Formula for user and item we are calculating the score value

```python
import numpy as np

def predict(ratings, similarity, type='user'):
    if type == 'user':
        mean_user_rating = ratings.mean(axis=1)
        # we use np.newaxis so that mean_user_rating has the same format as ratings
        ratings_diff = (ratings - mean_user_rating[:, np.newaxis])
        pred = mean_user_rating[:, np.newaxis] + similarity.dot(ratings_diff) / np.array([np.abs(similarity).sum(axis=1)]).T
    elif type == 'item':
        pred = ratings.dot(similarity) / np.array([np.abs(similarity).sum(axis=1)])
    return pred
```

This method is used to predict ratings that a user might give to items based on the ratings given by similar users or the similarities between items. Let's break down how this function works in detail:

Function Parameters

ratings: A matrix where rows represent users and columns represent items. The values in the matrix are the ratings given by users to items.

similarity: A matrix that represents the similarity between users (if type='user') or items (if type='item'). Similarity can be computed using various methods like cosine similarity or Pearson correlation.

type: A string that determines whether the prediction is based on user-user similarity ('user') or item-item similarity ('item').

User-Based Collaborative Filtering (type='user')

Compute Mean User Ratings:

mean_user_rating = ratings.mean(axis=1)

This calculates the average rating for each user.

Normalize Ratings by Subtracting Mean User Ratings:

ratings_diff = (ratings - mean_user_rating[:, np.newaxis])

Each user's ratings are normalized by subtracting their mean rating. The np.newaxis ensures that mean_user_rating has the same dimensions as ratings.

Compute Predicted Ratings:

pred = mean_user_rating[:, np.newaxis] + similarity.dot(ratings_diff) / np.array([np.abs(similarity).sum(axis=1)]).T

This predicts the ratings by adding the mean user ratings back to the weighted sum of the normalized ratings. The weights are derived from the similarity matrix. The normalization term np.array([np.abs(similarity).sum(axis=1)]).T ensures that the weighted sum is properly scaled.