Practice: Modeling Sparse Ratings Using Factorization Machines and the R API

The data set PUBLICEVALS contains information about ratings given by students to courses. A rating is a numeric value between 0 (very bad) and 4 (very good). Not all students take all courses, or even rate them. Therefore, the matrix of ratings is very sparse. The data source contains 88,344 rows and three columns.

Name	Model Role	Measurement Level	Description
STUDENT	Input	Nominal	ID of student
COURSE	Input	Nominal	ID of course
RATING	Target	Nominal	Rating given by student for course

1. From the Jupyter Lab Home directory, select the plus symbol to open a new page and then select R under Notebook to create a new notebook.

2. Load the SWAT and repr packages. Use the options functions to change the size of the graphical output.

3. Connect to the CAS server and create a connection object.

4. Use the cas.read.sas7bdat SWAT function to load the publicevals.sas7bdat file onto the CAS server and create a table object reference for the data. Name the new CAS table **evals**.

5. Use SWAT functionality to find the dimension and head of the table, and also the average course rating.

6. Load the simple action set and then use the distinct action to find the number of distinct values for all three variables of the **evals** table (student, course, rating). Then, use the freq action from the simple action set to find the frequency of each rating level.

7. Load the sampling action set and use the srs action to partition the data. Use 90% for training and the remaining 10% for validation. Set the partind argument to TRUE to add the partition indicator to the CAS table.

8. Build the factorization machine by loading the factmac action set and using the factmac action. The target for this model is rating and the inputs and nominals are both student and course. Set the maximum iterations and number of factors both to 20 and the learnstep for regularization to 0.1. Use the saveState and output arguments to save the model and the scored training data respectively. Find the MSE in the output for a model fit reference.

9. Score the validation data by first loading the aStore action set and then using the score action. Pass the score action the validation data using the partition indicator variable and use the rstore argument to pass the action the saved factorization machine model. Save the scored validation data as a CAS table and keep all three variables (student, course, rating) in the table using the copyVars argument.

10. Find the MSE for the validation data and compare it to the training data MSE. Update the CAS table reference using defCasTable and then add the error between the rating and predicted rating as a new variable to the table using open source functionality. Then add the squared error as another variable and use the mean function to find the MSE.

11. End the CAS session.