第四章

一、坐标下降法求解Lasso

1. 试根据授课内容，使用坐标下降法给出Lasso问题：$\min\limits_{\beta,\beta_0}\sum_{i=1}^{n}(y_i-\beta^T x_i-\beta_0)^2+\lambda\|\beta\|_1$ 求解的详细推导步骤。

$$L=\sum_{i=1}^{n}\Big(y_i-\sum_{j=1}^{p}x_{ij}\beta_j\Big)^2+\lambda\sum_{j=1}^{p}|\beta_j|$$

$$=\sum_{i=1}^{n}\Big(y_i-\sum_{j\neq k}^{p}x_{ij}\beta_j-x_{ik}\beta_k\Big)^2+\lambda|\beta_k|+\lambda\sum_{j\neq k}^{p}|\beta_j|$$

$$\frac{\partial L}{\partial\beta_k}=2\sum_{i=1}^{n}x_{ik}\Big(\sum_{j\neq k}^{p}x_{ij}\beta_j+x_{ik}\beta_k-y_i\Big)+\lambda\,\mathrm{sign}(\beta_k)$$

$$=2\sum_{i=1}^{n}x_{ik}\Big(\sum_{j\neq k}^{p}x_{ij}\beta_j-y_i\Big)+2\sum_{i=1}^{n}x_{ik}^2\beta_k+\lambda\,\mathrm{sign}(\beta_k)$$

令 $a_k=\sum_{i=1}^{n}x_{ik}\Big(\sum_{j\neq k}^{p}x_{ij}\beta_j-y_i\Big)$ $\quad b_k=\sum_{i=1}^{n}x_{ik}^2$

则 $\frac{\partial L}{\partial\beta_k}=2a_k+2b_k\beta_k+\lambda\,\mathrm{sign}(\beta_k)$

令 $\frac{\partial L}{\partial\beta_k}=0\ \Rightarrow\ \beta_k+\frac{1}{b_k}\Big(a_k+\frac{\lambda}{2}\mathrm{sign}(\beta_k)\Big)=0$

当 $a_k>\frac{\lambda}{2}$ 时，$\beta_k=-\frac{1}{b_k}\Big(a_k-\frac{\lambda}{2}\Big)<0$

当 $-\frac{\lambda}{2}<a_k<\frac{\lambda}{2}$ 时，$\beta_k=0$

当 $a_k<-\frac{\lambda}{2}$ 时，$\beta_k=-\frac{1}{b_k}\Big(a_k+\frac{\lambda}{2}\Big)>0$

$$\Rightarrow\beta_k=\begin{cases}-\frac{1}{b_k}\Big(a_k-\frac{\lambda}{2}\Big)<0, & a_k>\frac{\lambda}{2}\\ 0, & -\frac{\lambda}{2}<a_k<\frac{\lambda}{2}\\ -\frac{1}{b_k}\Big(a_k+\frac{\lambda}{2}\Big)>0, & a_k<-\frac{\lambda}{2}\end{cases}$$

二、根据字典学习问题学习人脸图像的稀疏表示

（1）从 orl_faces 文件中读取数据初始化样本 x 和字典 b

```
# 读取单个 pgm，转成行向量
def load_single_pgm(file_path):
    f = open(file_path, 'rb')
    # p5 格式 pgm
    f.readline()  # P5\n
    (width, height) = [int(i) for i in f.readline().split()]  # 92 112
    depth = int(f.readline())
    # 按行读取像素
    data = []
    for y in range(height):
        row = []
        for x in range(width):
            row.append(ord(f.read(1)))
        data.append(row)
    data = np.array(data)  # list->ndarray
    data = data.reshape(width * height)
    return data   # 返回一维数组/向量 1*10304


# 初始化样本 x p*n x_i p*1
def init_x():
    x_matrix = []
    for i in range(40):  # 40 个文件夹
        for j in range(10):  # 每个文件夹 10 个 pgm 文件
            f_path = "orl_faces/s{}/{}.pgm".format(i+1, j+1)
            x_ij = load_single_pgm(f_path)
            x_matrix.append(x_ij)
    x_matrix = np.array(x_matrix).T  # x 10304*400
    return x_matrix


# 初始化字典 B p*k    B 的第 i 列通过第 i 个文件夹随机初始化
def init_b():
    b = []
    for i in range(40):
        j = np.random.randint(1, 10)
        f_path = "orl_faces/s{}/{}.pgm".format(i+1, j)
        b_i = load_single_pgm(f_path)
        b.append(b_i)
    b = np.array(b).T
    return b
```

（2）固定字典 b 优化 α [坐标下降法求解 Lasso]

```
#  坐标下降法求解 Lasso
 coordinate_descent(y, X, w, lam):   # x b alpha lambda
 n, p = X.shape
 # 使用坐标下降法优化回归系数 alpha
 for k in range(p):
     b_k = sum([(X[i, k] ** 2) for i in range(n)])
     a_k = 0
     for i in range(n):
         a_k += X[i, k] * (sum([(X[i, j] * w[j]) for j in range(p) if j != k]) - y[i])
     if a_k < -lam / 2:
         w_k = -(a_k + lam / 2) / b_k
     elif a_k > lam / 2:
         w_k = -(a_k - lam / 2) / b_k
     else:
         w_k = 0
     w[k] = w_k
```

（3）固定 α ， 优化字典 b

```
# 固定 alpha，优化字典 b
 train_b(x, b, alpha, lam):
 alpha_alpha_t = np.dot(alpha, alpha.T)
 det_number = np.linalg.det(alpha_alpha_t)  # 求行列式
 # 若可逆
 if det_number:
     one = np.dot(x, alpha.T)
     two = np.linalg.inv(alpha_alpha_t)   # 矩阵求逆
     b[:, :] = np.dot(one, two)
 # 若不可逆
 else:
     m, n = alpha_alpha_t.shape
     I = np.identity(m)  # 单位矩阵
     one = np.dot(x, alpha.T)
     two = np.linalg.inv((alpha_alpha_t + lam * I))
     b[:, :] = np.dot(one, two)
```

（4）主函数调用，反复迭代上面两步

```
if __name__ == '__main__':
 lam = 1
 epochs = 10
 # 初始化 x 和字典 b
```

```python
x = load_data.init_x()  # p*n
b = load_data.init_b()  # p*k
alpha = np.random.randn(b.shape[1], x.shape[1])  # k*n 随机标准正态分布初始化
# 开始训练
p, n = x.shape
for epoch in range(epochs):
    print("第{}次迭代: ".format(epoch+1))
    # 先固定固定字典 b,优化 alpha(alpha_1,alpha_2,....) k*n,alpha_i(k*1)为 x_i(p*1)的稀疏表示
    # 坐标下降求解 Lasso
    for i in range(n):
        print("优化 alpha 的第{}列".format(i+1))
        coordinate_descent(x[:, i], b, alpha[:, i], lam)
    # 固定 alpha，优化字典 b
    train_b(x, b, alpha, lam)
# 训练结束
print("训练结束.")
print(b)
print(alpha)
```