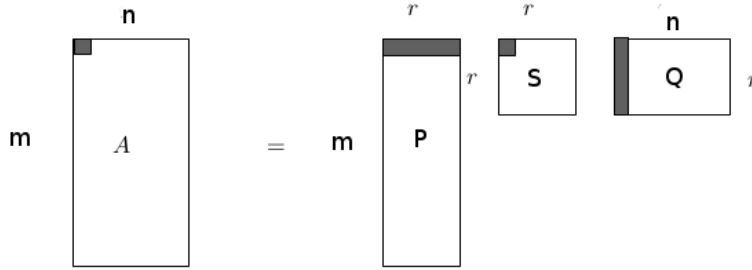


Yaklasiksal SVD ile Tavsiye Sistemleri

SVD, *Toplu Tavsiye* yazisinda Movielens verisine SVD uygulayarak once boyut azaltmistik. Azaltilmis boyut uzerinden, yeni bir kullanicinin diger mevcut kullanicilara mesafesini hesaplamis, ve Boylece en cok benzedigi diger kullaniciyi bulmustuk. Bu kullanicinin bir film icin verdigi notu yeni kullanıcı için tahmin olarak baz almistik.

SVD uygulamanin degisik bir yolu daha var. Netflix yarismasinda kullanılan [1] bir yaklasim soyle. Altteki SVD ayristirmasina bakalim,



1. kullanıcıni 1. filme verdigi not ustte koyu gosterilen satirlarin carpimi ile oluyor, eger ufak harfler ve kullanıcı (user) için u, film için i indisini kullanırsak, ve q, p vektorlerini Q, P matrislerinin sirasiyla kolon ve satirlarini gostermek için kullanırsak, ayristirma sonrasi begeni degerinin onemli bir kismi $q_i^T p_u$ carpimindadir. Carpim icinde S'ten gelecek tekil degeri (singular value) ne olacak? Formulasyonu biraz degistirelim dedik, bu degeri carpim disina alarak bir kac toplam olarak gosterebiliriz. Bu toplamlar mesela bir kullanicinin ne kadar yanli (bias) not verdigini, ya da bir filmin kabaca, ortalama nasil not almaya meyilli oldugunu modelleyebilirler (ki bu da bir yanlilik olcusu). Bu durumda bir begeni notunu tahmin edecek formül soyle gosterilebilir,

$$\hat{r}_{ui} = \mu + b_i + b_u + q_i^T p_u$$

μ bir skalar, tum filmlere verilen ortalama yi gosteriyor, ki tum begenilerin ortalama si uzerinden basit bir sekilde hizla hesaplanabilir. \hat{r}_{ui} 'ya bir tahmin dedik cunku modelimizdeki vektorlerin degerlerini bulduktan sonra (egitim verisiyle bu hesabi yapacagiz) modeli kullanarak r_{ui} 'yi (gercek not) bulabiliyoruz.

Egitim için ne yapmalı? Minimize edeceğimiz bir hedef fonksiyonu bulabiliriz, ki cogunlukla bu karesi alinmis hata uzerinden olur. Bunun için gerçek not r_{ui} degerinden tahmin notu \hat{r}_{ui} 'yi cikartip karesini alabiliriz. Bu islemi tum u, i'ler için yaparak bu toplami minimize etmeye ugrasabiliriz.

$$\min_{b^*, q^*, p^*} \sum_{u,i} (r_{ui} - \hat{r}_{ui})^2 + \lambda (b_i^2 + b_u^2 + \|q_i\|^2 + \|p_u\|^2)$$

$$= \min_{b^*, q^*, p^*} \sum_{u,i} (r_{ui} - \mu - b_i - b_u - q_i^T p_u)^2 + \lambda(b_i^2 + b_u^2 + \|q_i\|^2 + \|p_u\|^2)$$

$$e_{ui} := r_{ui} - \hat{r}_{ui}$$

$$b_u \leftarrow b_u + \gamma(e_{ui} - \lambda \cdot b_u)$$

$$b_i \leftarrow b_i + \gamma(e_{ui} - \lambda \cdot b_i)$$

$$q_i \leftarrow q_i + \gamma(e_{ui} \cdot p_u - \lambda \cdot q_i)$$

$$p_u \leftarrow p_u + \gamma(e_{ui} \cdot q_i - \lambda \cdot p_u)$$

```

from numpy.linalg import linalg as la
import numpy as np
import random
import pandas as pd, os

def create_training_test(df, collim=2, rowlim=200):
    test_data = []
    df_train = df.copy()
    for u in range(df.shape[0]):
        row = df.ix[u]; idxs = row.index[row.notnull()]
        if len(idxs) > collim:
            i = random.choice(idxs); val = df.ix[u,i]
            test_data.append([u,i,val])
            df_train.ix[u,i] = np.nan
        if len(test_data) > rowlim: break
    return df_train, test_data

def ssvd(df_train, rank):
    print 'rank', rank
    gamma = 0.02 # regularization
    lam = 0.05

    mu = df_train.mean().mean()
    m,n = df_train.shape
    c = 0.03
    b_u = np.ones(m) * c
    b_i = np.ones(n) * c
    p_u = np.ones((m, rank)) * c
    q_i = np.ones((rank, n)) * c
    r_ui = np.array(df_train)
    for u in range(m):
        #print "user", u
        row = df_train.ix[u]; idxs = row.index[row.notnull()]
        for i in idxs:

