Recommender System for Compass Data

24th July 2018

Ravi Raju Krishna

kidaptive

Descriptive Analytics

Data Preprocessing

- Unique key for item : "media_id", Unique key for user:"learner_id"
  Rating of each learner to media : "like_rating"
- Total:
  No of learners : 44048 No of videos : 8110 No of events : 424659
- Events from 2016-06-07 to 2018-04-27
- Learners provide like_rating [1,2,3] scale
- Few (**3** events) outliers with 0 rating was filtered
- No of user,item pairs with multiple ratings : **60669**
  Identical Ratings : 55423, Distinct Ratings : 5246
- Hence only latest rating w.r.t. event_time were chosen
- Total:
  No of learners : 44048 No of videos : 8110 No of events : 363987
  (14% : **-60672**)

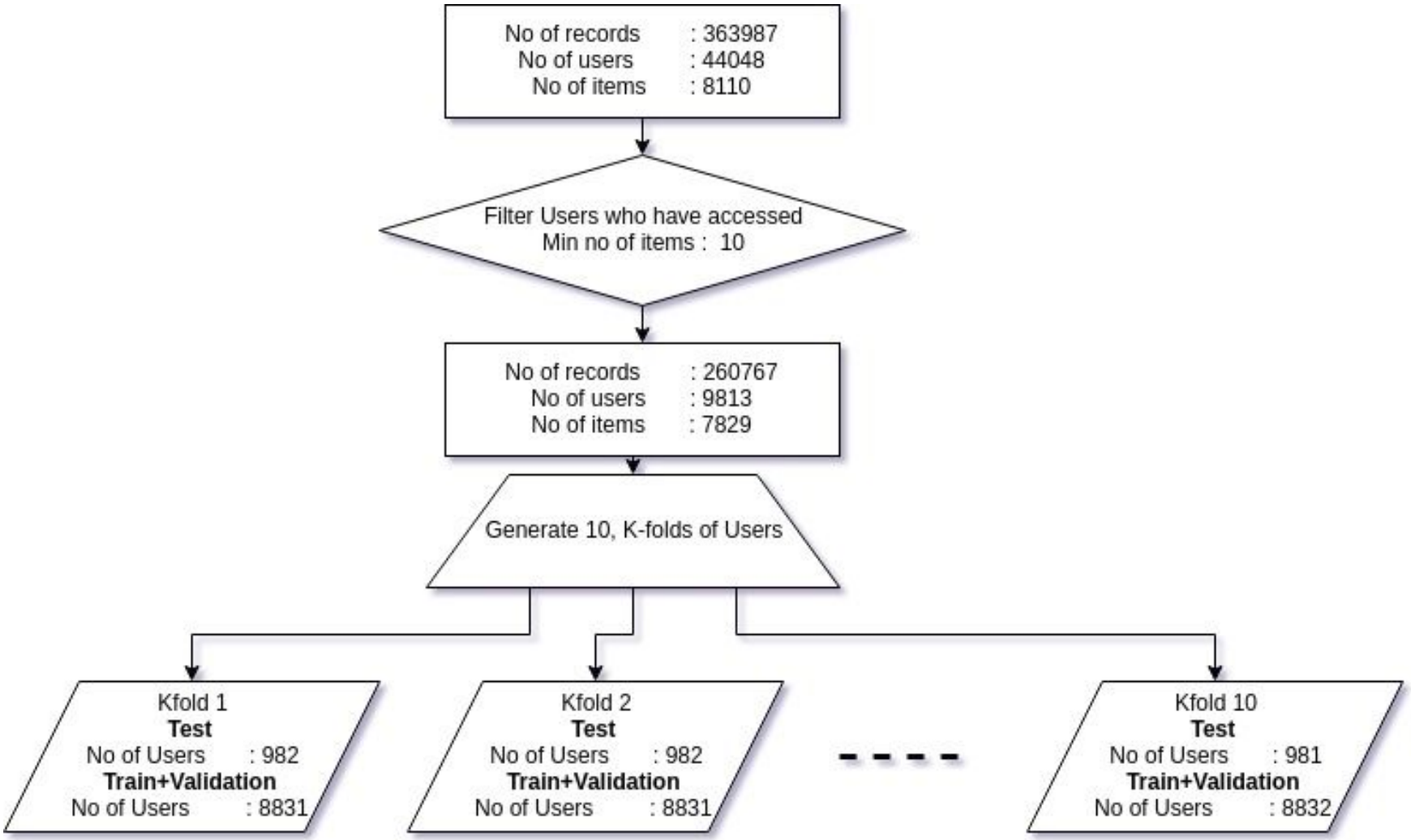
kidaptive

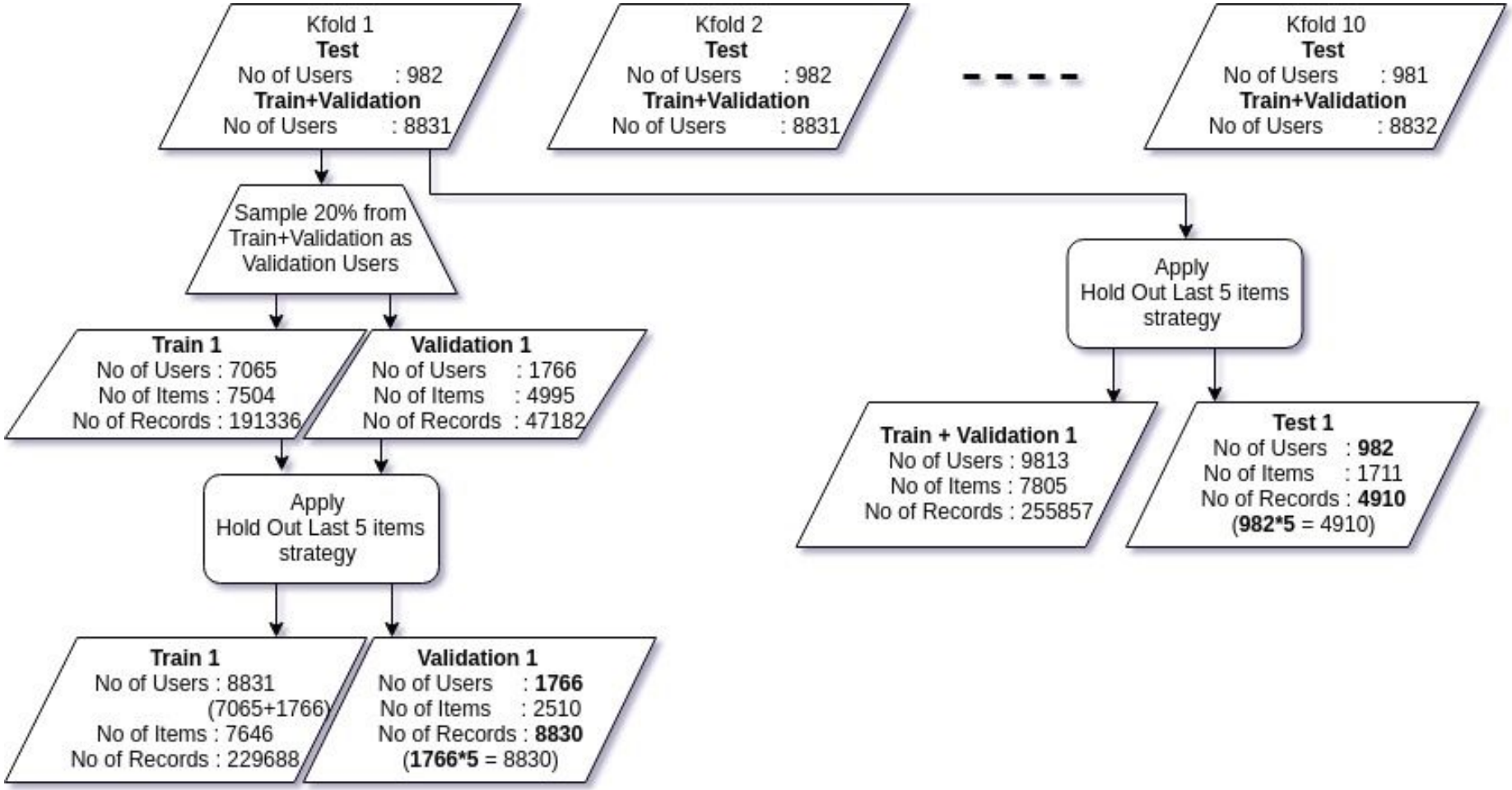
Train, Validation and Test Data

- Total:
  No of learners : 44048 No of videos : 8110 No of events : 363987
- Filter Users who have accessed min of 10 items
- Total:
  No of learners : 9813 No of videos : 7829 No of events : 260767
  (28% : -103220)
- Generate 10 k-folds of users, each fold having ~981 users as test and ~ 8832 as train+validation users

