PSYCH 734 / CSE 734

Assignment 2: Restricted Boltzmann Machines

Sara Jamil

Introduction

This report examines the behaviour of a 1-layer Restricted Boltzmann Machine (RBM) with varying numbers of hidden units. The RBM was trained on a set of users' preference ratings on four different genres of movies: sci-fi & fantasy, comedy, horror, and drama. As a result, it should to learn hidden features that predict combinations of movies that a user will like or dislike.

The training patterns that the RBM was trained on consisted of binary ratings on 12 movies, with 3 movies for each of the 4 genre categories. By examining the input patterns, we can observe that the movies within each category tends to be rated similarly by the user, with some between-category overlap. Each user seems to prefer one of the categories over the others. The following table shows this trend by taking the sum of preferred movies in each category for all of the users.

Example	Sci-fi	Comedy	Horror	Drama
1	3	0	0	1
2	3	0	1	0
3	2	1	0	0
4	2	1	0	0
5	3	0	0	1
6	2	1	1	1
7	3	0	1	0
8	1	3	0	0
9	1	2	0	0
10	0	2	1	0
11	0	2	1	1
12	1	3	1	1
13	0	2	0	1
14	1	2	1	0
15	0	1	3	0
16	0	0	2	1
17	1	0	2	0
18	0	1	2	0
19	0	0	3	1
20	0	0	2	0
21	1	0	2	0
22	0	1	0	3
23	1	0	1	2
24	0	1	0	2
25	1	0	0	2
26	0	1	1	3
27	1	0	0	2
28	0	0	1	2

We can also observe that there are 4 groups of users that prefer one of the categories more than the rest. By dividing the users into these 4 groups we can show this trend by taking the average ratings of each category, as shown in the following table.

Group	Sci-fi	Comedy	Horror	Drama
1	2.571429	0.428571	0.428571	0.428571
2	0.571429	2.285714	0.571429	0.428571
3	0.285714	0.285714	2.285714	0.285714
4	0.428571	0.428571	0.428571	2.285714

By training the Restricted Boltzmann Machine, we hope to find that it is able to learn some of these trends in the data. Some of the following simulations will try to test what the learned hidden features are and if they correspond to these observed trends.

Methods

In order to simulate the model and examine the results of the RBM, various hidden layer sizes were used and various tests were conducted. The experiments were split up into 5 different parts as explained below. Parts 1 and 2 involve RBMs trained with various hidden layer sizes while the rest only use the network with 4 hidden units.

Part 1 – Training the RBM

The RBM was trained with one layer of weights including 12 input units and four different hidden layer sizes: 2, 4, 6, and 8 hidden units. It is important to note that there are also bias units including one for the input and one for the output. To simulate the model, the initialization script was executed and then the train function was repeatedly called until the model appears to have converged.

In order to determine when the model has converged, the difference in energy of successive runs of the train function was plotted. This gives an approximate derivative of the energy of the system. When the energy difference seems to lower, the simulation is said to have converged. Another indicator that the model has converged is that the values of all the weights in the model do not continue to change by large amounts. The weights were plotted after each run to be able to visually determine whether the model has converged.

Once the model has converged, the values of the weights were examined to determine if the different hidden units were able to capture expected clusters of the input feature triplets corresponding to movies from the same category. This will be discussed in the Discussion section below.

Part 2 – Clamping the idealized users' ratings

Once the RBM models pertaining to each hidden layer size were trained, they were tested using the idealized test input patterns. This was done to see whether the hidden layers were able to capture the general trend in the data outlined in the introduction. Each of the 4 idealized test patterns were

clamped to the input and then activated the hidden layer by calling the forward function with 1 step of brief Gibbs sampling. This was done repeatedly to examine how the hidden states change for the same input. The probability of a unit activating was also examined as discussed below.

Part 3 – Clamping the idealized hidden activations

The network trained on 4 hidden units was tested using the idealized hidden unit activation test patterns. The idealized patterns indicate one hidden unit being activated for each of the four categories. These patterns were clamped to the hidden layer and then activated the input layer by calling the backward function with 1 step of brief Gibbs sampling. This was tested repeatedly to see whether the output would approximately match any of the 4 idealized input patterns.

