Collaborative Filtering

Problem Statement

Let us consider a restaurant rating problem. There are 4 restaurant and 3 people. Not everyone have gone through every restaurant. So the rating matrix ($T_{r,p}$) for restaurant r and person p will looks something like the following (The row is index by restaurant and the column represent person). The dash spot means that there is no rating for that person yet. Specifically, the rating person number 1 (p=1) gives restaurant number 2 is the number in the row number 2+1=3 and column 1+1 = 2, which is 5.

$$\tilde{T}_{r,p} = \begin{bmatrix} 5 & 2 & 5 \\ 5 & 1 & - \\ - & 5 & - \\ - & 3 & 0 \end{bmatrix} \tag{1}$$

In the table above the rating of person $p = 2$ for restaurant $r = 1$ is not given. The goal is the use information from the rating $p = 2$ gives to other restaurant and how restaurant $r = 1$ is rated by others.

The idea is the following, on the table above, we found that person $p = 1$ and person $p = 2$ rated the first restuarant very similarly. That would means that if person $p = 1$ rated restaurant $r = 1$ as 5. It's probably likely that person $p = 2$ would rate restaurant $r = 1$ the same.

Hypothesis

The justification for how this happens would be that there is an underlying attribute for the restuarant like how spicy the restaurant is and how expensive the restaurant. There is also a preference for each person like spiciness and price sensitivity of each person.

This can be quantified with a vector for Example restaurant number the attribute for restaurant number r can be written as

$$\vec{\rho}_r = [5, -5, -1, 10, -100] \tag{2}$$

and the preference for person p can be written as

$$\pi_p = [-1, 0, 4, 19, 20] \tag{3}$$

The guess for the restaurant rating should be some function of these two vector

$$T_{r,p} = f(\vec{\rho}_r, \vec{\pi}_p)$$

One of the choice for this function is a simple dot product.

$$T_{r,p} = \vec{\rho}_r^T \vec{\pi}_p$$

This can be written very succintly using matrix notation

$$T = \begin{bmatrix} \vec{\rho}_1 \cdot \vec{\pi}_1 & \vec{\rho}_1 \cdot \vec{\pi}_2 & \vec{\rho}_1 \cdot \vec{\pi}_3 & \cdots \\ \vec{\rho}_2 \cdot \vec{\pi}_1 & \vec{\rho}_2 \cdot \vec{\pi}_2 & \vec{\rho}_2 \cdot \vec{\pi}_3 & \cdots \\ \vec{\rho}_3 \cdot \vec{\pi}_1 & \vec{\rho}_3 \cdot \vec{\pi}_2 & \vec{\rho}_3 \cdot \vec{\pi}_3 & \cdots \\ \vdots & \vdots & \vdots & \end{bmatrix} = \begin{bmatrix} — & \vec{\rho}_1 & — \\ — & \vec{\rho}_2 & — \\ — & \vec{\rho}_3 & — \\ & \vdots & \\ — & \vec{\rho}_r & — \\ & \vdots & \end{bmatrix} \times \begin{bmatrix} | & | & | & | & | & | \\ \vec{\pi}_1 & \vec{\pi}_3 & \vec{\pi}_3 & \cdots & \vec{\pi}_p & \cdots \\ | & | & | & | & | & | \end{bmatrix} = R^T \times P \tag{4}$$

where we define the matrix R to be matrix of restaurant attribute and matrix P to be matrix of person perference.

Then all we need to do is to give a score on how bad our guess for R and P is. One way to do it is to use the usual square error.

$$cost(R, P) = \sum_{\text{ratings defined in } T} (T - R^T P)^{\hat{2}} \tag{5}$$

where the $\hat{2}$ denotes element-wise square and not matrix multiplication. Also note that the sum is the sum over all the element in T in which the rating is defined. This is because we don't want to penalized the rating we haven't built just yet.

However the problem is that for the data given, we have neither R or P. To get R you will need to hire a bunch of expert and to get P you need to pay for survey. Both of these are expensive. All we have is a T.

