TransparentRec

November 3, 2015

In [1]:
```
import numpy as np
import UserResults as ur
import UserMatrix as um
from Classifiers import TransparentRidge
```

In [2]:
```
# Loading user matrix
user_id = 944
user_matrix = um.get_user_matrix(user_id)
```

In [3]:
```
# Loading user ratings and movie list
ratings = np.genfromtxt('postprocessed-data/user_ratings', delimiter=',', dtype=int)
movies = np.genfromtxt('postprocessed-data/movie_list', delimiter='|', dtype=str)
user_ratings = ratings[user_id-1]
```

In [4]:
```
# Creating model
clf =TransparentRidge(alpha=0.001)
user_cols = user_matrix.shape[1]
data = user_matrix[:, 1:(user_cols-1)]
target = user_matrix[:, (user_cols-1)]
clf.fit(data,target)
weights = clf.coef_
neg_evi, pos_evi = clf.predict_evidences(data)
bias = clf.get_bias()
y_pred = clf.predict(data)
indices = np.argsort(y_pred)
```

In [5]:
```
# The Highest Rating
j = indices[-1]
movie_id = user_matrix[j][0]
res = um.get_avg_rating_for_movie(ratings, movie_id-1)
avg_rating = res[0]
num_rating = res[1]
movie_features = ur.gen_movie_weights(movie_id,weights,user_matrix)
print "Movie Title: ", movies[movie_id-1]
print "Average Rating: ", avg_rating
print "Number of Ratings: ", num_rating
print "Prediction: ",  clf.predict(data[j])[0]
print "Bias and evidences:", bias, neg_evi[j], pos_evi[j]
print "Positive Features"
print movie_features[0].head(10)
print "Negative Features"
print movie_features[1].head(10)
```

```
Movie Title:  Star Wars (1977)
Average Rating:  4.3595890411
```

```
Number of Ratings:  584
Prediction:  5.00002788598
Bias and evidences: 3.62444155016 -2.19407376566 3.56966010147
Positive Features
              Feature  Weights
0              Sci-Fi   0.7745
1              Action   0.3451
2     death-of-friend   0.2353
3       strangulation   0.1284
4              escape   0.1069
5               sword   0.1049
6  Ford, Harrison (I)   0.1008
7   Jones, James Earl   0.0921
8              combat   0.0769
9    good-versus-evil   0.0713
Negative Features
              Feature  Weights
0           Adventure  -0.9270
1             Romance  -0.5481
2              hitman  -0.2780
3                duel  -0.1657
4            computer  -0.1451
5  mixed-martial-arts  -0.1213
6                 bar  -0.0913
7      Average Rating  -0.0844
8             soldier  -0.0806
9            disguise  -0.0777
```