Part 4 – Applying brief Gibbs sampling using idealized hidden activations

Using the same idealized hidden unit test patterns as in part 3, several iterations of alternating input and hidden states were obtained with 5 steps of brief Gibbs sampling. This was implemented by using backward to generate an input state and using forward on this input state to generate another hidden state, and continue to do this for a total of 5 times until a final input state was found. The code to implement this is shown as follows. This code determines the final input state after 5 steps of brief Gibbs sampling for each of the idealized hidden unit test patterns.

```
for i = 1:4
    [iState, iProb] = backward(testHiddenPatterns(:,i), weights);
    for k = 1:5
        [hState, hProb] = forward(iState, weights);
        [iState, iProb] = backward(hState, weights);
    end
    iStateOut(:,i,trial) = iState;
    iProbOut(:,i,trial) = iProb;
end
```

The final input state *iStateOut* was then compared to each of the training patterns used to train the RBM. The closest matching training pattern was found and the ratio of the matching units was recorded along with the index number of the closest matching training pattern. The following code implements this.

```
for i = 1:4
    for j = 1:28
        compare(:,j) = trainingPatterns(:,j) == iStateOut(:,i,trial);
    end
    [closestTrain(1,i,trial), indexTrain(1,i,trial)] = max(sum(compare));
    closestTrain(1,i,trial) = closestTrain(1,i,trial)/13; %ratio of matching bits
end
```

This was done 100 times for each hidden layer test pattern to see whether the model will continue to generate blends of input patterns or whether it will settle to a specific training pattern. The code for this section can be found in testHidden.m, along with the code used to implement Part 3.

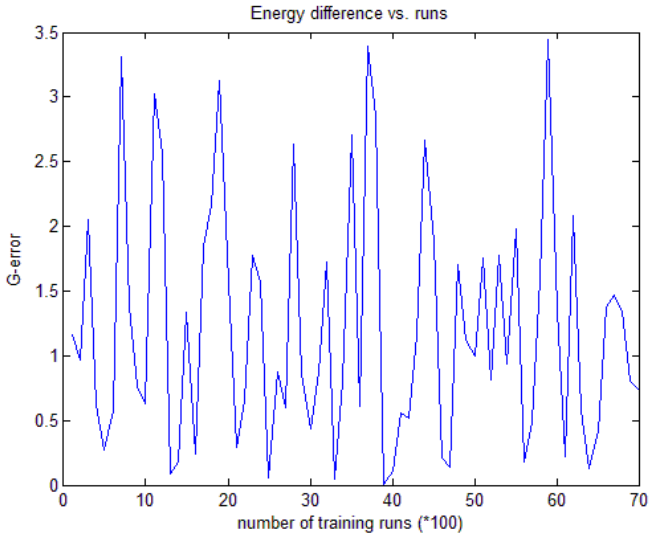
Part 5 – Applying brief Gibbs sampling using random initial state

Using the network trained on 4 hidden units, the network was given a completely random initial input state and a completely random initial hidden state. Then several steps of alternating hidden and input states were found by using the forward and backward function. Brief Gibbs sampling was applied for 10 steps and a final input layer activation pattern was generated. The code used to simulate this was similar to that of Part 4. There were two sets of final initial layer activation patterns (i.e. *iStateOut*) generated corresponding to starting from a random initial state, *I_iStateOut*, and starting from a random hidden state, *H_iStateOut*. Again, this was done for 100 repetitions and the closest matching pattern was found. The ratio of matching units and the index of the matching training pattern were also recorded. This was done to examine if the patterns generated corresponded to a specific training pattern or a blend of training patterns. The output was examined to determine if it corresponded to the idealized test input patterns and whether it was able to capture these trends. The code for this section can be found in testInput.m, along with the code used to implement Part 2.

