Collaborative Recommendation Engine with a Restricted Boltzmann Machine

Module: Introduction to Machine Learning

Common Track

École Centrale de Casablanca

Academic Year 2024/2025

Group 8: Adam LOZI, Reda BENKIRANE, Mohamed EL IDRISSI, Hamza EL OTMANI, Aymane EL FAHSI,

November 12, 2024

Contents

1 Introduction

1.1 Context of the Project

In the field of recommendation systems, the primary goal is to assist users in discovering relevant items (such as movies, books, products, etc.) based on their past preferences, in other words, **Content based filtering**, or based on the preferences of similar users, a concept that is called **Collaborative Filtering**. For instance, a movie recommendation system can analyze the movies a user has already watched and rated in order to suggest new movies they are likely to enjoy.

Recommendation systems are widely used by companies such as Netflix, Amazon, and YouTube to improve user experience and increase satisfaction and several research articles that Netflix published on how to attract customer satisfaction in the long-term. However, a major challenge in these systems is predicting user preferences in the absence of explicit data, i.e., predicting items that the user has not yet rated or viewed. Hence, the aim of our project.

1.2 Objectives of the Project

The objective of this project is to build a movie recommendation system using a **Restricted Boltzmann Machine (RBM)**, an unsupervised learning algorithm that can capture complex relationships between users and mobvies. More specifically, we aim to fill in missing ratings in a ratings matrix (where missing values are represented by zeros) by predicting the ratings that are most likely to be appreciated by the users.

This model will recommend the top 10 movies to a given user, considering only the movies they have not rated yet.

1.3 Justification of the Chosen Algorithm

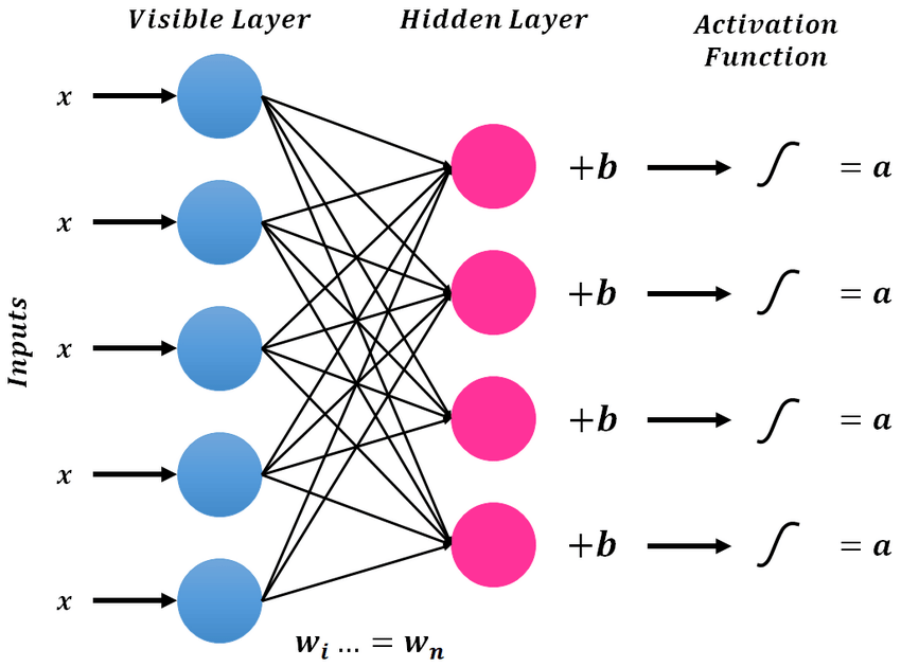

Figure 1: Boltzmann Machine

Restricted Boltzmann Machines (RBM) are probabilistic neural networks, particularly well-suited for modeling complex systems with latent variables a. By using an RBM, we can effectively model user preferences and relationships between movies, even when data is incomplete, in our case, there are missing ratings. This makes RBMs a powerful approach for recommendation systems, where the goal is to predict ratings for previously unseen movies.

In this project, the RBM will learn a pattern of both user and movie features, and it will be used to **reconstruct** the missing values in the ratings matrix. These reconstructed values will then be used to generate personalized recommendations.

2 Theoretical Foundation of Restricted Boltzmann Machines

2.1 Introduction to Restricted Boltzmann Machines

A **Restricted Boltzmann Machine** (RBM) is a generative stochastic neural network that can learn a probability distribution over its set of inputs. It consists of two layers:

- **Visible layer** $\mathbf{v} \in \mathbb{R}^D$: Represents the observable input data (continuous values), where D is the number of visible units.
- **Hidden layer** $\mathbf{h} \in \{0, 1\}^F$: Captures the dependencies and patterns in the data, where F is the number of hidden units.