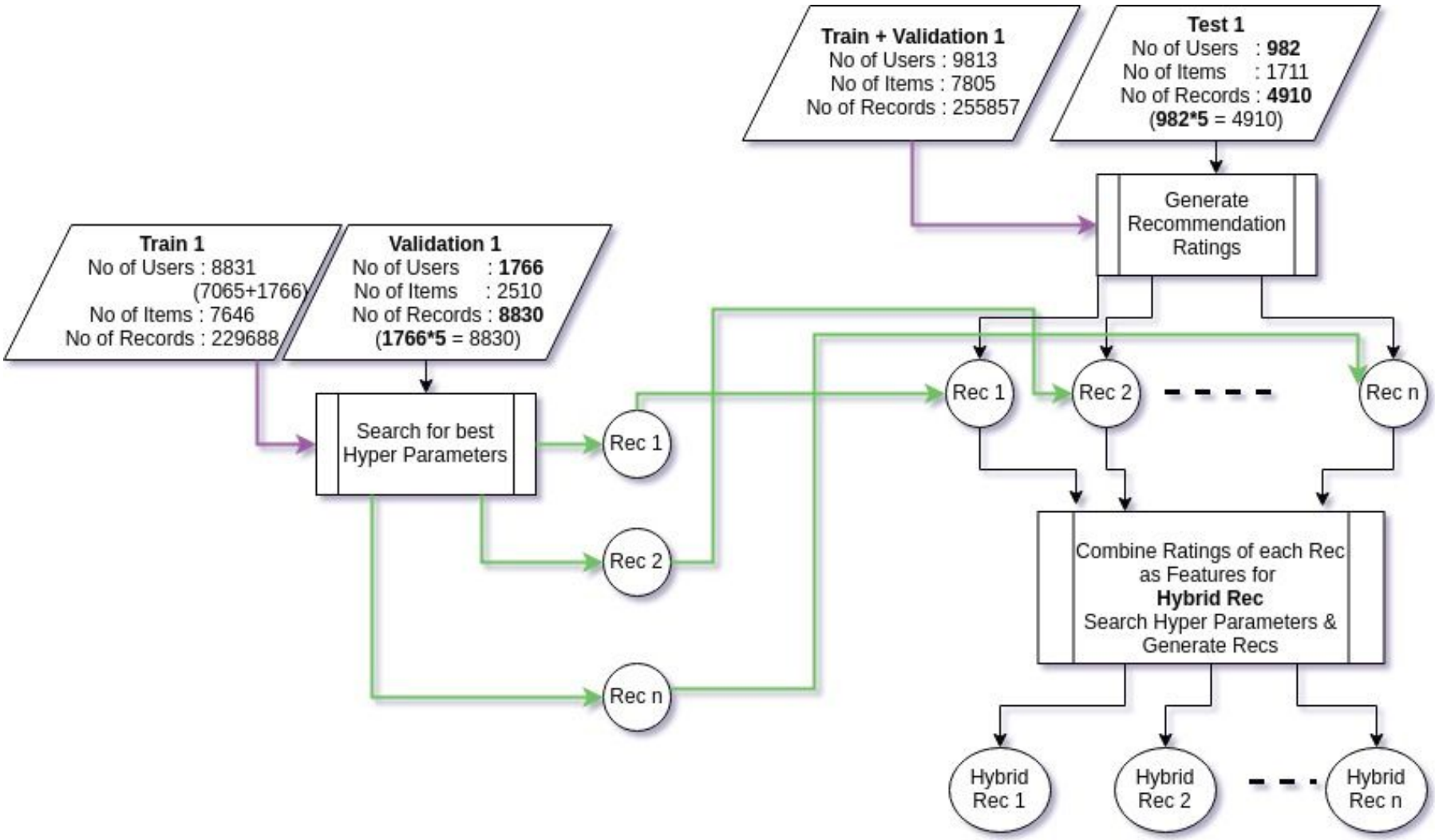
- Randomly sample 20% of train+validation users as validation

- Hold out last 5 items of validation users as validation data, adding n-5 items to training in each fold.
- Similarly apply hold_out_strategy for each k-fold of test users.