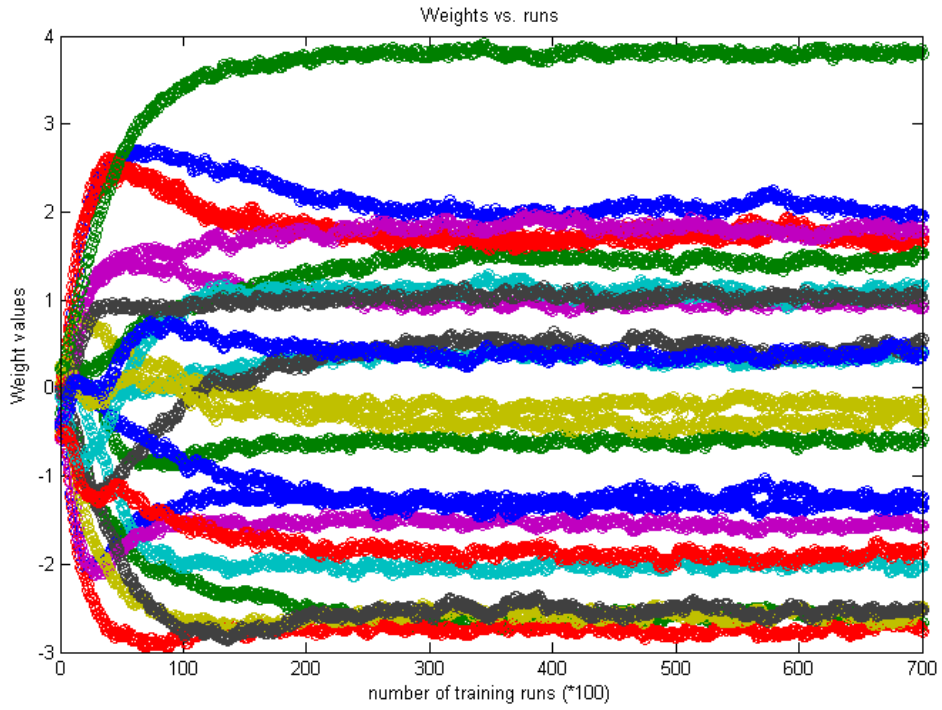
Results

Part 1 – Training the RBM

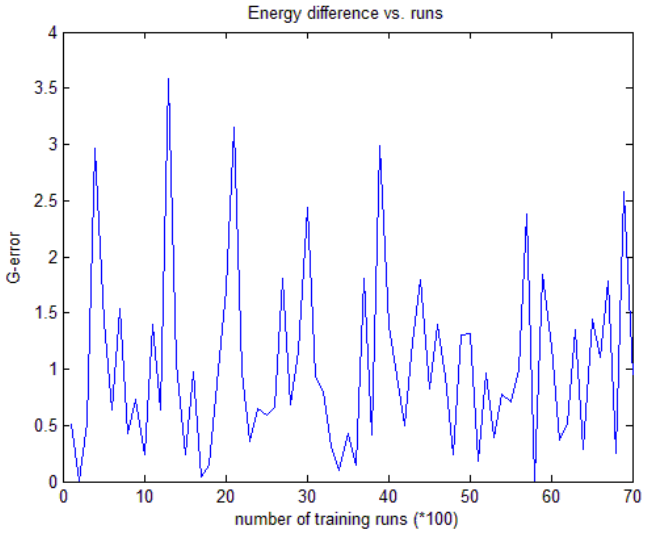
The first RBM was trained with a hidden layer size of 2. To ensure the model has converged, the train function was called 700 times where each run has 100 learning repetitions. The following plots illustrate the convergence of the model. The first figure takes the value of the G error given by the train function and takes the difference between successive runs in order to approximate the derivative of the energy difference. Then each 10 runs were averaged to show the general trend of the data. While it is not clear when the model has reached convergence from this plot, it seems that the rate of change in energy decreases in the end.