The term "restricted" refers to the network architecture where no intralayer connections exist within the visible or hidden units, only interlayer connections between visible and hidden units.

2.2 Energy Function and Probability Distribution

An RBM assigns an **energy** to every configuration of visible and hidden units. For a Gaussian-Bernoulli RBM, where the visible units are continuous and the hidden units are binary, the energy function is defined as:

$$E(\mathbf{v}, \mathbf{h}) = \sum_{i=1}^{D} \frac{(v_i - b_i - \sum_{j=1}^{F} W_{ij} h_j)^2}{2\sigma^2} - \sum_{j=1}^{F} c_j h_j \tag{1}$$

where:

- $\mathbf{W} \in \mathbb{R}^{D \times F}$: Weight matrix between visible and hidden units.
- $\mathbf{b} \in \mathbb{R}^D$: Bias vector for visible units.
- $\mathbf{c} \in \mathbb{R}^F$: Bias vector for hidden units.
- σ^2: Variance of the Gaussian distribution for visible units (often set to 1 for simplicity).

The joint probability distribution over visible and hidden units is defined using the energy function:

$$p(\mathbf{v}, \mathbf{h}) = \frac{1}{Z} \exp\left(-E(\mathbf{v}, \mathbf{h})\right) \tag{2}$$

where Z is the partition function:

$$Z = \int_{\mathbf{v}} \sum_{\mathbf{h}} \exp\left(-E(\mathbf{v}, \mathbf{h})\right) d\mathbf{v} \tag{3}$$

2.3 Marginal and Conditional Probabilities

The marginal probability of a visible vector $\mathbf{v}$ is:

$$p(\mathbf{v}) = \sum_{\mathbf{h}} p(\mathbf{v}, \mathbf{h}) \tag{4}$$

Due to the bipartite structure, the conditional probabilities factorize nicely:

- **Conditional probability of hidden units given visible units**:

$$p(h_j = 1|\mathbf{v}) = \sigma\left(c_j + \sum_{i=1}^{D} \frac{v_i W_{ij}}{\sigma^2}\right) \tag{5}$$

where $\sigma(x)$ is the sigmoid activation function:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{6}$$

- **Conditional probability of visible units given hidden units**:

$$p(v_i|\mathbf{h}) = \mathcal{N}\left(v_i \mid \mu_i, \sigma^2\right) \tag{7}$$

where:

$$\mu_i = b_i + \sum_{j=1}^{F} W_{ij} h_j \tag{8}$$

and $\mathcal{N}(v_i \mid \mu_i, \sigma^2)$ denotes a Gaussian distribution with mean μ_i and variance σ^2.

2.4 Training the RBM

The goal is to learn the parameters $\theta = \{\mathbf{W}, \mathbf{b}, \mathbf{c}\}$ that maximize the likelihood of the observed data $\{\mathbf{v}^{(n)}\}_{n=1}^{N}$. The gradient of the log-likelihood with respect to the weights is:

$$\frac{\partial \log p(\mathbf{v})}{\partial W_{ij}} = \left\langle \frac{(v_i - \mu_i)}{\sigma^2} h_j \right\rangle_{\text{data}} - \left\langle \frac{(v_i - \mu_i)}{\sigma^2} h_j \right\rangle_{\text{model}} \tag{9}$$

Similarly for biases:

$$\frac{\partial \log p(\mathbf{v})}{\partial b_i} = \left\langle \frac{(v_i - \mu_i)}{\sigma^2} \right\rangle_{\text{data}} - \left\langle \frac{(v_i - \mu_i)}{\sigma^2} \right\rangle_{\text{model}} \tag{10}$$

$$\frac{\partial \log p(\mathbf{v})}{\partial c_j} = \langle h_j \rangle_{\text{data}} - \langle h_j \rangle_{\text{model}} \tag{11}$$

where:

- $\langle \cdot \rangle_{\text{data}}$: Expectation under the data distribution.
- $\langle \cdot \rangle_{\text{model}}$: Expectation under the model distribution.

