CSE 546 HW #3 Problem 5 Revisited

Sam Kowash

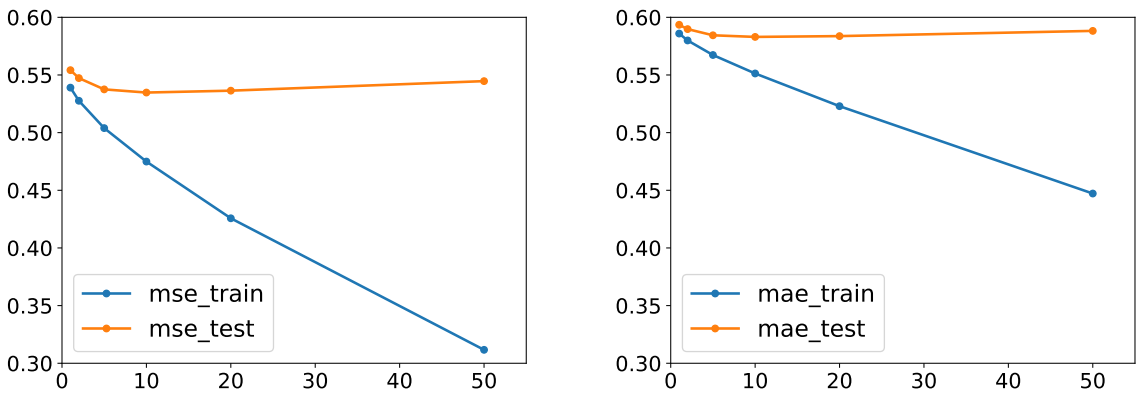
December 12, 2018

1 Movie Recommender System

5. Based on a partial data set of $n = 1000$ users rating in $[-10, 10]$ a subset of $m = 500$ movies, we estimate those users' ratings of unrated movies. We do this by developing d-dimensional vector models representing each user by a vector u_i and each movie by a vector v_j, then estimating user i's rating of movie j by $\langle u_i, v_j \rangle$.

   (a) First, we try the most naive approach and simply complete each empty entry in each column j with the average user's rating of movie j. This achieves a mean squared training error of 0.559 and a mean absolute training error of 0.596. The squared and absolute *test* errors (from users' additional held-out ratings) were 0.570 and 0.602, respectively.

   (b) Next, we suppose all omitted ratings are zero and generate vector representations with $\{u_i\}$ and $\{v_j\}$ as the first d left- and right-singular vectors of the data matrix, respectively (with the singular value matrix multiplied into V^T). Squared and absolute train and test errors shown in Fig. 1.1.



(a) MSEs for SVD representations at different d (b) MAEs for SVD representations at different d

Figure 1.1

   (c) Lastly, we try to improve on these results by minimizing an objective which only accumulates error over extant ratings, rather than substituting values.

$$L\left(\{u_i\}, \{v_j\}\right) = \sum_{(i,j) \in T} \left(\langle u_i, v_j \rangle - R_{ij}\right)^2 + \lambda \sum_{i=1}^{n} \|u_i\|_2^2 + \lambda \sum_{j=1}^{m} \|v_j\|_2^2$$

   Here λ is a regularization hyperparameter to constrain the vector magnitudes. We can approach this through alternating minimization, where if we fix $\{v_i\}$ we can see that each

u_i should become

$$\hat{u}_i = \left(\sum_j v_j v_j^T + \lambda I \right)^{-1} \left(\sum_j R_{ij} v_j \right),$$