Gradient Descent Revisited

So our job is to find both R and P. This can be done by start guess some value for R and P and then keep improving R and P based on our cost function. We can use the old trick we did before, the gradient descent. But this time it's a matrix version.

Updating both R and P at the same time can be really hard and time consuming. The idea here to try to update π_i one at a time to give us the best rating. Once we are done with π's we will go that given our new P we will try to find the best r_i for each restaurant.

This is a bit different from gradient descent we did last time in the aspect that when we try to optimize each π_i we hold all the other number constants.

Let us find the update rule for cost function defined in Equation 5. First, to make things more clear let us expand 5

$$cost(\vec{\rho}_1, \vec{\rho}_2, \ldots, \vec{\pi}_1, \vec{\pi}_2, \ldots) = \sum_r \sum_p h_{rp}(T_{rp} - \vec{\rho}_r \cdot \vec{\pi}_p)^2 \tag{6}$$

where

$$h_{rp} = \begin{cases} 0 & \text{if } T_{rp} \text{ is not defined} \\ 1 & \text{if } T_{rp} \text{ is defined} \end{cases} \tag{7}$$

This will simplify a lot of math.

Let us say if we want to update component number 7 of $\vec{\pi}_5$ written as $\pi_{5,7}$, we will treat all the other stuff as constant this bring as back to normal gradient descent rule

$$\vec{v} \leftarrow \vec{v} - \lambda \boldsymbol{\nabla}_v f$$

let us use this on equation 6 so we need to calculate

$$\frac{\partial cost}{\partial \pi_{5,7}} = \frac{\partial}{\partial \pi_{5,7}} \sum_r h_{r5}(T_{r5} - \vec{\rho}_r \cdot \vec{\pi}_5)^2 \tag{8}$$

here we use the fact that the term in sum over p will has nothing to do with $\pi_{5,7}$ unless p=5. Bringing the derivative in side we have

$$\frac{\partial cost}{\partial \pi_{5,7}} = -2 \sum_r h_{r5}(T_{r5} - \vec{\rho}_r \cdot \vec{\pi}_5)\rho_{r,7} \tag{9}$$

Try to convince yourself of this. This means that in general

$$\frac{\partial cost}{\partial \pi_{p,j}} = -2 \sum_r h_{rp}(T_{rp} - \vec{\rho}_r \cdot \vec{\pi}_p)\rho_{r,j} \tag{10}$$

or

$$\boldsymbol{\nabla}_{\vec{\pi_p}} cost = -2 \sum_r h_{rp}(T_{rp} - \vec{\rho}_r \cdot \vec{\pi}_p)\vec{\rho}_r \tag{11}$$

Thus the update rule for $\vec{\pi}_p$ read

$$\vec{\pi}_p \leftarrow \vec{\pi}_p + 2\lambda \sum_r h_{rp}(T_{rp} - \vec{\rho}_r \cdot \vec{\pi}_p)\vec{\rho}_r$$

or in fancy matrix equation we have (Try to convice your self that this is the matrix equation we want)

$$P \leftarrow P + 2\lambda \left(H \otimes (T - R^T P)^T\right) \times R \tag{12}$$

where H is the matrix for h_{rp}, $\otimes$ denotes element-wise multiplication.

Similarly the update rule for $\vec{\rho}_r$ reads

$$\vec{\rho}_r \leftarrow \vec{\rho}_r + 2\lambda \sum_p h_{rp}(T_{rp} - \vec{\rho}_r \cdot \vec{\pi}_p)\vec{\pi}_p$$

or in fancy matrix equation we have

$$R \leftarrow R + 2\lambda \left(H \otimes (T - R^T P)\right) \times P \tag{13}$$

note the lack of transpose.

This means that all we need to is to keep update R and P using Equation 12 and 13. As we iterate through, we will get better and better $T = R^T P$.

Caveat

Once we have done all the above we will find that it our model will predict rating more than 5 and negative rating which can be considered a non-sense. This is intrinsic to how the rating is calculate we are doing just a dot product. This means that the output will range from $-\infty$ to ∞. To fix this we need to use scale and shifted logistic function and all the update rule will need to be rederived. You will find it in your homework.