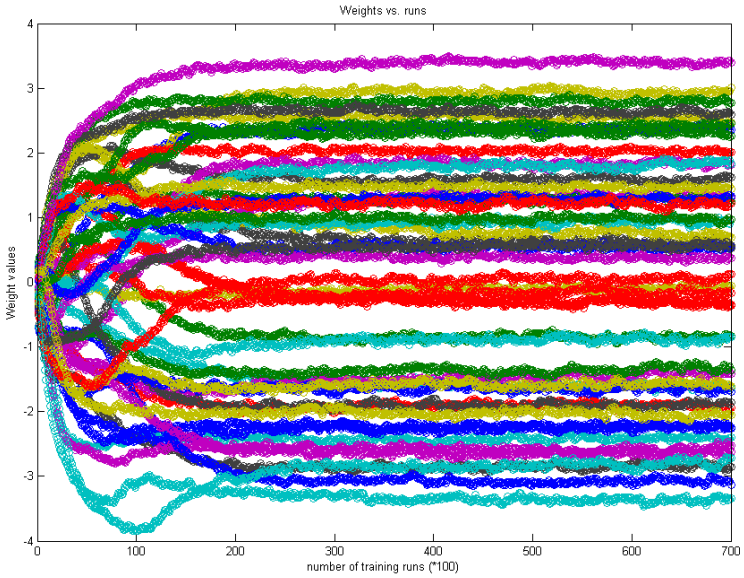


However, since this plot does not clearly show when the model has converged, the values of all the weights after each run was also plotted. This clearly shows whether the weights are continuing to change significantly or whether they have settled to some solution. The following plot shows that the value of the weights does not seem to change significantly after 200 runs, which indicates that the model has converged to some local minimum.

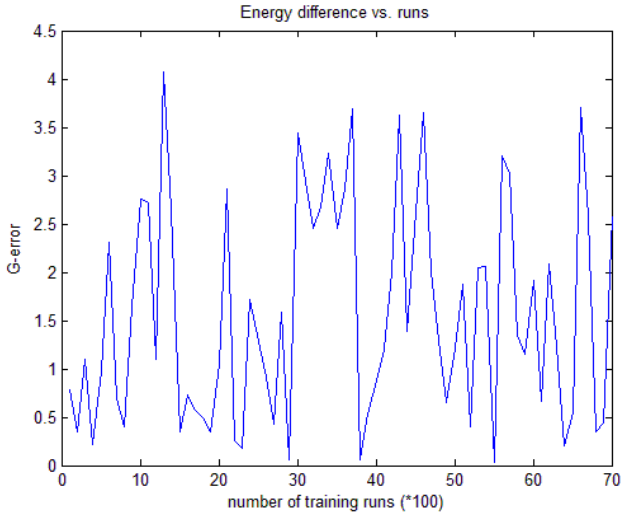


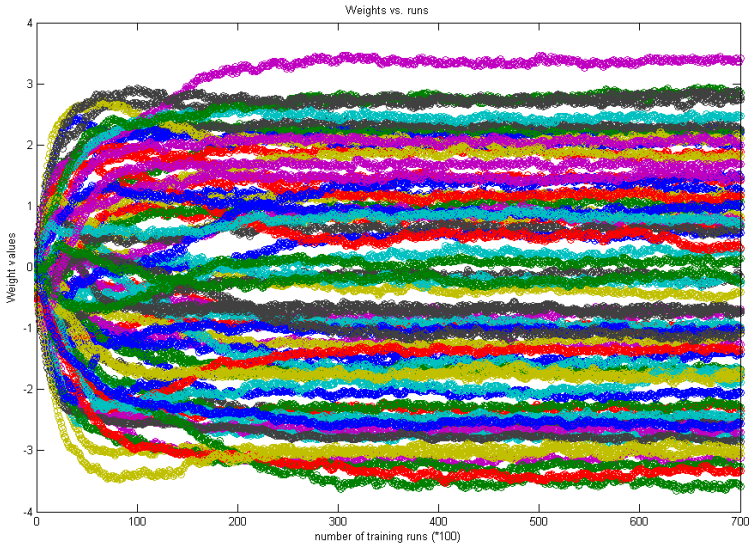
Similarly, the following plots show the results of training the 4 unit hidden layer RBM. Again, here the magnitude of the change in energy seems to steadily decrease in general. Looking at the values of the weights, the model seems to have converged after about 200 runs of train.