No of records : 363987
No of users : 44048
No of items : 8110
Filter Users who have accessed
Min no of items : 10
No of records : 260767
No of users : 9813
No of items : 7829
Generate 10, K-folds of Users
Kfold 1
Test
No of Users : 982
Train+Validation
No of Users : 8831
Kfold 2
Test
No of Users : 982
Train+Validation
No of Users : 8831
Kfold 10
Test
No of Users : 981
Train+Validation
No of Users : 8832


Kfold 1
Test
No of Users : 982
Train+Validation
No of Users : 8831
Kfold 2
Test
No of Users : 982
Train+Validation
No of Users : 8831
Kfold 10
Test
No of Users : 981
Train+Validation
No of Users : 8832
Sample 20% from Train+Validation as Validation Users
Apply Hold Out Last 5 items strategy
Train 1
No of Users : 7065
No of Items : 7504
No of Records : 191336
Validation 1
No of Users : 1766
No of Items : 4995
No of Records : 47182
Apply Hold Out Last 5 items strategy
Train + Validation 1
No of Users : 9813
No of Items : 7805
No of Records : 255857
Test 1
No of Users : 982
No of Items : 1711
No of Records : 4910
(982*5 = 4910)
Train 1
No of Users : 8831
(7065+1766)
No of Items : 7646
No of Records : 229688
Validation 1
No of Users : 1766
No of Items : 2510
No of Records : 8830
(1766*5 = 8830)

kidaptive


Train + Validation 1
No of Users : 9813
No of Items : 7805
No of Records : 255857
Test 1
No of Users : 982
No of Items : 1711
No of Records : 4910
(982*5 = 4910)
Generate Recommendation Ratings
Train 1
No of Users : 8831
(7065+1766)
No of Items : 7646
No of Records : 229688
Validation 1
No of Users : 1766
No of Items : 2510
No of Records : 8830
(1766*5 = 8830)
Search for best Hyper Parameters
Rec 1
Rec 2
Rec n
Rec 1
Rec 2
Rec n
Combine Ratings of each Rec as Features for Hybrid Rec Search Hyper Parameters & Generate Recs
Hybrid Rec 1
Hybrid Rec 2
Hybrid Rec n

Open Source Libraries

- Surprise Lib http://surpriselib.com/

  Simple Python **R**ecommendat**I**on **S**ystem **E**ngine.

  Surprise is a python scikit library for building and analyzing rating based recommender systems.

  Provide various ready-to-use prediction algorithms such as baseline algorithms, neighborhood methods, matrix factorization-based (SVD, PMF, SVD++, NMF). Various similarity measures (cosine, MSD, pearson…) are built-in.

- ML Flow https://mlflow.org/

  The MLflow Tracking component is an API and UI for logging parameters, code versions, metrics, and output files when running your machine learning code

kidaptive

Recommendation Algorithms (1/3)

- BaselineOnly

  Typical CF data exhibit large user and item effects—systematic tendencies for some users to give higher ratings than others—and for some items to receive higher ratings than others

  BaselineOnly is an algorithm to predict the baseline estimate for given user and item.

$$b_{ui} = \mu + b_u + b_i\,.$$

  The parameters $\mathbf{b_u}$ and $\mathbf{b_i}$ indicate the observed deviations of user u and item i, respectively, from the average.

  If user/item is unknown, then the respective bias is assumed to be zero.

  To estimate $\mathbf{b_u}$ and $\mathbf{b_i}$ the least squares problem is solved using ALS or SGD

$$\min_{b_*} \sum_{(u,i)\in\mathcal{K}} (r_{ui} - \mu - b_u - b_i)^2 + \lambda_1 \left(\sum_u b_u^2 + \sum_i b_i^2 \right)$$

kidaptive

Recommendation Algorithms (2/3)

- KNNBaseline

  User based methods estimate unknown ratings based on recorded ratings of like-minded users. In Item based methods, a rating is estimated using known ratings made by the same user on similar items.