2.5 Contrastive Divergence Approximation

Computing $\langle \cdot \rangle_{\text{model}}$ is intractable due to the partition function Z. To approximate this, we use the **Contrastive Divergence** (CD-k) algorithm:

1. **Positive Phase**:
   - Compute $p(h_j = 1|\mathbf{v}^{(n)})$ using Equation 5.
   - Sample $h_j^{(0)} \sim p(h_j|\mathbf{v}^{(n)})$.
2. **Negative Phase**:
   - For k steps:

$$\begin{aligned} \mu_i^{(k-1)} &= b_i + \sum_{j=1}^{F} W_{ij} h_j^{(k-1)} \\ v_i^{(k)} &\sim \mathcal{N}\left(\mu_i^{(k-1)}, \sigma^2\right) \\ p(h_j^{(k)} = 1|\mathbf{v}^{(k)}) &= \sigma\left(c_j + \sum_{i=1}^{D} \frac{v_i^{(k)} W_{ij}}{\sigma^2}\right) \\ h_j^{(k)} &\sim \text{Bernoulli}\left(p(h_j^{(k)} = 1|\mathbf{v}^{(k)})\right) \end{aligned}$$

3. **Parameter Update**:

$$\Delta W_{ij} = \eta \left(\left\langle \frac{(v_i^{(0)} - \mu_i^{(0)})}{\sigma^2} h_j^{(0)} \right\rangle - \left\langle \frac{(v_i^{(k)} - \mu_i^{(k)})}{\sigma^2} h_j^{(k)} \right\rangle \right) \tag{12}$$

$$\Delta b_i = \eta \left(\left\langle \frac{(v_i^{(0)} - \mu_i^{(0)})}{\sigma^2} \right\rangle - \left\langle \frac{(v_i^{(k)} - \mu_i^{(k)})}{\sigma^2} \right\rangle \right) \tag{13}$$

$$\Delta c_j = \eta \left(h_j^{(0)} - h_j^{(k)} \right) \tag{14}$$

where η is the learning rate.

2.6 Sampling Procedure

In practice, for continuous visible units and binary hidden units, the sampling involves:

1. **Sampling hidden units given visible units**:
   - Compute the probabilities using Equation 5.
   - Sample from Bernoulli distributions with these probabilities.
2. **Sampling visible units given hidden units**:
   - Compute the means $\mu_i = b_i + \sum_{j=1}^{F} W_{ij} h_j$.
   - Sample from Gaussian distributions: $v_i \sim \mathcal{N}(\mu_i, \sigma^2)$.

2.7 Regularization and Weight Decay

To prevent overfitting and improve generalization, a regularization term is added:

$$\text{Cost} = -\log p(\mathbf{v}) + \lambda \|\mathbf{W}\|_F^2 \tag{15}$$

where λ is the regularization coefficient and $\| \cdot \|_F$ denotes the Frobenius norm.

3 Application to Collaborative Filtering

In this project, we employ a Gaussian-Bernoulli Restricted Boltzmann Machine (RBM) to perform collaborative filtering on the MovieLens dataset. The objective is to build a recommendation system that predicts user preferences for movies based on their historical ratings and the ratings of other users. Collaborative filtering leverages patterns in user-item interactions to recommend items without requiring explicit feature engineering.

3.1 Data Representation

The MovieLens dataset provides user ratings for movies. We represent this data as a user-movie ratings matrix $\mathbf{R} \in \mathbb{R}^{U \times M}$, where:

- U is the number of users (610 in our dataset).
- M is the number of movies (9,742 in our dataset).
- Each entry $R_{u,m}$ corresponds to the rating given by user u to movie m.

To prepare the data for the RBM:

- **Normalization**: We normalize the ratings to the $[0, 1]$ range using Min-Max scaling:

$$R_{u,m} = \frac{R_{u,m} - R_{\min}}{R_{\max} - R_{\min}} \tag{16}$$

  where $R_{\min} = 0.5$ and $R_{\max} = 5$.

- **Handling Missing Data**: Unrated movies by a user are filled with zeros, indicating the absence of a rating.

3.2 RBM Architecture in Collaborative Filtering

In the context of collaborative filtering:

- **Visible Units** ($\mathbf{v} \in [0, 1]^M$): Each visible unit corresponds to a movie, and the input is the normalized ratings vector of a user (continuous values).
- **Hidden Units** ($\mathbf{h} \in \{0, 1\}^F$): Hidden units capture latent features that represent underlying factors influencing user preferences, where F is the number of hidden units (e.g., 200).

3.3 Model Equations

Energy Function The energy of the system is defined as:

$$E(\mathbf{v}, \mathbf{h}) = \sum_{i=1}^{M} \frac{(v_i - b_i - \sum_{j=1}^{F} W_{ij} h_j)^2}{2\sigma^2} - \sum_{j=1}^{F} c_j h_j \tag{17}$$

Conditional Probabilities The conditional probability distributions are:

$$p(h_j = 1|\mathbf{v}) = \sigma\left(c_j + \sum_{i=1}^{M} \frac{v_i W_{ij}}{\sigma^2}\right) \tag{18}$$

$$p(v_i|\mathbf{h}) = \mathcal{N}\left(v_i \mid b_i + \sum_{j=1}^{F} W_{ij} h_j, \sigma^2\right) \tag{19}$$

where $\sigma(x)$ is the sigmoid function.

