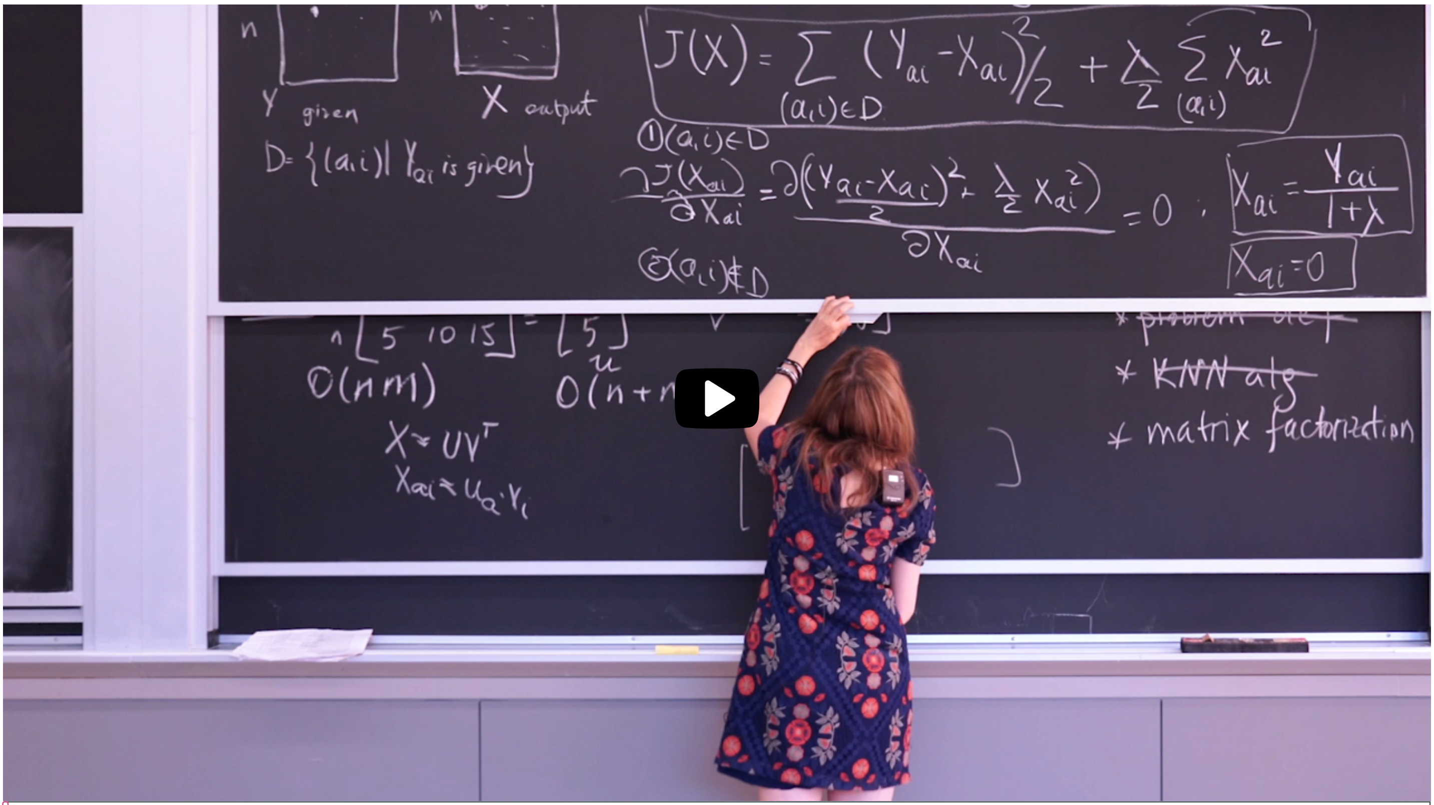
6. Alternating Minimization

Alternating Minimization



0:00 / 0:00 1.50x

Video

Download video file

Transcripts

Download SubRip (.srt) file

Download Text (.txt) file

Alternating Minimization Concept Question

1/1 point (graded)

As in the video above, we now want to find U and V that minimize our new objective

$$J = \sum_{(a,i)\in D} \frac{\left(Y_{ai} - [UV^T]_{ai}\right)^2}{2} + \frac{\lambda}{2}\left(\sum_{a,k} U_{ak}^2 + \sum_{i,k} V_{ik}^2\right).$$

To simplify the problem, we fix U and solve for V, then fix V to be the result from the previous step and solve for U, and repeat this alternate process until we find the solution.

Consider the case $k=1$. The matrices U and V reduce to vectors u and v, such that $u_a = U_{a1}$ and $v_i = V_{i1}$.

When v is fixed, minimizing J becomes equivalent to minimizing ...

- [ ] $\frac{\left(Y_{ai} - u_a v_i\right)^2}{2} + \frac{\lambda}{2}\sum_a (u_a)^2$

- [x] $\sum_{(a,i)\in D} \frac{\left(Y_{ai} - u_a v_i\right)^2}{2} + \frac{\lambda}{2}\sum_a (u_a)^2$

$$\sum_{(a,i)\in D} \frac{(Y_{ai} - u_a v_i)^2}{2}$$

$$\sum_{(a,i)\in D} \frac{(Y_{ai} - u_a v_i)^2}{2} + \frac{\lambda}{2} \sum_i (v_i)^2$$

Solution:

Regarding terms containing only V as constants, minimizing J is equivalent to minimizing

$$\sum_{(a,i)\in D} \frac{(Y_{ai} - u_a v_i)^2}{2} + \frac{\lambda}{2} \sum_a (u_a)^2.$$

Submit You have used 1 of 3 attempts

Answers are displayed within the problem

Fixing V and Finding U

2.0/2 points (graded)

Now, assume we have 2 users, 3 movies, and a 2 by 3 matrix Y given by

$$Y = \begin{bmatrix} 1 & 8 & ? \\ 2 & ? & 5 \end{bmatrix}$$

Our goal is to find U and V such that $X = UV^T$ closely approximates the observed ratings in Y.