```

```

        i = int(i)
        r_ui_hat = mu + b_i[i] + b_u[u] + np.dot(q_i[:,i].T,p_u[u,:])
        e_ui = r_ui[u,i] - r_ui_hat
        b_u[u] = b_u[u] + gamma * (e_ui - lam*b_u[u])
        b_i[i] = b_i[i] + gamma * (e_ui - lam*b_i[i])
        q_i[:,i] = q_i[:,i] + gamma * (e_ui*p_u[u,:].T - lam*q_i[:,i])
        p_u[u,:] = p_u[u,:] + gamma * (e_ui*q_i[:,i].T - lam*p_u[u,:])
    return mu,b_u,b_i,q_i,p_u

import pandas as pd
import ssvd
d = np.array(
[[ 5.,    5.,    3.,   nan,    5.,    5.],
 [ 5.,   nan,    4.,   nan,    4.,    4.],
 [ nan,    3.,   nan,    5.,    4.,    5.],
 [ 5.,    4.,    3.,    3.,    5.,    5.],
 [ 5.,    5.,   nan,   nan,   nan,    5.]
])
data = pd.DataFrame (d, columns=['0','1','2','3','4','5'],
                      index=['Ben','Tom','John','Fred','Bob'])
mu,b_u,b_i,q_i,p_u = ssvd.ssvd(data,rank=3)
print mu
print 'b_u',b_u
print 'b_i',b_i
print 'q_i',q_i
print 'p_u',p_u
u = 4; i = 2
r_ui_hat = mu + b_i[i] + b_u[u] + np.dot(q_i[:,i].T,p_u[u,:])
print r_ui_hat

rank 3
4.31388888889
b_u [ 0.05129388  0.01927226  0.0206893  0.0065487  0.06568321]
b_i [ 0.07820389  0.01958841 -0.03217881  0.01561187  0.04071886  0.07140383]
q_i [[ 0.03132989  0.02957741  0.02802317  0.02951804  0.0301854  0.03108419]
 [ 0.03132989  0.02957741  0.02802317  0.02951804  0.0301854  0.03108419]
 [ 0.03132989  0.02957741  0.02802317  0.02951804  0.0301854  0.03108419]]
p_u [[ 0.03053543  0.03053543  0.03053543]
 [ 0.0295772  0.0295772  0.0295772 ]
 [ 0.02963018  0.02963018  0.02963018]
 [ 0.02921864  0.02921864  0.02921864]
 [ 0.03100583  0.03100583  0.03100583]]
4.34999993855

import pandas as pd, os
df = pd.read_csv("%s/Downloads/movielens.csv" % os.environ['HOME'] ,sep=';')
print df.shape
df = df.ix[:,1:3700] # id kolonunu atla,
df.columns = range(3699) # kolon degerlerini tekrar indisle
print df.shape

(6040, 3731)
(6040, 3699)

```

```
import ssvd
df_train, test_data = ssvd.create_training_test(df, rowlim=500, collim=300)
print len(test_data)
```

501

```
import ssvd
mu, b_u, b_i, q_i, p_u = ssvd.ssvd(df_train, rank=25)
print 'mu', mu
```

```
rank 25
mu 3.23841096846
```

```
rmse = 0; n = 0
for u, i, real in test_data:
    r_ui_hat = mu + b_i[i] + b_u[u] + np.dot(q_i[:, i].T, p_u[u, :])
    rmse += (real - r_ui_hat)**2
    n += 1
    #print u, i, real, r_ui_hat
print "rmse", np.sqrt(rmse / n)
```

```
rmse 0.878340489577
```

Kaynaklar

<http://sifter.org/~simon/journal/20061211.html>

Koren, Bell, *Recommender Systems Handbook*, http://www.cs.bme.hu/nagyadat/Recommender_systems_handbook.pdf

<http://www2.research.att.com/~volinsky/papers/ieeecomputer.pdf>

<http://www.cs.nyu.edu/~yann/talks/lecun-20071207-nonconvex.pdf>

http://courses.cs.washington.edu/courses/cse528/09sp/sanger_pca_nn.pdf

<http://users.ics.aalto.fi/oja/Oja1982.pdf>

<http://arxiv.org/pdf/1308.3509>

http://www.maths.qmul.ac.uk/~wj/MTH5110/notes/MAS235_lecturenotes1.pdf

<http://heim.ifi.uio.no/~tom/powerandqrslides.pdf>

<http://math.stackexchange.com/questions/649701/gradient-descent-on-non-convex-function-works-but-how>