Group 10: Restricted Boltzmann Machines

Introduction on problem and technique

Recommendation systems is an important information filter used in a variety of areas. It is commonly used for playlist generators for video and music services, product recommenders for online stores, or content recommenders for social media platforms and open web content recommenders. Therefore, we are interested in the system and trying to create our own version.

We use packages like BernoulliRBM, pipeline from sklearn to generate a Restricted Boltzmann Machines model and modules such as numpy, pandas and matplotlib to visualize and analyze our dataset and result quantitatively.

In addition, we not only used the processed datasets provided by instructor from Canvas, but also adapted some of the original datasets from Kaggle: MovieLens 20M Dataset

Read Data and visualization

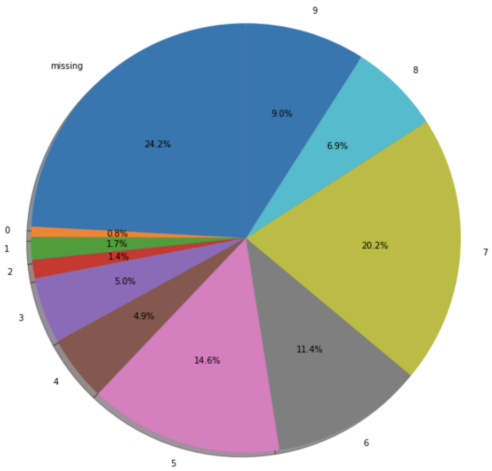
```
with open('mlens-small/top-names.txt') as f: names = f.read().split('\n')

ratings = np.loadtxt('mlens-small/top-ratings-missing.txt')

nUsers,nMovies = ratings.shape

X = (ratings >= 7).astype(int);  # originally set movies ratings over 7 as recommandated
```

We read data from the dataset, and did a little visualization to know a basic distribution of the dataset.

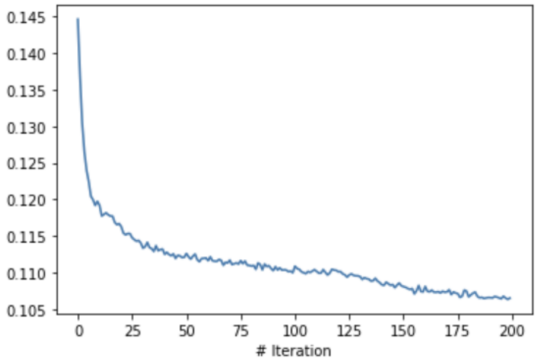


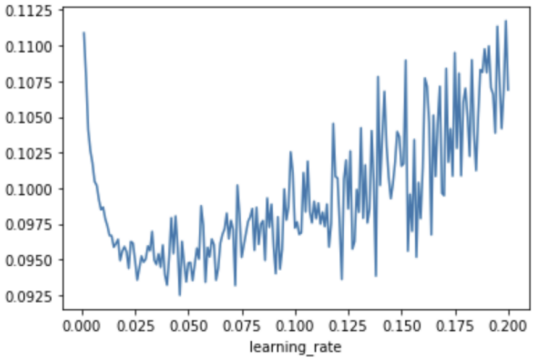
This Pie Chart shows that from the raw data, we have a lot of missing ratings, and ratings over 5 share a big portion. According to this distribution, setting ratings over 7 as recommended movies is a reasonable threshold.