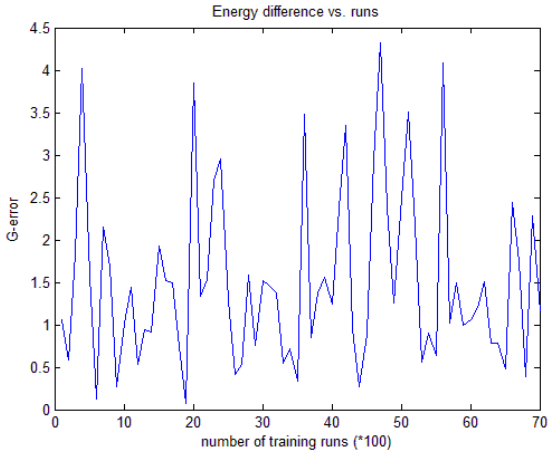


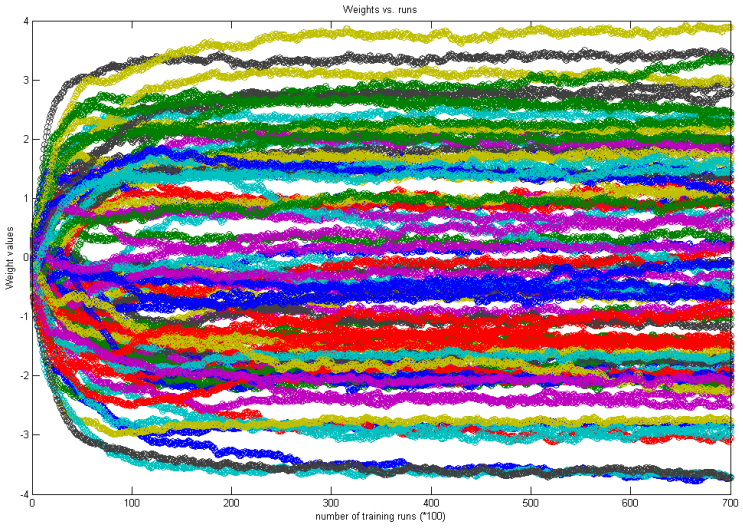


The following plots show the convergence of the 6 unit hidden layer RBM and the 8 unit hidden layer RBM, respectively. The models seem to converge after about 300 runs for the 6 hidden unit model and 400 runs for the 8 hidden unit model. Although the values of the weights continue to change slightly, their rate of change seems to decrease by the end of 700 runs.




Weights vs. runs
Weight values
number of training runs (*100)


Energy difference vs. runs
G-error
number of training runs (*100)


Weights vs. runs
Weight values
number of training runs (*100)

Part 2 – Clamping the idealized users' ratings

After clamping the idealized users' ratings for the RBM model with 2 hidden units, the results of the output hidden states are shown below. The table on the left shows the probability of the hidden units being active (according to the sigmoid function) with respect to the four movie categories: [1 1 1 0 0 0 0 0 0 0 0 0], [0 0 0 1 1 1 0 0 0 0 0 0], [0 0 0 0 0 0 1 1 1 0 0 0], and [0 0 0 0 0 0 0 0 0 1 1 1] respectively. The figure on the right shows an example of what the hidden state would look like after one run.

0.0061	0.0855	0.6200	0.9991
0.0028	0.9976	0.9895	0.3800
0.9687	0.0080	0.0094	0.0046

0	0	1	1
0	1	1	0
1	1	1	1

By looking at the probabilities of the hidden states, we can see that each of the four categories roughly outputs one of the four possible states with the 2 hidden unit RBM: [0 0], [0 1], [1 0], and [1 1]. Note that the last unit is the bias unit and can be ignored since its activation is always set to 1. However, because some of the probabilities are not close to either 1 or 0, like in the third and fourth outputs, they will sometimes output hidden states that are different from the hidden state shown above when activated by the idealized test input patterns.

The following tables show the output of the probability and states of the hidden layer with the 4 hidden unit RBM.