In [6]:
```python
# The Lowest Rating
j = indices[0]
movie_id = user_matrix[j][0]
res = um.get_avg_rating_for_movie(ratings, movie_id-1)
avg_rating = res[0]
num_rating = res[1]
movie_features = ur.gen_movie_weights(movie_id,weights,user_matrix)
print "Movie Title: ", movies[movie_id-1]
print "Average Rating: ", avg_rating
print "Number of Ratings: ", num_rating
print "Prediction: ",  clf.predict(data[j])[0]
print "Bias and evidences:", bias, neg_evi[j], pos_evi[j]
print "Positive Features"
print movie_features[0].head(10)
print "Negative Features"
print movie_features[1].head(10)
```

```
Movie Title:  Beverly Hillbillies, The (1993)
Average Rating:  1.93333333333
Number of Ratings:  15
Prediction:  1.00005299666
Bias and evidences: 3.62444155016 -2.7331792002 0.1087906467
Positive Features
                        Feature  Weights
0                        Comedy   0.5027
1                Average Rating   0.1886
2                        remake   0.1178
```

```
3                   helicopter   0.0437
4  lifting-someone-into-the-air   0.0343
5                       school   0.0213
Negative Features
                   Feature  Weights
0                  shotgun  -0.1656
1  title-directed-by-female  -0.0774
2            swimming-pool  -0.0689
```

```
In [7]: # The case that has the most negative evidence, regardless of positive evidence
        j = np.argsort(neg_evi)[0]
        movie_id = user_matrix[j][0]
        res = um.get_avg_rating_for_movie(ratings, movie_id-1)
        avg_rating = res[0]
        num_rating = res[1]
        movie_features = ur.gen_movie_weights(movie_id,weights,user_matrix)
        print "Movie Title: ", movies[movie_id-1]
        print "Average Rating: ", avg_rating
        print "Number of Ratings: ", num_rating
        print "Prediction: ",  clf.predict(data[j])[0]
        print "Bias and evidences:", bias, neg_evi[j], pos_evi[j]
        print "Positive Features"
        print movie_features[0].head(10)
        print "Negative Features"
        print movie_features[1].head(10)
```

```
Movie Title:  Batman Forever (1995)
Average Rating:  2.66086956522
Number of Ratings:  115
Prediction:  1.99997056937
Bias and evidences: 3.62444155016 -5.13228379437 3.50781281358
Positive Features
         Feature  Weights
0         Comedy   0.5027
1         Action   0.3451
2           kiss   0.2093
3          blood   0.1307
4  blood-splatter   0.1101
5         escape   0.1069
6  Average Rating   0.1067
7    car-accident   0.1007
8      fistfight   0.0941
9      flashback   0.0936
Negative Features
                 Feature  Weights
0              Adventure  -0.9270
1              obsession  -0.2090
2               violence  -0.1470
3                 orphan  -0.1143
4    semiautomatic-pistol  -0.0987
5                   love  -0.0972
6            one-man-army  -0.0895
7  character-name-in-title  -0.0814
8                   bomb  -0.0790
9                  brawl  -0.0707
```

```
In [8]: # The case that has the most positive evidence, regardless of negative evidence
        j = np.argsort(pos_evi)[-1]
        movie_id = user_matrix[j][0]
        res = um.get_avg_rating_for_movie(ratings, movie_id-1)
        avg_rating = res[0]
        num_rating = res[1]
        movie_features = ur.gen_movie_weights(movie_id,weights,user_matrix)
        print "Movie Title: ", movies[movie_id-1]
        print "Average Rating: ", avg_rating
        print "Number of Ratings: ", num_rating
        print "Prediction: ",  clf.predict(data[j])[0]
        print "Bias and evidences:", bias, neg_evi[j], pos_evi[j]
        print "Positive Features"
        print movie_features[0].head(10)
        print "Negative Features"
        print movie_features[1].head(10)
```

```
Movie Title:  Terminator, The (1984)
Average Rating:  3.93377483444
Number of Ratings:  302
Prediction:  3.99996506368
Bias and evidences: 3.62444155016 -3.36404071618 3.7395642297
Positive Features
                  Feature  Weights
0                  Sci-Fi   0.7745
1                  Action   0.3451
2         death-of-friend   0.2353
3                    kiss   0.2093
4              photograph   0.1907
5                Thriller   0.1639
6          police-station   0.1312
7                   blood   0.1307
8          blood-splatter   0.1101
9  Schwarzenegger, Arnold   0.1087
Negative Features
                   Feature  Weights
0                   corpse  -0.2403
1                  shotgun  -0.1656
2                 violence  -0.1470
3           police-officer  -0.1417
4        shot-in-the-chest  -0.1281
5       mixed-martial-arts  -0.1213
6             one-man-army  -0.0895
7  character-name-in-title  -0.0814
8                  soldier  -0.0806
9                    brawl  -0.0707
```

```
In [9]: # Most conflicted
        conflict = np.min([abs(neg_evi), pos_evi], axis=0)
        indices = np.argsort(conflict)
        j=indices[-1]
        movie_id = user_matrix[j][0]
        res = um.get_avg_rating_for_movie(ratings, movie_id-1)
        avg_rating = res[0]
        num_rating = res[1]
```

```
        movie_features = ur.gen_movie_weights(movie_id,weights,user_matrix)
        print "Movie Title: ", movies[movie_id-1]
        print "Average Rating: ", avg_rating
        print "Number of Ratings: ", num_rating
        print "Prediction: ",  clf.predict(data[j])[0]
        print "Bias and evidences:", bias, neg_evi[j], pos_evi[j]
        print "Positive Features"
        print movie_features[0].head(10)
        print "Negative Features"
        print movie_features[1].head(10)
```

```
Movie Title:  Batman Forever (1995)
Average Rating:  2.66086956522
Number of Ratings:  115
Prediction:  1.99997056937
Bias and evidences: 3.62444155016 -5.13228379437 3.50781281358
Positive Features
          Feature  Weights
0          Comedy   0.5027
1          Action   0.3451
2            kiss   0.2093
3           blood   0.1307
4  blood-splatter   0.1101
5          escape   0.1069
6  Average Rating   0.1067
7    car-accident   0.1007
8       fistfight   0.0941
9       flashback   0.0936
Negative Features
                   Feature  Weights
0                Adventure  -0.9270
1                obsession  -0.2090
2                 violence  -0.1470
3                   orphan  -0.1143
4     semiautomatic-pistol  -0.0987
5                     love  -0.0972
6             one-man-army  -0.0895
7  character-name-in-title  -0.0814
8                     bomb  -0.0790
9                    brawl  -0.0707
```

```
In [10]: # Least amount of info
         information = np.max([abs(neg_evi), pos_evi], axis=0)
         indices = np.argsort(information)
         j=indices[0]
         movie_id = user_matrix[j][0]
         res = um.get_avg_rating_for_movie(ratings, movie_id-1)
         avg_rating = res[0]
         num_rating = res[1]
         movie_features = ur.gen_movie_weights(movie_id,weights,user_matrix)
         print "Movie Title: ", movies[movie_id-1]
         print "Average Rating: ", avg_rating
         print "Number of Ratings: ", num_rating
         print "Prediction: ",  clf.predict(data[j])[0]
         print "Bias and evidences:", bias, neg_evi[j], pos_evi[j]
```

```python
        print "Positive Features"
        print movie_features[0].head(10)
        print "Negative Features"
        print movie_features[1].head(10)
```

```
Movie Title:  GoodFellas (1990)
Average Rating:  3.95154185022
Number of Ratings:  227
Prediction:  3.99951166789
Bias and evidences: 3.62444155016 -0.298179669772 0.673249787496
Positive Features
Empty DataFrame
Columns: [Feature, Weights]
Index: []
Negative Features
          Feature  Weights
0  Average Rating  -0.0385
```