag{24}
$$

发现，当$y=0$时，第一项都为0；当$y=1$时，第二项都为0。

因此式子可以更为：

$$
\begin{aligned}
\hat{\theta} &= argmax(\sum_{(x_i,y_i)\in C_1} log(N(\mu_1,\Sigma)) + \sum_{(x_i,y_i)\in C_2} log(N(\mu_2,\Sigma))) \\
&= argmax(\sum_{(x_i,y_i)\in C_1} -\frac{1}{2}\log|\Sigma| - \frac{1}{2}(x_i-\mu_1)^T(\Sigma)^{-1}(x_i-\mu_1)\ + \sum_{(x_i,y_i)\in C_2} -\frac{1}{2}|\Sigma| - \frac{1}{2}(x_i-\mu_2)^T(\Sigma)^{-1}(x_i-\mu_2))
\end{aligned} \tag{25}
$$

我们观察 $(x_i-\mu)^T(\Sigma)^{-1}(x_i-\mu)$ 的形状：$(1,p)*(p,p)*(p,1)=(1,1)$，因此可以对它加上迹(tr)的符号，将其看作一个矩阵，而在迹的内部，矩阵的顺序是可以随意交换的：

$$
\begin{aligned}
\hat{\theta} &= argmax(-\frac{N}{2}\log|\Sigma| - \frac{1}{2}tr(\sum_{(x_i,y_i)\in C_1}(x_i-\mu_1)^T(\Sigma)^{-1}(x_i-\mu_1)) - \frac{1}{2}tr(\sum_{(x_i,y_i)\in C_2}(x_i-\mu_2)^T(\Sigma)^{-1}(x_i-\mu_2))) \\
&= argmax(-\frac{N}{2}\log|\Sigma| - \frac{1}{2}tr(\sum_{(x_i,y_i)\in C_1}(x_i-\mu_1)^T(x_i-\mu_1)(\Sigma)^{-1}) - \frac{1}{2}tr(\sum_{(x_i,y_i)\in C_2}(x_i-\mu_2)^T(x_i-\mu_2)(\Sigma)^{-1})) \\
&= argmax(-\frac{N}{2}\log|\Sigma| - \frac{1}{2}tr(N_1S_1(\Sigma)^{-1}) - \frac{1}{2}tr(N_2S_2(\Sigma)^{-1}))
\end{aligned} \tag{26}
$$

其中，$S$为协方差矩阵。

下面对式子求偏导：

$$
\frac{\partial \Delta}{\partial \Sigma} = -\frac{1}{2}(N\frac{1}{|\Sigma|}|\Sigma|(\Sigma)^{-1} - N_1S_1(\Sigma)^{-2} - N_2S_2(\Sigma)^{-2}) = 0 \tag{27}
$$

因此求解出 $\hat{\Sigma}$：

$$
\begin{aligned}
& N\Sigma^{-1} - N_1 S_1^T \Sigma^{-2} - N_2 S_2^T \Sigma^{-2} = 0 \\
& \Longrightarrow \hat{\Sigma} = \frac{N_1 S_1 + N_2 S_2}{N}
\end{aligned} \tag{28}
$$

最后，当我们要预测的时候，只需比较 $p(x|y=0)p(y=0)$ 与 $p(x|y=1)p(y=1)$哪一个更大即可。

## Implement

```
import numpy as np
import os
os.chdir("../")
from models.linear_models import GDA

n1 = 1000
n_test = 100
x = np.linspace(0, 10, n1 + n_test)
w1, w2 = 0.3, 0.5
b1, b2 = 0.1, 0.2
x1 = x[:n1]
x_test = x[n1:]
v1 = x1 * w1 + b1
v2 = x1 * w2 + b2
cla_1 = np.c_[x1, v1]
cla_2 = np.c_[x1, v2]
l1 = np.ones(shape=(cla_1.shape[0], 1))
l2 = np.zeros(shape=(cla_2.shape[0], 1))
train_data = np.r_[cla_1, cla_2]
train_label = np.r_[l1, l2]

v_test = x_test * w2 + b2
data_test = np.c_[x_test, v_test]

model = GDA()
model.fit(train_data, train_label)
print(model.get_params())
print("accuary:", model.evaluate(data_test, 0))
```

```
(0.5, array([[4.54504095],
       [1.46351228]]), array([[4.54504095],
       [2.47252047]]), array([[14.96809381,  8.07251381],
       [ 8.07251381, 14.96809381]]))
accuary: 1
```