1、证明题

坐标下降法求解 Lasso

$$L = \cancel{\min} \sum_{i=1}^{n}\left(y_i - \sum_{j=1}^{p} x_{i,j}\beta_j\right)^2 + \lambda \sum_{j=1}^{p}|\beta_j|$$

$$= \sum_{i=1}^{n}\left(y_i - \sum_{j\neq k} x_{i,j}\beta_j - x_{i,k}\beta_k\right)^2 + \lambda\sum_{j\neq k}|\beta_j| + \lambda|\beta_k|$$

令 $C_1 = y_i - \sum_{j\neq k} x_{i,j}\beta_j \qquad C_2 = \lambda\sum_{j\neq k}|\beta_j|$

$$\therefore L = \sum_{i=1}^{n}\left(C_1 - x_{i,k}\beta_k\right)^2 + C_2 + \lambda|\beta_k|$$

$$= \sum_{i=1}^{n}\left(C_1^2 - 2C_1 x_{i,k}\beta_k + x_{i,k}^2\beta_k^2\right) + C_2 + \lambda|\beta_k|$$

$$\therefore \frac{\partial L}{\partial \beta_k} = \sum_{i=1}^{n}\left(-2C_1 x_{i,k} + 2x_{i,k}^2\beta_k\right) + \lambda\, sign(\beta_k)$$

$$= 2\beta_k\sum_{i=1}^{n} x_{i,k}^2 - 2\sum_{i=1}^{n} C_1 x_{i,k} + \lambda\, sign(\beta_k)$$

令 $a_k = -\sum_{i=1}^{n} C_1 x_{i,k}$

$$= -\sum_{i=1}^{n} x_{i,k}\left(y_i - \sum_{j\neq k} x_{i,j}\beta_j\right)$$

$$= \sum_{i=1}^{n} x_{i,k}\left(\sum_{j\neq k} x_{i,j}\beta_j - y_i\right)$$

$b_k = \sum_{i=1}^{n} x_{i,k}^2$

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

令 $\frac{\partial L}{\partial \beta_k} = 0$ 得

$$\beta_k = -\frac{1}{b_k}\left(a_k + \frac{\lambda}{2} sign(\beta_k)\right)$$

(1) 当 $a_k > \frac{\lambda}{2}$ 时 ∵ $sign(\beta_k) > -\frac{\lambda}{2}$

∴ $\beta_k < 0$ ∴ $sign(\beta_k) = -1$

∴ $\beta_k = -\frac{1}{b_k}(a_k - \frac{\lambda}{2})$

(2) 当 $a_k < -\frac{\lambda}{2}$ 时 ∵ $sign(\beta_k) < \frac{\lambda}{2}$

∴ $\beta_k > 0$ ∴ $sign(\beta_k) = 1$

∴ $\beta_k = -\frac{1}{b_k}(a_k + \frac{\lambda}{2})$

(3) 当 $-\frac{\lambda}{2} < a_k < \frac{\lambda}{2}$ 时

① 设 $\beta_k < 0$ 即 $sign(\beta_k) = -1$

∴ $\beta_k = -\frac{1}{b_k}(a_k - \frac{\lambda}{2}) > 0$ 矛盾

② 设 $\beta_k > 0$ 即 $sign(\beta_k) = 1$

∴ $\beta_k = -\frac{1}{b_k}(a_k + \frac{\lambda}{2}) < 0$ 矛盾

因此 $\beta_k = 0$

综上所述

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

2、上机实验

（1）从文件中读取数据，得到数据集 X 和初始化的字典 D

```python
def read_pgm(filename):
    """..."""
    f = open(filename, 'rb')
    f.readline()  # P5\n
    (width, height) = [int(i) for i in f.readline().split()]
    depth = int(f.readline())
    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)
    data = data.reshape(width * height)
    return data
def get_data():
    """得到数据集X，其中X的每列为一个样本，  每一行为一个特征，X为p*n的矩阵"""
    X = []
    for i in range(1, 41):
        for j in range(1, 11):
            fn = "../orl_faces/s{}/{}.pgm".format(i, j)
            data = read_pgm(fn)
            X.append(data)
    X = np.array(X)
    X = X.T
    X_mean = np.mean(X, axis=0)
    X_std = np.std(X, axis=0)
    X = (X - X_mean) / X_std
    return X

def get_init_D():
    """得到初始的字典D，D为p*k维的矩阵"""
    D = []
    for i in range(1, 41):
        j = np.random.randint(1, 10)
        fn = "../orl_faces/s{}/{}.pgm".format(i, j)
        data = read_pgm(fn)
        D.append(data)
    D = np.array(D)
    D = D.T
    D_mean = np.mean(D, axis=0)
    D_std = np.std(D, axis=0)
    D = (D - D_mean) / D_std
    return D
```