3.4 Training Procedure

The RBM is trained using the Contrastive Divergence (CD-k) algorithm (with $k = 1$):

1. **Positive Phase**:
   - Compute the probability of hidden units given the visible units: $p(h_j = 1|\mathbf{v})$.
   - Sample $\mathbf{h}^{(0)}$ from this distribution.
2. **Negative Phase**:
   - Reconstruct the visible units: compute $\mu_i = b_i + \sum_{j=1}^{F} W_{ij} h_j^{(0)}$ and sample $\mathbf{v}^{(1)}$ from $\mathcal{N}(\mu_i, \sigma^2)$.
   - Recompute the hidden activations: compute $p(h_j = 1|\mathbf{v}^{(1)})$ and sample $\mathbf{h}^{(1)}$.
3. **Parameter Updates**:

$$\Delta W_{ij} = \eta \left(\left\langle \frac{(v_i^{(0)} - \mu_i^{(0)})}{\sigma^2} h_j^{(0)} \right\rangle - \left\langle \frac{(v_i^{(1)} - \mu_i^{(1)})}{\sigma^2} h_j^{(1)} \right\rangle \right) \tag{20}$$

$$\Delta b_i = \eta \left(\left\langle \frac{(v_i^{(0)} - \mu_i^{(0)})}{\sigma^2} \right\rangle - \left\langle \frac{(v_i^{(1)} - \mu_i^{(1)})}{\sigma^2} \right\rangle \right) \tag{21}$$

$$\Delta c_j = \eta \left(h_j^{(0)} - h_j^{(1)} \right) \tag{22}$$

   where η is the learning rate.

3.5 Implementation Details

Model Definition In our implementation, the RBM is defined as:

- **Weight Matrix** $\mathbf{W} \in \mathbb{R}^{F \times M}$ initialized with small random values from a normal distribution.
- **Bias Vectors** $\mathbf{b} \in \mathbb{R}^{M}$ and $\mathbf{c} \in \mathbb{R}^{F}$ initialized to zero.
- **Variance** σ^2 set to 1 for simplicity.

Sampling Functions We define functions to sample the hidden and visible units:

- `sample_h(v)`: Computes $p(\mathbf{h}|\mathbf{v})$ and samples $\mathbf{h}$.
- `sample`$\check{}$`(h)`: Computes $\mu_i = b_i + \sum jW_{ij}h_j$ and samples $\mathbf{v}$ from $\mathcal{N}(\mu_i, \sigma^2)$.

Forward Pass The forward pass involves:

1. Computing the hidden probabilities and sampling the hidden units: $(p_h, \mathbf{h}) =$ `sample_h`$(\mathbf{v})$.
2. Reconstructing the visible units: $\mathbf{v}' =$ `sample_v`$(\mathbf{h})$.

Training Loop The training function `train_rbm` iterates over the data in mini-batches:

1. For each batch:
   - Perform the positive phase to compute $p_h^{(0)}$ and sample $\mathbf{h}^{(0)}$.
   - Perform the negative phase to compute $\mathbf{v}^{(1)}$ and sample $\mathbf{h}^{(1)}$.
   - Calculate the gradients and update the weights and biases using the parameter updates.
2. Compute the reconstruction loss to monitor training progress.

3.6 Generating Recommendations

To generate recommendations for a user:

1. **Input User Ratings**: Obtain the user's normalized ratings vector $\mathbf{v}$.
2. **Compute Hidden Activations**: Compute $p(\mathbf{h}|\mathbf{v})$ and sample $\mathbf{h}$.
3. **Reconstruct Ratings**: Compute $\hat{\mathbf{v}} = \mathbf{b} + \mathbf{h}\mathbf{W}^\top$.
4. **Select Unrated Movies**: Identify movies the user hasn't rated (where $v_i = 0$).
5. **Recommend Top N Movies**: Recommend movies with the highest predicted ratings among the unrated movies.

3.7 Evaluation Metrics

We evaluate the recommendation system using:

- **Hit Rate@K**:

$$\text{Hit Rate@K} = \frac{\text{Number of Users with Hits}}{\text{Total Number of Users}} \tag{23}$$

  where a "hit" means the user's actual rated movie is among the top K recommendations.