Assume we start by fixing V to initial values of $[4, 2, 1]^T$. Find the optimal 2×1 vector U in this case. (Express your answer in terms of λ).

First element of U is:

20/(lambda+20) **Answer:** 20/(20+lambda)

The second element of U is:

13/(lambda+17) **Answer:** 13/(17+lambda)

STANDARD NOTATION

Solution:

To compute the first element (u_1), compute the objective (ignore missing elements from Y), derive and compare to zero to find the minimum:

$$\frac{\partial}{\partial u_1}\left[\frac{(1-4u_1)^2}{2} + \frac{(8-2u_1)^2}{2} + \frac{\lambda}{2}u_1^2\right] = (\lambda + 20)\, u_1 - 20 = 0.$$

Submit You have used 2 of 3 attempts

Answers are displayed within the problem

Discussion

Hide Discussion

Topic: Unit 2 Nonlinear Classification, Linear regression, Collaborative Filtering (2 weeks):Lecture 7. Recommender Systems / 6. Alternating Minimization

Add a Post

Sage & Coffee - Alternating Minimization

question posted 8 months ago by **GuilhermeKinzel**

Tried some ways to see this algorithm working, so, for this one I used Sage (its came from Python) because of symbolic equations/derivatives. Using Sage Cell Server:

https://sagecell.sagemath.org/

I used the same dataset of the video, with:

$$Y = \begin{pmatrix} 5 & ? & 7 \\ 1 & 2 & ? \end{pmatrix}$$

$$u_1 = \begin{pmatrix} \frac{66}{69} & \frac{16}{54} \end{pmatrix}$$

X will be updated every iteration, but at even iteration is calculated $u \cdot v = X$ without random values. For instance:

- First iteration: update v
- Second iteration: update u
- Is calculated $u \cdot v = X$, the line "X_Done = uAntigo.transpose()*vAntigo"

```
SHOW_EVERY_STEP = 0
var('U_1, U_2, V_1, V_2, V_3')

Y = matrix(SR,[[5,0,7],[1,2,0]])

#v = matrix(SR,[[V_1, V_2, V_3]])
v = matrix(SR,[[2, 7, 8]]) ##Random vector

for i in range(100):

    u = matrix(SR,[[U_1, U_2]])
    X = u.transpose()*v
    #show(X) Importante

    H = 0 ##Initialize
    for i in range(X.nrows()):
        H = 0
        for j in range(X.ncols()):
            if(Y[i,j] != 0):
                H = H + (Y[i,j] - X[i,j])^2/2
        Eq = H + u[0,i]^2/2
        #show(Eq)
        Eqq = diff(Eq, u[0,i])
        SolveEq = Eqq.solve(u[0,i])
        u[0,i] = SolveEq[0].rhs().n()
    #u.n()
    uAntigo = u
    #show(u)

    ## Agora zera o v
    v = matrix(SR,[[V_1, V_2, V_3]]) ##Random vector
    X = u.transpose()*v
    #show(X) ##Importante

    H = 0 ##Initialize
    for j in range(X.ncols()):
        H = 0
        for i in range(X.nrows()):
            if(Y[i,j] != 0):
                H = H + (Y[i,j] - X[i,j])^2/2
        Eq = H + v[0,j]^2/2
        #show(Eq)
        Eqq = diff(Eq, v[0,j])
        #show(Eqq)
        SolveEq = Eqq.solve(v[0,j])
        v[0,j] = SolveEq[0].rhs().n()
    #v = v.n()
    vAntigo = v
    #show(v)

    X_Done = uAntigo.transpose()*vAntigo
    if SHOW_EVERY_STEP:
        show(X_Done)

print("Finished")
X_Done = uAntigo.transpose()*vAntigo
if SHOW_EVERY_STEP == 0:
    show(X_Done)
#X_Done
```

Well, after some iterations, only a far away decimals change between them (ocurring convergence I guess). I took X_{100} just for instance.

$$X_{100} = \begin{pmatrix} 4.36412901058397 & 2.71245640601817 & 6.19908252148648 \\ 1.25122626729700 & 0.777680195951360 & 1.77732025456042 \end{pmatrix}.$$

Comparing with:

$$Y = \begin{pmatrix} 5 & ? & 7 \\ 1 & 2 & ? \end{pmatrix}$$

I guess the user 1, movie 2 (Y_{12}) will be rated as "bad movie" (3 stars for him) if he watch. I guess; and the user 2 watch the movie 3 (Y_{23}), he will probably rate with 2 stars.

If someone want to see all the steps, remove the comments of show(X) or show(u) or show(v). The teachers matrix will appears as the first one.

Remark: with no closed formula or solutions, I don't know if the algo is 100% right (I even don't know why X_{22} become smaller than X_{21}). Disclaimer needed. 😉

This post is visible to everyone.

dkonomis (Staff)
8 months ago - marked as answer 8 months ago by **dkonomis** (Staff)

It looks 100% correct to me, thanks for sharing your solution. I get the same values for X_{100}.

Add a comment

Preview

Submit