  Pearson correlation coefficient, measures the tendency of users to rate items i and j similarly. **Pearson-baseline correlation coefficient** between all pairs of items (or users) uses baselines for centering instead of means.

$$\text{pearson\_baseline\_sim}(i,j) = \hat{\rho}_{ij} = \frac{\sum\limits_{u \in U_{ij}} (r_{ui} - b_{ui}) \cdot (r_{uj} - b_{uj})}{\sqrt{\sum\limits_{u \in U_{ij}} (r_{ui} - b_{ui})^2} \cdot \sqrt{\sum\limits_{u \in U_{ij}} (r_{uj} - b_{uj})^2}}$$

  Since many ratings are unknown, some items may share only a handful of common observed raters. The empirical correlation coefficient, is based only on the common user support. Similarities based on a greater user support are more reliable.

KNN Baseline

The **shrunk Pearson-baseline correlation coefficient** is defined as:

$$\text{pearson\_baseline\_shrunk\_sim}(i,j) = \frac{|U_{ij}| - 1}{|U_{ij}| - 1 + \text{shrinkage}} \cdot \hat{\rho}_{ij}$$

Uij : the set of all users that have rated both items i and j.

shrinkage parameter helps to avoid overfitting when only few ratings are available

Using the similarity measure, we identify the k items rated by u, which are most similar to i. This set of k neighbors is denoted by $\mathbf{N_u^k(i)}$

The predicted value of $\mathbf{r_{ui}}$ is taken as a weighted average of the ratings of neighboring items, while adjusting for user and item effects through the baseline estimates

$$\hat{r}_{ui} = b_{ui} + \frac{\sum\limits_{j \in N_u^k(i)} \text{sim}(i,j) \cdot (r_{uj} - b_{uj})}{\sum\limits_{j \in N_u^k(j)} \text{sim}(i,j)}$$

kidaptive

Recommendation Algorithms (3/3)

- Latent Factor Models

  Latent factor models comprise an alternative approach to collaborative filtering with the more holistic goal to uncover latent features that explain observed ratings.

  A typical model associates each user u with a user-factors vector $\mathbf{p_u} \in \mathbf{R}^{\mathbf{f}}$, and each item i with an item-factors vector $\mathbf{q_i} \in \mathbf{R}^{\mathbf{f}}$.

  The prediction is done by taking an inner product:

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T p_u$$

  To estimate all the unknown, the following regularized squared error is minimized:

$$\sum_{r_{ui} \in R_{train}} (r_{ui} - \hat{r}_{ui})^2 + \lambda \left(b_i^2 + b_u^2 + ||q_i||^2 + ||p_u||^2 \right)$$

SVD algorithm

The minimization is performed by a very straightforward stochastic gradient descent:

$$
\begin{aligned}
b_u &\leftarrow b_u + \gamma(e_{ui} - \lambda b_u) \\
b_i &\leftarrow b_i + \gamma(e_{ui} - \lambda b_i) \\
p_u &\leftarrow p_u + \gamma(e_{ui} \cdot q_i - \lambda p_u) \\
q_i &\leftarrow q_i + \gamma(e_{ui} \cdot p_u - \lambda q_i)
\end{aligned}
$$

where $e_{ui} = r_{ui} - \hat{r}_{ui}$

These steps are performed over all the ratings of the trainset and repeated n_epochs times.

Baselines are initialized to 0. User and item factors are randomly initialized according to a normal distribution

SVD++ algorithm

The SVD++ algorithm, an extension of SVD taking into account implicit ratings.

The prediction rating is set as:

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T \left(p_u + |I_u|^{-\frac{1}{2}} \sum_{j \in I_u} y_j \right)$$

Where

$\mathbf{I_u}$: the set of all items rated by user u

$\mathbf{y_j}$ terms are a new set of item factors that capture implicit ratings.

An implicit rating describes the fact that a user u rated an item j, regardless of the rating value.

Hybrid Recommendation Algorithms

Baseline_Only, Knn_UserBased_Baseline, Knn_ItemBased_Baseline, SVD_biased and SVDpp_biased algorithms were used to generate recommenders for last 5 items held out for 9813 users.

These generated recommendations were used as features of the following hybrid recommenders:

- Elastic Net Linear Regression
- Gradient Boosting Regressor

Hybrid Recommendation Algorithms (1/2)

- Elastic Net Linear Regression

  Elastic Net is a linear regression model trained with L1 and L2 prior as regularizer. This combination allows for learning a sparse model where few of the weights are non-zero like Lasso, while still maintaining the regularization properties of Ridge.

