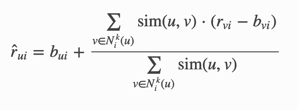
**Process (BaselineOnly and KNNBaseline):**

1. Obtain User x Movie matrix (R)
2. Calculate baselines for users and movies
   1. Initialize as 0
   2. Calculate average
   3. Use these averages to predict all ratings:
   4. Minimize the least square error:

      2. (to avoid overfitting)
      3. Alternating Least Squared (ALS)
         1. Hold constant
         2. Calculate for each movie using
         3. Hold constant
         4. Calculate for each user using
         5. Repeat
      4. Stochastic Gradient Descent (SGD)
         1. repeat
3. Apply algorithm
   1. BaselineOnly:
   2. KNNBaseline
      1. User based
         1. Calculate similarity between users: Cosine, MSD, Pearson, Pearson Baseline (just vector multiplication, no iteration or anything complicated)
         2. 
      2. Item based – same as above but using item similarity

*Note that you can use Pearson Baseline with all KNN basic, KNN with means, and KNN with Z-score, which would mean calculating baselines first, which of course could also be part of our experiment.*

**Process (SVD):**

1. Obtain User x Movie matrix (R)
2. Calculate baselines for users and movies – same as above
3. Compute the SVD of the User x Movie matrix:
   1. Aldous-Hoover Factorization
   2. Use the SVD and the baseline to predict all ratings:
   3. Minimize the least squares:

      2. to avoid overfitting
      3. Use Stochastic Gradient Descent (SGD)

SVDpp:

**Alternating Least Squares (ALS) for baseline calculation:**1: use gradient descent to update user bias

2: use gradient descent to update movie bias

3: repeat

**Alternating Least Squares (ALS) for matrix factorization:**1: use gradient descent to update all the latent user factors

2: use gradient descent to update latent movie factors

3: repeat

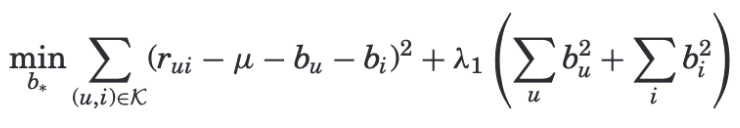
**Stochastic Gradient Descent (SGD) for baseline calculation:**

1. update one user-movie baseline at a time

**Stochastic Gradient Descent (SGD) for matrix factorization:**1: update one latent user-movie factor at a time

**Baselines: https://github.com/polymathnexus5/rec-engine-CS205L-W19/blob/master/references/a1-koren.pdf**

1: we can simply use existing data to get a baseline i.e. mean but that’s not super userfule so

2: typically we ESTIMATE the baseline for a user or item by minimizing the RMSE IF we were to use JUST the baseline as the model i.e. 

Where the lambda term avoids overfitting

Regularization Parameter(s) (lambda(s)):

Purpose – reduce overfitting

In alternating least squares they determined by hand fine tuning

In SGD they are optimized

**Similarity Measures: https://surprise.readthedocs.io/en/stable/similarities.html**

Cosine

MSD = mean squared difference

Pearson

Pearson baseline

**Methods for optimizing:**

ALS = alternating least squares

SGD = stochastic gradient descent

**Accuracy Measures: https://surprise.readthedocs.io/en/stable/accuracy.html**

RMSE = root mean squared error

MAE = mean absolute error

FCP = fraction of concordant pairs

**Recommendation Algorithms:**

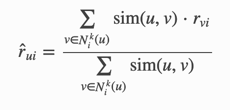
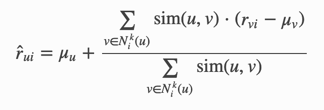
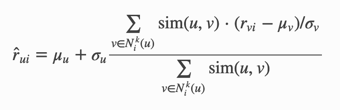
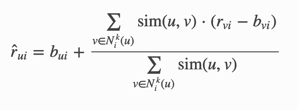
https://surprise.readthedocs.io/en/stable/basic\_algorithms.html

1. Normal Predictor: assumes normal distribution, predicts rating using Maximum Liklihood EstimatorA screenshot of a cell phone

   Description automatically generated
2. Baseline Only: predicts unknown ratings as the baseline for that user / itemA picture containing furniture

   Description automatically generated

https://surprise.readthedocs.io/en/stable/knn\_inspired.html

1. KNN Basic: 
2. KNN with Means: 
3. KNN with Zscore: 
4. KNN with Baseline: 

https://surprise.readthedocs.io/en/stable/matrix\_factorization.html

1. SVD: A screenshot of a cell phone

   Description automatically generatedA screenshot of a cell phone

   Description automatically generated
2. SVD++: Same as SVD
3. NMF: Non-negative Matrix Factorization, similar to SVD except

<https://surprise.readthedocs.io/en/stable/slope_one.html>

https://arxiv.org/abs/cs/0702144

1. SlopeOne: A screenshot of a cell phone

   Description automatically generated

Algorithms Code: https://github.com/NicolasHug/Surprise/tree/711fb80748140c44e0ed870e573c735307e6c3cc/surprise/prediction\_algorithms

**Notation: https://surprise.readthedocs.io/en/stable/notation\_standards.html#koren-2010**

R : the set of all ratings.

R\_{train}, R\_{test} and \hat{R} denote the training set, the test set, and the set of predicted ratings.

U : the set of all users. u and v denotes users.

I : the set of all items. i and j denotes items.

U\_i : the set of all users that have rated item i.

U\_{ij} : the set of all users that have rated both items i and j.

I\_u : the set of all items rated by user u.

I\_{uv} : the set of all items rated by both users u and v.

r\_{ui} : the true rating of user u for item i.

\hat{r}\_{ui} : the estimated rating of user u for item i.

b\_{ui} : the baseline rating of user u for item i.

\mu : the mean of all ratings.

\mu\_u : the mean of all ratings given by user u.

\mu\_i : the mean of all ratings given to item i.

\sigma\_u : the standard deviation of all ratings given by user u.

\sigma\_i : the standard deviation of all ratings given to item i.

N\_i^k(u) : the k nearest neighbors of user u that have rated item i. This set is computed using a [:mod:`similarity metric <surprise.similarities>`](https://github.com/NicolasHug/Surprise/blob/711fb80748140c44e0ed870e573c735307e6c3cc/doc/source/notation_standards.rst#id1).

N\_u^k(i) : the k nearest neighbors of item i that are rated by user u. This set is computed using a [:py:mod:`similarity metric <surprise.similarities>`](https://github.com/NicolasHug/Surprise/blob/711fb80748140c44e0ed870e573c735307e6c3cc/doc/source/notation_standards.rst#id3).