0.9968	0.9037	0.0011	0.9974
0.9647	0.1231	0.9814	0.3717
0.0057	0.0231	0.9853	0.9996
0.0014	0.9996	0.5703	0.4789
0.1805	0.0167	0.1204	1.0680e-04

1	1	0	1
1	0	1	0
0	0	1	1
0	1	0	0
1	1	1	1

Again, since the probabilities of the hidden states do not always end up being close to 0 or 1 when activated by the idealized test input patterns, the output of the hidden states can change with each run of the forward function. The state shown above gives an example of a likely output of the model. While there seems to be a distinct pattern that forms with respect to the four input patterns, it is possible to get variations on these patterns; however it is unlikely that two different idealized inputs will activate the same hidden state. Since more than one hidden unit can be active (i.e. it is not a winner-take-all situation) it is possible for the network to capture more features than just the idealized patterns of the four categories.

By running the test on the training patterns instead of the idealized patterns, we are able to see which training inputs are likely to activate the same hidden states. The results of this can be obtained from the *hProb_train* and *hState_train* variables in the code. We are able to notice that each of the four groups of training patterns generally activate the same 1 or 2 hidden layer patterns, but the network is able to capture other similarities within and between the groups as well. The results of examining the training pattern activations are similar in the 6 hidden unit RBM model.

The following tables show the output of the probability and states of the hidden layer with the 6 hidden unit RBM.

0.5204	0.5906	0.0051	0.9837	0	1	0	1
0.9995	4.2904e-04	0.2965	0.3941	1	0	1	0
0.9864	0.9862	0.4536	0.4762	1	1	0	0
0.0012	0.0949	0.4962	0.9993	0	0	1	1
0.0013	9.6904e-04	0.9996	0.0178	0	0	1	0
0.0529	0.9848	0.7846	0.0061	0	1	0	0
0.0441	0.3206	0.1374	0.0158	1	1	1	1

Similarly, here we see different activation patterns for the four different categories. Again, some of the units may be activated by different categories but it is unlikely that the different idealized patterns would activate the same hidden layer pattern. This model also has the capacity to represent more than just the feature of the four movie categories but it is unclear what other similarities in the training patterns the model was able to capture by just looking at the activation by the idealized input.

The following tables show the output of the probability and states of the hidden layer with the 8 hidden unit RBM.

0.0831	0.9987	0.8367	0.8586	0	1	1	1
0.5802	0.1125	0.5436	0.0710	1	0	0	1
0.9822	0.9975	7.3890e-04	0.0038	1	1	0	0
0.2175	0.0078	0.0035	0.9997	1	0	0	1
0.0522	0.3170	0.1881	0.8250	0	0	0	0
0.7902	0.3265	0.9815	0.8303	1	1	1	0
0.9997	8.9203e-04	0.4409	0.2740	1	0	1	1
0.0019	0.0265	0.9991	0.0040	0	0	1	0
0.0269	0.0638	0.0410	0.0386	1	1	1	1

The results here are similar because while the output has the capacity to capture more information than the feature of the four movie categories. When the model was tested on the training patterns instead of the idealized patterns, the four groups of users had more variation in the hidden state outputs than the previous models, which may suggest that it represents each training pattern with its own hidden layer activation.

Part 3 – Clamping the idealized hidden activations

In the 4 hidden unit RBM model, the following tables show the resulting input activation patterns when clamping the idealized hidden unit activation patterns: [1 0 0 0], [0 1 0 0], [0 0 1 0], and [0 0 0 1] respectively. The probabilities of the input activation is shown in the table on the left and an example of the input activation state is shown in the table on the right.

