Week 16

August 18, 2021

Recommender systems

- Many technological businesses consider recommender systems to be critical (amazon, Ebay, iTunes genius).
- Based on previous purchases, try to propose new content to you.
- It's not so much a method as it is a concept.

Example: predict movie ratings.

- You work at a firm that sells movies.
- You allow viewers to rate movies on a scale of 1 to 5.
- You have five films.
- You also have four users.

Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)	x_1 (romance)	x_2 (action)
Love at last	5	5	0	0	0.9	0
Romance forever	5	?	?	0	1.0	0.01
Cute puppies of love	?	4	0	?	0.99	0
Nonstop car chases	0	0	5	4	0.1	1.0
Swords vs. karate	0	0	5	?	0	0.9

- n_u - number of users.
- n_m - number of movies.
- $r(i, j)$ - 1 if user j has rated movie i.
- $y^{(i,j)}$ - rating given by user j to move i.
- If we have features like this, a feature vector may recommend each film.
- For each film, add an additional feature that is x0 = 1.
- So we have a $[3 \times 1]$ vector for each film, which for film number three ("Cute Puppies of Love") would be:

$$x^{(3)} = \begin{bmatrix} 1 \\ 0.99 \\ 0 \end{bmatrix}$$

So, let's take a look at user 1 (Alice) and see what she thinks of the modern classic Cute Puppies of Love (CPOL). She is associated with the following parameter vector:

$$\theta^{(1)} = \begin{bmatrix} 0 \\ 5 \\ 0 \end{bmatrix}$$

Our prediction:

$$(\theta^1)^T x^3 = (0 \cdot 1) + (5 \cdot 0.99) + (0 \cdot 0) = 4.95$$

How do we learn (θ^j)?

This is analogous to linear regression with least-squares error:

$$min_{\theta^j} = \frac{1}{2m^{(j)}} \sum_{i:r(i,j)=1} ((\theta^{(j)})^T (x^{(i)}) - y^{(i,j)})^2 + \frac{\lambda}{2m^{(j)}} \sum_{k=1}^{n} (\theta_k^{(j)})^2$$

We have the gradient descent algorithm to find the minimum:

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha \sum_{i:r(i,j)=1} ((\theta^{(j)})^T (x^{(i)}) - y^{(i,j)}) x_k^{(i)} \ (for\ k = 0)$$

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha (\sum_{i:r(i,j)=1} ((\theta^{(j)})^T (x^{(i)}) - y^{(i,j)}) x_k^{(i)} + \lambda \theta_k^{(j)}) \ (for\ k \neq 0)$$

Collaborative filtering

The collaborative filtering algorithm has a very intriguing property: it can learn what characteristics it needs to learn for itself.

If we are given user preferences $(\theta^{(1)}, ..., \theta^{(n_u)})$ we may use them to find out the film's features $(x^{(1)}, ..., x^{(n_m)})$ and vice versa:

$$min_{x^{(1)},...,x^{(n_m)}} \frac{1}{2} \sum_{i=1}^{n_m} \sum_{i:r(i,j)=1} ((\theta^{(j)})^T (x^{(i)}) - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^{n} (x_k^{(i)})^2$$

One thing you could do is:

- Randomly initialize parameter.
- Go back and forth in time.

Vectorization: Low rank matrix factorization

How can we enhance the collaborative filtering algorithm now that we've looked at it?

So, in our previous example, take all ratings from all users and organize them into a matrix Y.

5 movies and 4 users, give us a $[5 \times 4]$ matrix:

$$Y = \begin{pmatrix} 5 & 5 & 0 & 0 \\ 5 & ? & ? & 0 \\ ? & 4 & 0 & ? \\ 0 & 0 & 5 & 4 \\ 0 & 0 & 5 & 0 \end{pmatrix}$$

Given [Y] there's another way of writing out all the predicted ratings:

$$X \cdot \Theta^T = \begin{pmatrix} (\theta^{(1)})^T(x^{(1)}) & (\theta^{(2)})^T(x^{(1)}) & ... & (\theta^{(n_u)})^T(x^{(1)}) \\ (\theta^{(1)})^T(x^{(2)}) & (\theta^{(2)})^T(x^{(2)}) & ... & (\theta^{(n_u)})^T(x^{(2)}) \\ \vdots & \vdots & \vdots & \vdots \\ (\theta^{(1)})^T(x^{(n_m)}) & (\theta^{(2)})^T(x^{(n_m)}) & ... & (\theta^{(n_u)})^T(x^{(n_m)}) \end{pmatrix}$$

Where matrix X is constructed by taking all the features for each movie and stacking them in rows:

$$X = \begin{bmatrix} -(x^{(1)})^T- \\ -(x^{(2)})^T- \\ \vdots \\ -(x^{(n_m)})^T- \end{bmatrix}$$

And matrix Θ is constructed by taking all the features for each movie and stacking them in rows:

$$\Theta = \begin{bmatrix} -(\theta^{(1)})^T- \\ -(\theta^{(2)})^T- \\ \vdots \\ -(\theta^{(n_u)})^T- \end{bmatrix}$$

Mean Normalization

Consider a user who hasn't reviewed any movies.

Movie	Alice (1)	Bob (2)	Carol (3)	Dave (4)	Eve (5)
Love at last	5	5	0	0	?
Romance forever	5	?	?	0	?
Cute puppies of love	?	4	0	?	?
Nonstop car chases	0	0	5	4	?
Swords vs. karate	0	0	5	?	?

$$Y = \begin{bmatrix} 5 & 5 & 0 & 0 & ? \\ 5 & ? & ? & 0 & ? \\ ? & 4 & 0 & ? & ? \\ 0 & 0 & 5 & 4 & ? \\ 0 & 0 & 5 & 0 & ? \end{bmatrix}$$

- There are no films for which r(i,j) = 1.
- So this term places no role in determining θ^5.
- So we're just minimizing the final regularization term.

$$\min_{\substack{x^{(1)},\ldots,x^{(n_m)} \\ \theta^{(1)},\ldots,\theta^{(n_u)}}} \frac{1}{2} \sum_{(i,j):r(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^{n} (x_k^{(i)})^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^{n} (\theta_k^{(j)})^2$$

This term is irrelevant

We're only minimizing this

Which can for our single example be simplified to this $\frac{\lambda}{2}\left[(\theta_1^{(5)})^2 + (\theta_2^{(5)})^2\right]$

- As previously, put all of our ratings into matrix Y.
- We now compute the average rating for each movie and store it in a n_m - dimensional column vector.

$$\mu = \begin{bmatrix} 2.5 \\ 2.5 \\ 2 \\ 2.25 \\ 1.25 \end{bmatrix}$$

- If we take a look at all of the movie ratings in [Y], we can subtract the mean rating.

$$Y = \begin{pmatrix} 2.5 & 2.5 & -2.5 & -2.5 & ? \\ 2.5 & ? & ? & -2.5 & ? \\ ? & 2 & -2 & ? & ? \\ -2.25 & -2.25 & 2.75 & 1.75 & ? \\ -1.25 & -1.25 & 3.75 & -1.25 & ? \end{pmatrix}$$

- That is, we normalize each film to have an average rating of 0.