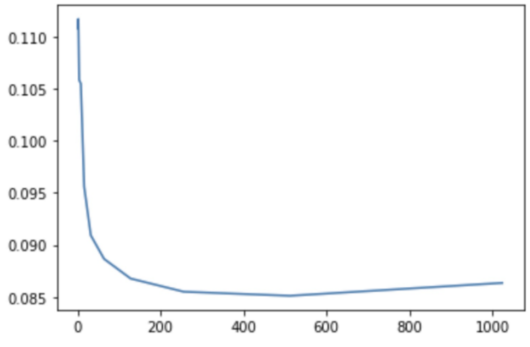
Training

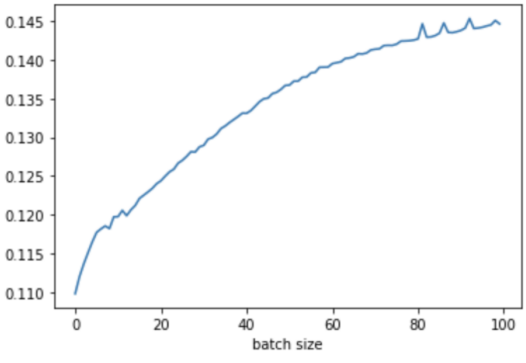
Since RBM is a generative model, we evaluate it by dropping 30% of ratings in the validation data, then use Gibbs sampling to generate dropped ratings several times to get a mean number so that we can achieve a "predict" result. During the training process of RBM, we face four decisions for hyperparameters. Therefore, we want to know how these four hyperparameters will affect the result. We did cross validation when training.

- In conclusion, the best learning rate occurs in the range from 0.025 to 0.05.
- When iteration increases, the results get better when the iteration number is less to 200. However, it may cause overfitting.
- In our test, the best component number is around 400.
- When batch size increases, the mean squared error increases too which still makes me confused.









Result

After tuning hyperparameters, we systematically evaluate the performance on the movie dataset. We choose components: 512, learning_rate = 0.006, batch_size =10 and iteration = 200 as our hyperparameters. The *pseudo-likelihood* of this model estimated by applying Monte Carlo is: -216.58

Our error rate is about 0.282. As a movie recommender, I believe precision score is more important than recall score. Since customers will browse the recommended list quickly, as long as a few movies are interesting to customers, customers will believe that the recommendation system is useful. Therefore, we should pursue a precision score as best as possible.

$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN}$$

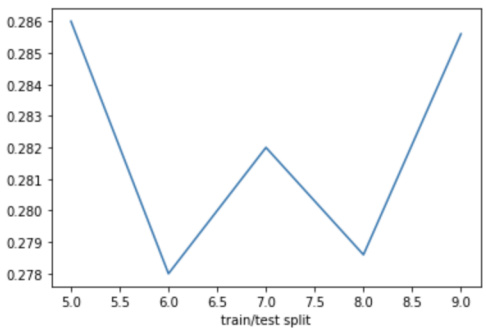
ROC AUC	AUC	Recall	Precision	Mean Squared Error (MSE)
0.907	0.915	0.882	0.876	0.085

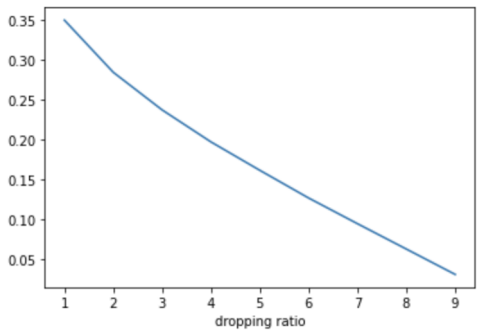
Train/Test split ratio on error rate

We also trained our model under a different Train/Test split ratio. From the plot below, it clearly shows that 5.0 to 6.0 is a range of underfitting and 8.0 to 9.0 is a range of overfitting. (See train/test split figure)

Dropping rate on Test data

Furtherly, the dropping rate when we create our test data also reflects the error rate. When we have more information, for example a user rates more movies, the prediction will be more accurate. This is reasonable. (See dropping rate figure)