  The objective function to minimize is

$$\min_{w} \frac{1}{2n_{samples}}||Xw - y||_2^2 + \alpha\rho||w||_1 + \frac{\alpha(1-\rho)}{2}||w||_2^2$$

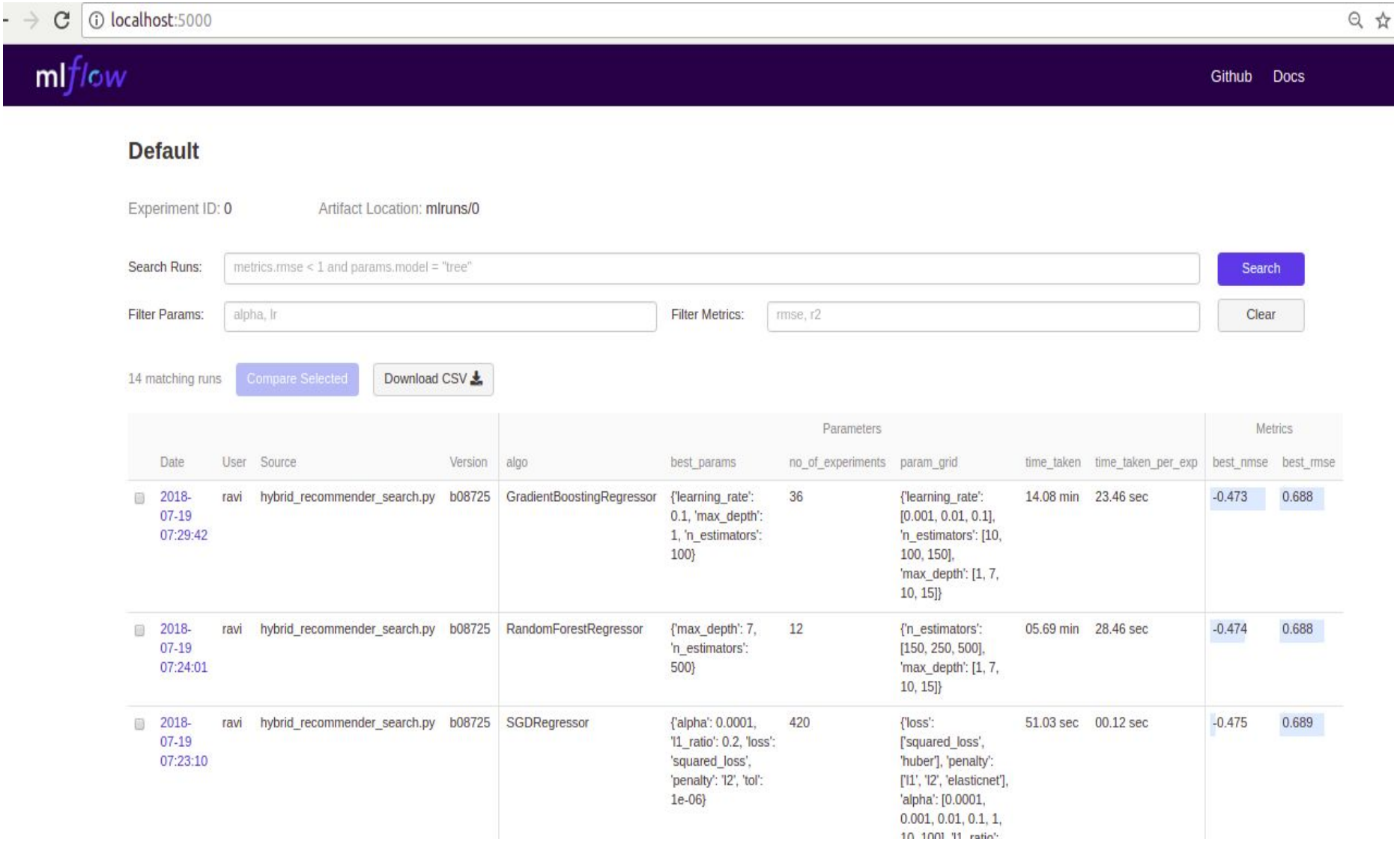
Hybrid Recommendation Algorithms (2/2)

- Gradient Boosting Regressor

  Gradient Boosted Regression Trees (GBRT) is a generalization of boosting to arbitrary differentiable loss functions.