（2）坐标下降法完成字典学习的第一步

```python
def coordinate_descent(y, X, w, lamda=1):
    n, p = X.shape
    loss = loss_function(y, X, w)
    # 使用坐标下降法优化回归系数alpha
    for it in range(10):
        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 < -lamda / 2:
                w_k = -(a_k + lamda / 2) / b_k
            elif a_k > lamda / 2:
                w_k = -(a_k - lamda / 2) / b_k
            else:
                w_k = 0
            w[k] = w_k
        loss_prime = loss_function(y, X, w)
        print("loss:", loss_prime)
        delta = abs(loss_prime - loss)
        loss = loss_prime
        if delta < 0.1:
            break
```

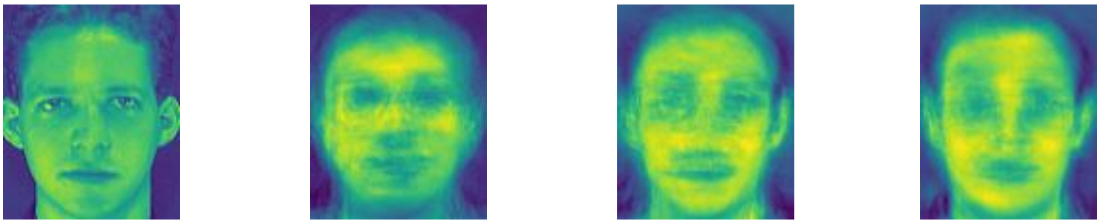
（3）更新字典

```python
def update_D(X, D, A, lamda=1):
    """更新字典D"""
    det_number = np.linalg.det(np.dot(A, A.T))
    if det_number:
        one = np.dot(X, A.T)
        two = np.linalg.inv(np.dot(A, A.T))
        D[:, :] = np.dot(one, two)
    else:
        m, n = np.dot(A, A.T).shape
        I = np.identity(m)
        one = np.dot(X, A.T)
        two = np.linalg.inv(np.dot(A, A.T) + lamda * I)
        D[:, :] = np.dot(one, two)
```

（4）交替优化

```python
def sparse_learning(X, D, A, lamda=1, max_iter=10):
    """交替迭代求解D和A"""
    p, n = X.shape
    for iter in range(max_iter):
        print("第{}次迭代".format(iter))
        for i in range(n):
            print("优化A的第{}列".format(i))
            coordinate_descent(X[:, i], D, A[:, i], lamda=lamda)
        update_D(X, D, A, lamda=lamda)
```

（5）重构图片对比



图一为原图；图二是在进行一轮交替优化，每轮当中的坐标下降只进行一次的结果，运行了 1 个小时；图三是在进行两轮交替优化，每轮当中的坐标下降进行三次的结果，运行了 6 个小时；图四是在进行两轮交替优化，每轮当中的坐标下降进行五次的结果，运行了 10 个小时。可以看出，图片的重构效果越来越好，但由于运行时间过长，没有进行更多轮的优化。

（5）损失函数比较

```
第0次迭代
优化A的第0列
loss: 97095.71634037342
loss: 57557.78801896795
loss: 36831.47520747347
loss: 25914.61228947116
loss: 19730.458354126597
loss: 15870.086936797798
loss: 13260.45059051608
loss: 11390.890580105315
loss: 9991.776877044653
loss: 8907.333520293108
优化A的第1列
```

在进行一轮优化时，坐标下降进行 10 次，可以看出损失函数还是在下降，因此可以知道，进行更多次优化后，可以得到更好的结果。

```
Run:    comparation
最初的损失函数为： 626324097.8756695
循环1*1次的损失函数为： 512108.7371025787
循环2*3次的损失函数为： 47127.504425275285
循环2*5次的损失函数为： 33295.712006840826

Process finished with exit code 0
```

比较损失函数的结果，可以看出，在进行更多次优化后，损失得到了明显的下降。