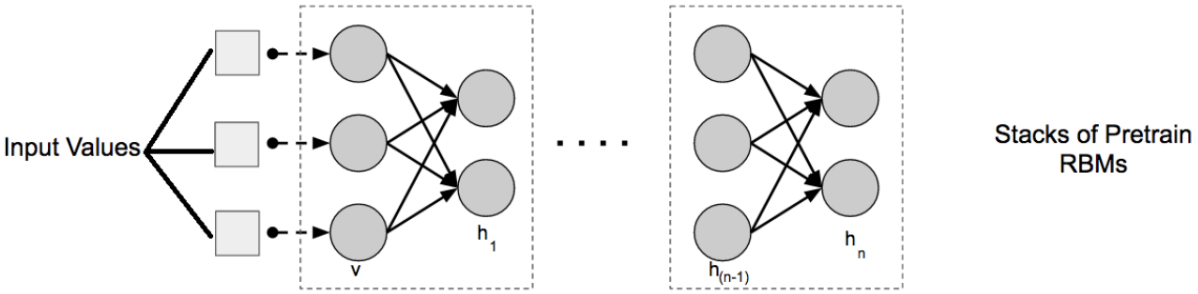
Since Restricted Boltzmann Machine is computing conditional probabilities. For example, when we sample from RBM, we are sampling Hidden layer H given Visible X. I this sklearn.neural_network.BernoulliRBM.transform method to Compute the hidden layer activation probabilities, P(h=1|v=X). I found that when one hidden layer is activated, there exists some kind of relationship between the test data and the probabilities from the transform function. I guess that when a hidden layer is activated it means that "a" user likes some movies more likely and others to be less likely. This high probability means high similarities in some aspects for some movies.

Resources

Sklearn documentation helps a lot. Meanwhile, this github repository illustrates many aspects of Restricted Boltzmann Machine: https://github.com/yell/boltzmann-machines.

One more interesting point

When I was researching the Restricted Boltzmann Machine, I found that RBM can be mixed with other classification models such as logistic regression, SVM and so on. This stackflow post given some insight:
https://stackoverflow.com/questions/52166308/stacking-rbms-to-create-deep-belief-network-in-sklearn
This post shows that, a deep belief network constructs like this:



By using logistic regression in the final step of the pipeline(RBM) to evaluate if the image from MNIST transformations by an RBM or by a stack of RBMs/a DBN improves classification. Here is some comparison given in the post. This post makes me more interested in this RBM/DBM model. I also tried to adapt this post to predict something interesting but I didn't succeed in that. However, it is obvious that a stack of RBM can improve performance.

Logistic regression by itself:

```
Model performance:
             precision    recall  f1-score   support

        0.0       0.95      0.97      0.96       995
        1.0       0.96      0.98      0.97      1121
        2.0       0.91      0.90      0.90      1015
        3.0       0.90      0.89      0.89      1033
        4.0       0.93      0.92      0.92       976
        5.0       0.90      0.88      0.89       884
        6.0       0.94      0.94      0.94       999
        7.0       0.92      0.93      0.93      1034
        8.0       0.89      0.87      0.88       923
        9.0       0.89      0.90      0.89      1020

avg / total       0.92      0.92      0.92     10000
```

Logistic regression on outputs of an RBM:

```
Model performance:
             precision    recall  f1-score   support

        0.0       0.98      0.98      0.98       995
        1.0       0.98      0.99      0.99      1121
        2.0       0.95      0.97      0.96      1015
        3.0       0.97      0.96      0.96      1033
        4.0       0.98      0.97      0.97       976
        5.0       0.97      0.96      0.96       884
        6.0       0.98      0.98      0.98       999
        7.0       0.96      0.97      0.97      1034
        8.0       0.96      0.94      0.95       923
        9.0       0.96      0.96      0.96      1020

avg / total       0.97      0.97      0.97     10000
```

Logistic regression on outputs of a stacks of RBMs / a DBN:

```
Model performance:
             precision    recall  f1-score   support

        0.0       0.99      0.99      0.99       995
        1.0       0.99      0.99      0.99      1121
        2.0       0.97      0.98      0.98      1015
        3.0       0.98      0.97      0.97      1033
        4.0       0.98      0.97      0.98       976
        5.0       0.96      0.97      0.97       884
        6.0       0.99      0.98      0.98       999
        7.0       0.98      0.98      0.98      1034
        8.0       0.98      0.97      0.97       923
        9.0       0.96      0.97      0.96      1020

avg / total       0.98      0.98      0.98     10000
```