0.8704	0.2170	0.0842	0.0550	1	1	0	0
0.4883	0.8403	0.0148	0.0263	0	1	0	0
0.8211	0.4816	0.2048	0.0961	1	1	0	0
0.1194	0.1949	0.0149	0.6955	0	0	0	1
0.2550	0.1495	0.0901	0.8281	0	0	0	1
0.5969	0.0125	0.0702	0.6846	1	0	0	0
0.0478	0.8981	0.7415	0.1812	0	1	1	0
0.0298	0.6174	0.1060	0.3062	0	1	0	0
0.0398	0.1024	0.8836	0.4994	0	0	1	1
0.1726	0.0200	0.1008	0.0254	1	0	0	0
0.1062	0.0476	0.0961	0.0693	0	0	0	0
0.4169	0.0138	0.8627	0.0264	0	0	0	0
0.6992	0.8163	0.7804	0.6509	1	1	1	1

While some of the activations approximately match one of the 4 idealized test patterns, not all of the patterns were represented by these 4 hidden unit activations. The hidden unit activations here do not necessarily correspond to the hidden unit activations learned for each of the 4 idealized test patterns, so this may be why they are not well represented in this test. Another example of output activation is shown below.

1	0	0	0
1	1	0	0
1	1	0	0
0	0	0	1
1	0	0	1
1	0	0	0
1	1	1	0
0	1	0	0
0	0	1	1
0	0	0	0
0	0	0	0
0	0	1	0
1	1	1	1

Again, these patterns do not closely match an idealized input pattern after one step of brief Gibbs sampling. It seems to produce a blend of the different input patterns and does not seem to settle to an expected output pattern.

Part 4 – Applying brief Gibbs sampling using idealized hidden activations

Using the same hidden unit activation patterns as above, the following results show the input activation patterns generated after applying 5 steps of brief Gibbs sampling. This was done for 100 trials. The tables below show 3 different examples of input activations by the 4 hidden unit activations.

0	0	0	1	0	1	0	0	1	0	0	0
0	0	0	0	0	1	1	1	1	0	1	0
0	1	0	0	0	1	0	0	1	0	0	0
0	0	0	1	0	0	0	1	0	0	1	0
1	0	0	0	0	0	0	0	0	0	1	0
1	0	0	0	0	0	0	0	0	0	1	1
0	0	0	0	1	0	0	1	0	1	0	0
0	1	0	0	0	1	1	1	1	1	1	0
0	1	1	0	1	0	1	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	1
1	1	1	1	1	1	1	1	1	1	1	1

Some of the input activations that result resemble the idealized test input patterns. The model isn't able to consistently provide one of the four idealized patterns for each of the 4 idealized hidden layer patterns. However, after comparing them to the training patterns, they seem to closely resemble one of the training patterns. The following table shows the ratio of matching units to the training pattern input units for the examples above.

0.9231	0.8462	0.9231	0.9231	1.0000	1.0000	0.8462	0.9231	1.0000	0.9231	0.9231	0.8462

Some of the patterns exactly match one of the training patterns. After examining all of the 100 trails, it seems that they all match between 9 to 12 of the 12 input units. The following table shows the index of the closest matching training pattern.

11	18	16	3	20	7	17	17	7	15	12	13

The indices show that even though they were able to closely match one of the training patterns, each hidden unit pattern did not necessarily activate a pattern from the same group of users (that fall into the 4 movie categories). The third hidden unit activation (i.e. [0 0 1 0]) seems to more consistently activate one of the training patterns from the same group of users. This may be because this hidden unit activation most closely resembles the activation learned for one of the four categories.

Part 5 – Applying brief Gibbs sampling using random initial state

Starting from a random initial input state, 10 steps of brief Gibbs sampling were applied. The following table shows 3 of the 100 examples of final input activation states, their ratio of correct units to the closest matching training pattern, and the index of the training pattern.

0 1 0 1 0 0 1 1 1 0 0 0 1	0 0 0 1 1 0 1 1 0 0 1 0 1	0 1 0 0 1 0 0 0 0 0 0 0 1
0.9231	0.8462	0.9231
15	10	4

Similarly, starting from a random initial hidden state, 10 steps of brief Gibbs sampling were applied and the results are shown in the table below.

0 0 0 1 0 0 1 1 1 0 0 0 1	0 1 0 0 0 0 0 0 0 1 0 0 1	0 0 0 1 1 1 0 0 0 1 0 0 1
1	0.8462	0.9231
15	1	13

The most significant results that were found was that although the initial input and hidden states were completely random, the output activation patterns all closely resembled the input training patterns. Most of the outputs had between 10 to 12 units that were the same as one of the input training patterns. The patterns usually corresponded closely to a specific training pattern rather than a blend of training patterns. Some of the outputs were also similar to those of the idealized test input patterns used in Part 2, as seen by some of the examples shown above. It seems that all of the categories were generated equally often. Which category is generated will roughly depend on how close a particular initial state was to a training pattern.