  The number of weak learners (i.e. regression trees) is controlled by the parameter **n_estimators**; The size of each tree can be controlled either by setting the tree depth via **max_depth**. The **learning_rate** is a hyper-parameter in the range (0.0, 1.0] that controls overfitting

H2O Flow Tracking


localhost:5000
mlflow
Github
Docs
Default
Experiment ID: 0
Artifact Location: mlruns/0
Search Runs:
metrics.rmse < 1 and params.model = "tree"
Search
Filter Params:
alpha, lr
Filter Metrics:
rmse, r2
Clear
14 matching runs
Compare Selected
Download CSV
Parameters
Metrics
Date
User
Source
Version
algo
best_params
no_of_experiments
param_grid
time_taken
time_taken_per_exp
best_nmse
best_rmse
2018-07-19 07:29:42
ravi
hybrid_recommender_search.py
b08725
GradientBoostingRegressor
{'learning_rate': 0.1, 'max_depth': 1, 'n_estimators': 100}
36
{'learning_rate': [0.001, 0.01, 0.1], 'n_estimators': [10, 100, 150], 'max_depth': [1, 7, 10, 15]}
14.08 min
23.46 sec
-0.473
0.688
2018-07-19 07:24:01
ravi
hybrid_recommender_search.py
b08725
RandomForestRegressor
{'max_depth': 7, 'n_estimators': 500}
12
{'n_estimators': [150, 250, 500], 'max_depth': [1, 7, 10, 15]}
05.69 min
28.46 sec
-0.474
0.688
2018-07-19 07:23:10
ravi
hybrid_recommender_search.py
b08725
SGDRegressor
{'alpha': 0.0001, 'l1_ratio': 0.2, 'loss': 'squared_loss', 'penalty': 'l2', 'tol': 1e-06}
420
{'loss': ['squared_loss', 'huber'], 'penalty': ['l1', 'l2', 'elasticnet'], 'alpha': [0.0001, 0.001, 0.01, 0.1, 1,
51.03 sec
00.12 sec
-0.475
0.689

Results

Recommenders

Algorithm	AVG RMSE of 10-fold Validation Data	AVG RMSE of 10-fold Testing Data
Knn_UserBased_Baseline_SGD_Tuned	0.6909	0.6905
BaselineOnly_SGD_Tuned	0.6944	0.6939
Knn_ItemBased_Baseline_SGD_Tuned	0.6943	0.6940
Knn_UserBased_Baseline_ALS_Tuned	0.7019	0.7018
BaselineOnly_ALS_Tuned	0.7036	0.7036
Knn_ItemBased_Baseline_ALS_Tuned	0.7035	0.7036
SVDpp_biased_Tuned	0.7067	0.7066
SVD_biased_Tuned	0.7076	0.7075
SVD_unbiased_Tuned	0.7580	0.7538

kidaptive

Hybrid Recommenders

Meta Learner	AVG RMSE of 10-fold Testing Data
GradientBoostingRegressor	0.6866
RandomForestRegressor	0.6869
ElasticNet	0.6880
LinearRegression	0.6880

kidaptive

Comparison with existing recommender

- In order to compare with existing recommender in production, Anti-Test set(User/Item Pairs not in input event data) was generated.
- Individual recommender algorithms and hybrid recommenders were used to predict ratings for user/item pairs in anti-test set and top 50 recommendations were generated for each user.

Recommender	RMSE
hybrid_Gradient Boosting Regressor	0.6291
hybrid_Elastic Net Regressor	0.6300
Knn_UserBased_Baseline_SGD	0.6378
BaselineOnly_SGD	0.6385
Knn_ItemBased_Baseline_SGD	0.6437
current	**0.6752**
SVD ++	0.6875
SVD	0.6877

kidaptive

Enhancements

1. Current recommenders used only latest rating, leverage multiple rating events, analyse whether time duration spent on video impacts change in rating.
2. Use other metadata of events like difficult_rating, playlist/channel information
3. Aggregation of events data to analyse no of repeated access in addition to time spent, to associate confidence value for rating
4. Incorporate Temporal Dynamics effect on user, item biases and user preferences to model rating as function of time
5. Average time between events to get heuristics of engagement.

kidaptive

Personalized, Playful, Lifelong Learning