- **Precision@K**:

$$\text{Precision@K} = \frac{\text{Relevant Items Recommended}}{K} \tag{24}$$

- **Recall@K**:

$$\text{Recall@K} = \frac{\text{Relevant Items Recommended}}{\text{Total Relevant Items}} \tag{25}$$

3.8 Experimental Results

Training We trained the RBM with the following parameters:

- **Hidden Units**: 200
- **Learning Rate**: 0.01
- **Epochs**: 100
- **Batch Size**: 16

The training process showed a decreasing trend in reconstruction loss, indicating that the model was learning to represent user preferences effectively.

3.9 Hyperparameter Tuning

We performed grid search over:

- **Hidden Units**: [100, 200, 300]
- **Learning Rates**: [0.01, 0.005, 0.001]

The best performance was achieved with 200 hidden units and a learning rate of 0.005. The RBM achieved:

- **Hit Rate@10**: 0.9918
- **Average Precision@10**: 0.6016
- **Average Recall@10**: 0.0868

4 Challenges Encountered During the Project

Throughout the development of this recommendation system using a Restricted Boltzmann Machine (RBM), several challenges arose that required careful analysis and experimentation to address effectively. Some of the primary challenges we encountered include:

4.1 Choosing the Appropriate RBM Variant (Bernoulli-Bernoulli vs. Gaussian-Bernoulli):

One of the initial challenges was selecting the optimal RBM variant for our data. Since our user-item rating matrix contains continuous ratings (e.g., values from 0.5 to 5 in increments of 0.5), a Bernoulli-Bernoulli RBM—typically used for binary input data—was not ideal. Instead, we needed an RBM capable of handling continuous values for visible units, making Gaussian-Bernoulli RBM a more suitable choice. However, Gaussian-Bernoulli RBMs come with their own complexities, particularly in tuning parameters like the variance of the Gaussian distribution for each visible unit. This decision ultimately impacted both the architecture and the training complexity of our model.

4.2 Hyperparameter Tuning for Optimal Performance:

Like many machine learning models, RBMs are sensitive to hyperparameters, including learning rate, number of hidden units, and batch size. For Gaussian-Bernoulli RBMs, additional parameters such as the Gaussian variance need tuning as well. Finding the right combination of these hyperparameters required extensive experimentation and computational resources. Additionally, overfitting was a risk due to the model's flexibility, so techniques like regularization and early stopping were considered.

4.3 Balancing Model Complexity and Computational Efficiency:

Training an RBM with a high number of hidden units can improve the model's ability to capture intricate patterns in the data, but this comes at the cost of increased computation time and memory usage. The need to iterate between training steps and model evaluation with limited computational resources became a bottleneck, especially when handling a large user-item matrix. Striking the right balance between complexity and efficiency was crucial, and sometimes required simplifying the model to ensure feasible training times without significantly sacrificing recommendation quality.

4.4 Interpreting and Validating Model Outputs:

Since RBMs are generative models, interpreting their outputs can be more challenging than with traditional collaborative filtering methods. The predicted ratings generated for missing values do not always correspond directly to a simple user-item preference score, and additional validation metrics like RMSE and Hit Rate had to be used to evaluate how well the recommendations matched users' actual preferences. Moreover, analyzing the latent features learned by the RBM to ensure they aligned with real-world user behavior added a layer of complexity to the project.

5 Conclusion

In this project, we successfully developed a recommendation system using a Gaussian-Bernoulli Restricted Boltzmann Machine (RBM) to address the task of predicting user preferences and recommending new items based on implicit patterns in their past ratings. Through an extensive exploration of probabilistic modeling, we created a recommendation framework capable of handling the continuous nature of user ratings and learned effective user-item representations that significantly enhance recommendation quality.

One of the primary accomplishments of this project was navigating the challenges of model selection, hyperparameter tuning, and handling sparse data. By carefully selecting the Gaussian-Bernoulli RBM variant, we accommodated continuous input values in our dataset, ensuring our model's relevance for real-world rating data. Additionally, we overcame computational constraints and achieved a balance between model complexity and efficiency, allowing us to generate timely recommendations suitable for practical applications.

Our approach demonstrated that RBMs can be a powerful tool for recommendation systems, offering flexibility and effectiveness in learning complex relationships in high-dimensional data. However, we also recognize that RBMs can be sensitive to hyperparameter settings and computationally demanding for large-scale datasets. These aspects open up pathways for further research, such as exploring hybrid approaches that combine RBMs with other collaborative filtering techniques or neural network architectures to enhance scalability and accuracy.

In conclusion, this project has provided a deep understanding of probabilistic recommendation systems and laid the groundwork for future developments in user-centered AI applications. With continuous improvements in computational methods and model efficiency, probabilistic models like RBMs remain a promising avenue for recommendation engines, especially when integrated into modern machine learning pipelines.