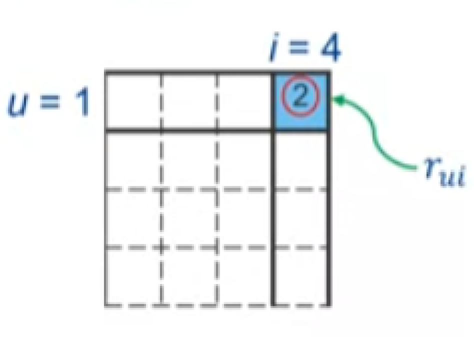
# Factorization Machines

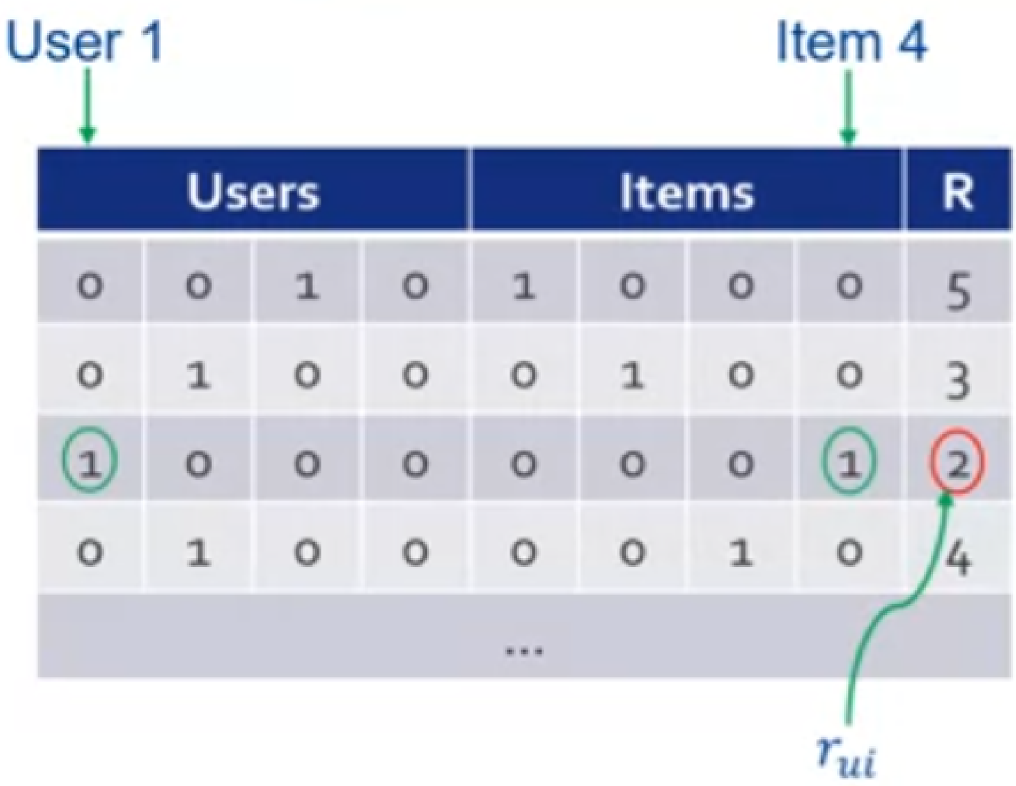
It is a generalization of the linear regression model and the matrix factorization model.

Input Data Representation:

User_item matrix is represented by following.




However, it can also be represented as following.



Equation:

$$\tilde{r}^{(k)} = \omega_0 + \sum_{i=1}^{n} \omega_i \cdot x_i^{(k)} + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \omega_{i,j} \cdot x_i^{(k)} \cdot x_j^{(k)}$$

columns

row

$x_i^{(k)}$ : element of the table

| Users | | | | Items | | | | R |
|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 5 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 3 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 2 |

$$\tilde{r}^{(k)} = \omega_0 + \sum_{i=1}^{n} \omega_i \cdot x_i^{(k)} + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \omega_{i,j} \cdot x_i^{(k)} \cdot x_j^{(k)}$$

$\omega_0, \ \omega_i, \ \omega_{i,j}$ : parameters to be learned to estimate $\tilde{r}$

$$\hat{y}(\mathbf{x}) := w_0 + \sum_{i=1}^{n} w_i \, x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle \, x_i \, x_j \qquad (1)$$

where the model parameters that have to be estimated are:

$$w_0 \in \mathbb{R}, \quad \mathbf{w} \in \mathbb{R}^n, \quad \mathbf{V} \in \mathbb{R}^{n \times k} \qquad (2)$$

And $\langle \cdot, \cdot \rangle$ is the dot product of two vectors of size $k$:

$$\langle \mathbf{v}_i, \mathbf{v}_j \rangle := \sum_{f=1}^{k} v_{i,f} \cdot v_{j,f} \qquad (3)$$

$W_0$ : Unknown global bias.

$W_i$ : Vector of unknown biases.

$W_{i,j}$ : Matrix of unknown cofficients.

Differnce between Factorization Machine and Matrix Factorization:

1. Matrix Factorization are bound to have only one user and one item per interaction.
2. Factorization Machine can model the group preference and provide a better recommendation.
3. we can extend the model without changing the equations of model.

   a.) We can create a collobrative and content based model.

| "Paolo" | | | | "Titanic" | | | | "Di Caprio" | | | | "Romance" | | | "Drama" | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Users** | | | | **Items** | | | | **Actors** | | | | **Genres** | | | | **R** |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 5 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 3 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 2 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 4 |
| ... | | | | | | | | | | | | | | | | |

In recommender systems, where Matrix Factorization is generally used, we cannot use side-features. For a movie recommendation system, we cannot use the movie genres, its language etc in Matrix Factorization.

b.) We can extend the model with the demographic context, date of the week user have watched

4. Matrix Factorization is solely a collaborative filtering approach which needs user engagement on the items. So it doesn't work for what is called "cold start" problems. Think of a new movie released on Netflix. As no one would have watched it, matrix factorization doesn't work for it. But as Netflix would know the genre, actors, director etc, Factorization Machine can kick-start the recommendations for this movie from day 1 itself, which is a crucial component for many websites that use recommendation systems.

Also, when we are modeling group interactions, we can reduce the importance of some users and some items by assigning values of x_i smaller than one.

> For example, we can put 0.5 for the user number 1 and value of 1 for user number 2. This would mean that the opinion of user number 2 will have more importance.

## For Implicit ratings:

- A dataset with implicit ratings would only hold positive samples.

| Users | | | | Items | | | | Context | | | | Content | | | | R |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| ... | | | | | | | | | | | | | | | | |

We need to balance the positive and negative samples.

One approach to balance is to use the missing ratings as negative samples. However, for missing ratings we donot know whether the user saw the item or didn't like it or otherwise whether the user has never had interaction with the item. So, that item could potentially be good recommendation.

**Challenges Involved:**

1. Since we're using one-hot encoding. This causes NN to spew very bad results as they need a lot of data to train eficiently.
2. While FM, has yielded greate promise in many prediction , but its performance can be limited by its linearity as well as the modeling of pairwise(i.e., second-order)feature interactions only.

DeepFM:

Instead of using the one hot vectors, it is using the dense vectors from FFM and then running through neural net.

Combination of FM and NN.