Discussion

Part 1 – Training the RBM

After the RBM models were trained with the different hidden layer sizes in Part 1, we found that the models were able to converge after approximately 300 runs of the train function. The difference in energy plots were expected to decrease as the model converges, however, the plots above did not show a clear indication of convergence. Thus, examining the values of the weights after each run provided a clearer indication that the model has converged.

Examining the weights learned by the models after convergence provides limited insight about what the RBM has learnt. While in some models, like the 2 hidden unit model, the values of the weights were usually either strongly correlated or uncorrelated with the hidden units, as the number of hidden units increased this relationship becomes weaker. The weights provide some indication about how each of the hidden units will respond to the input units, but it was not necessarily clear what the units learned in the 6 and 8 hidden unit models. In the 4 hidden unit model, some of the weights seemed to correspond to the idealized test input patterns but some were a blend and it was unclear what features were learned as well.

Part 2 – Clamping the idealized users' ratings

Testing the hidden unit activation by clamping the idealized test input patterns may be a better way of determining what the model has learned rather than examining the weights directly. The test input patterns activated the same hidden patterns with some amount of variation. The models with fewer hidden units (i.e. 2 and 4 hidden units) more consistently produced the same hidden patterns than those with more hidden units (i.e. 6 and 8 hidden units).

This suggests that the larger number of hidden units allows for the model to learn more features within the data. While it may be able to capture the 4 different categories of movies fairly well, it will also pick up on the more subtle differences within each group. The models with larger numbers of hidden units may have learnt a separate representation for each of the training patterns rather than learning the overall trend. This tells you how you should decide the optimal hidden layer size for a particular problem. If you would like it to be able to generalize well from the training data you would choose a smaller number of hidden units to capture a smaller number of features. If you would like to be able to "memorize" the training patterns rather than generalize, you would use a larger number of features so that it has the capacity to capture all of the differences in the training patterns.

Part 3 – Clamping the idealized hidden activations

After clamping the idealized hidden activation test patterns to the hidden layer, the output activations did not necessarily correspond to the idealized input test patterns corresponding to the four categories. This may be because the representations learnt for the four categories are more like distributed patterns rather than the sparse patterns that the idealized hidden activations suggest.

Perhaps using hidden patterns similar to those found after activation by the idealized test input patterns (from the Part 2) would produce patterns that more closely resemble the four movie categories.

Part 4 – Applying brief Gibbs sampling using idealized hidden activations

Applying 5 steps of brief Gibbs sampling using the same hidden activation patterns as Part 3 produced results that closely resembled the training patterns. The output activations seemed to settle to a training pattern rather than providing blends of input patterns.

This aligns with what would be expected from the RBM model. After several iterations of brief Gibbs cycles, the model settles to a pattern that it was trained on in the positive phase. In comparison to Part 3, one Gibbs cycle may not be enough to generate patterns that closely resemble the training patterns but applying this several times makes the model converge to a pattern rather that producing a blend state.

Part 5 – Applying brief Gibbs sampling using random initial state

When starting with a completely random initial state, whether it is an initial input state or initial hidden state, the output activation seems to converge to the training patterns rather than produce blends of training patterns. While the output activations do not always exactly correspond to a specific training pattern, it produces very close results.

This provides a better illustration of the point made above. Even though the model was given a completely random initial state, it was able to settle to one of the training patterns that it has learnt in the positive phase of the training. This means that the model was able to learn features in the training patterns that would allow it to generate similar patterns. Some of the patterns produced also resemble the idealized test input patterns, which suggests that it was able to learn the feature of the four movie categories. Therefore, we can conclude that this generative model was indeed able to learn features in the training patterns and can generate new patterns that have similar properties to the users' rating data it was trained on.