and each v_j should be updated by an exactly symmetric expression. Unfortunately, code still has bugs (dimension mismatch I haven't found yet) and I have no result for this part.

```
#!/usr/bin/env python
import numpy as np
import scipy.sparse as sp
from scipy.sparse.linalg import svds
import sys


def parse_raw(raw):
    n_users = int(raw[-1,0])
    n_movies = 500

    users = raw[:,0].astype(int)-1
    movies = raw[:,1].astype(int)-1

    result = sp.coo_matrix((raw[:,2], (users, movies))).tocsc()
    means = np.array(result.sum(axis=0)/result.getnnz(axis=0)).flatten()
    demeaned = sp.coo_matrix((raw[:,2]-means[movies], (users,movies)))


    return result, demeaned.tocsc(), means


def mse(R_pred,test_raw):
    n_data = len(test_raw)
    users = test_raw[:,0].astype(int)-1
    movies = test_raw[:,1].astype(int)-1
    errs = R_pred[users,movies] - test_raw[:,2]

    return np.sum(errs**2)/n_data

def mae(R_pred,test,test_raw):
    n_users,n_movies = test.shape
    n_ratings = test.getnnz(axis=1)

    users = test_raw[:,0].astype(int)-1
    movies = test_raw[:,1].astype(int)-1

    err = 0
    for i in range(n_users):
        ind = np.where(users == i)
        err_i = R_pred[users[ind],movies[ind]] - test_raw[ind,2]
        err += np.sum(np.abs(err_i))/n_ratings[i]

    return err/n_users


def train_dumb(train):
    n_users,n_movies = train.shape
    return (np.ones((n_users,1)),
            np.array(train.sum(axis=0)/train.getnnz(axis=0)))


def train_svd(train,d):
    U,S,Vt = svds(train,k=d)
    return U, sp.diags(S,0).dot(Vt)

def train_clever(train,train_raw,d,lam,thresh=1e-3):
    train_r = train.tocsr()
    n_users,n_movies = train.shape
```

```
    users = train_raw[:,0].astype(int)-1
    movies = train_raw[:,1].astype(int)-1

    u_old = np.random.randn(n_users, d)
    v_old = np.random.randn(n_movies,d)
    u = np.zeros_like(u_old)
    v = np.zeros_like(v_old)
    delta_u = 1000*np.ones_like(u_old)
    delta_v = 1000*np.ones_like(v_old)
    i = 1

    while (np.max(delta_u) > thresh) or (np.max(delta_v) > thresh):
        print(f"On step {i}")
        u,v = u_old, v_old

        for user in users:
            rhs = train_r[user].dot(v)
            lh_mat = v.T @ v + lam*np.eye(d)

            u[user] = np.linalg.solve(lh_mat, rhs)

        print("finished u")

        for movie in movies:
            rhs = (train[:,movie].T).dot(u)
            lh_mat = u.T @ u + lam*np.eye(d)

            v[movie] = np.linalg.solve(lh_mat, rhs)

        print("finished v")

        delta_u = np.abs(u-u_old)
        delta_v = np.abs(v-v_old)
        print(np.max(delta_u),np.max(delta_v))
        i += 1

    return u, v.T

def pred(U,Vt,means):
    n = U.shape[0]
    return U.dot(Vt) + np.outer(np.ones(n), means)



train_raw = np.genfromtxt('data/train.txt',delimiter=',')
test_raw = np.genfromtxt('data/test.txt',delimiter=',')

train, d_train, means_train = parse_raw(train_raw)
test, d_test, means_test = parse_raw(test_raw)

if len(sys.argv) < 2:
    print("Not enough args.")
    sys.exit()

if sys.argv[1] == 'svd':
    ds = np.array([1, 2, 5, 10, 20, 50])
    mses_train = []
    mses_test = []
    maes_train = []
```

```
    maes_test = []

    for d in ds:
        print(f"On d={d}")
        U,Vt = train_svd(d_train,d)
        R_pred = pred(U,Vt,means_train)


        mses_test.append(mse(R_pred,test_raw))
        maes_test.append(mae(R_pred,test,test_raw))
        mses_train.append(mse(R_pred,train_raw))
        maes_train.append(mae(R_pred,train,train_raw))

    mses_train = np.array(mses_train)
    maes_train = np.array(maes_train)
    mses_test = np.array(mses_test)
    maes_test = np.array(maes_test)
    np.savez('data/svd.npz',d=ds,mse_train=mses_train,mae_train=maes_train,
             mse_test=mses_test,mae_test=maes_test)



elif sys.argv[1] == 'clever':
    ds = np.array([1,2,5,10,20,50])
    mses_train = []
    mses_test = []
    maes_train = []
    maes_test = []

    for d in ds:
        print(f"On d={d}")
        U,Vt = train_clever(d_train,train_raw,d,1e-4)
        R_pred = pred(U,Vt,means_train)


        mses_test.append(mse(R_pred,test_raw))
        maes_test.append(mae(R_pred,test,test_raw))
        mses_train.append(mse(R_pred,train_raw))
        maes_train.append(mae(R_pred,train,train_raw))

    mses_train = np.array(mses_train)
    maes_train = np.array(maes_train)
    mses_test = np.array(mses_test)
    maes_test = np.array(maes_test)
    np.savez('data/clever.npz',d=ds,mse_train=mses_train,mae_train=maes_train,
             mse_test=mses_test,mae_test=maes_test)


elif sys.argv[1] == 'dumb':
    U,Vt = train_dumb(d_train)
    R_pred = pred(U,Vt,means_train)

    print(mse(R_pred,train_raw))
    print(mae(R_pred,train,train_raw))
    print(mse(R_pred,test_raw))
    print(mae(R_pred,test,test_raw))
```