Networked Life: Project

03/03/2021

Project Goal

- Introduce new algorithms and techniques to provide custom recommendations to users based on what other users liked
- Learn some basic skills and gain practice experience in machine learning, and enrich your resume

Main Tasks

Part 1 Linear regression on Netflix dataset

- Expand on the homework Q4 (i.e., linear regression) by reusing your written code previously, applying it to the Netflix training dataset

Part 2 Restricted Boltzmann Machines (RMB)

- Develop a classic neural network model to predict a user's rating: restricted Boltzmann machines (RBM)

Part 3 Extensions to the RBM model

- Build extensions to the RBM model using some hints we give you and any information you can find online

Part 1

Linear regression on Netflix dataset

- Expand on the homework Q4 (i.e., linear regression) by reusing your written code previously, applying it to the Netflix training dataset

Dataset

- Two datasets in the folder (*code*):
  - *training.csv*: imported to a matrix by *getTrainingData* function in *projectLib* file.
  - *validation.csv*: imported to a matrix by *getValidationData* function in *projectLib* file.

training set	validation set	test set
		hidden

e.g.

movie ID	user ID	rating ID
5	4	3
⋮	⋮	⋮

Question 1.1

- Finish the parameter estimation *param* function to compute the baseline estimator for the training set, without regularization
- Finish the *predict* function to compute the predicted rating of every (movie, user) pair of the training set
- Then you can compute the RMSE to compare the two

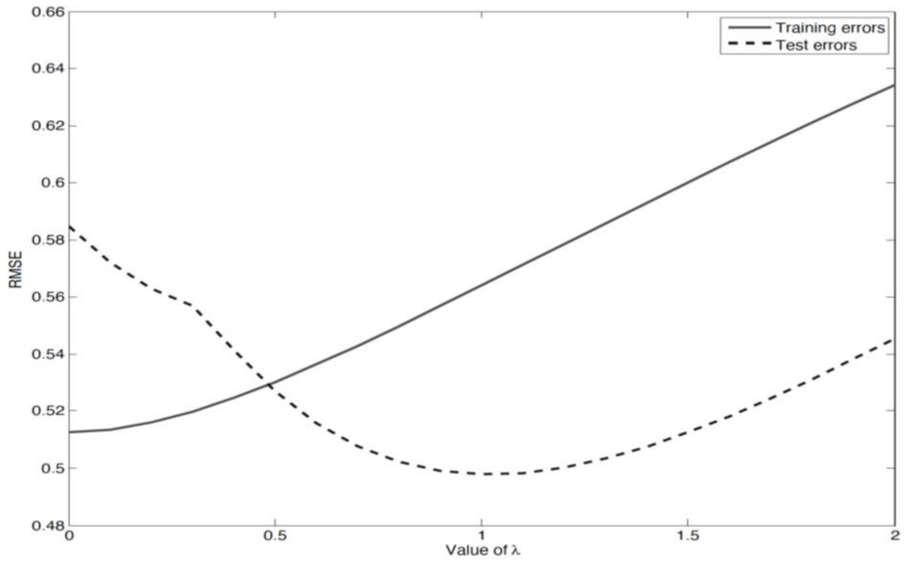
$$\{r_{ui}\} \longrightarrow \boxed{\text{Predictor } (b)} \longrightarrow \hat{r}_{uj} \qquad \forall u, j$$

Question 1.2

❖ Regularize this model so that it does not overfit, by adding a penalty to your biases. You should now complete the *param_reg* function.

$$\min_{\{b_u, b_i\}} ||Ab - c||_2^2 + \lambda||b||_2^2$$

Find the optimal λ



Review

Baseline Predictor

$$\hat{r}_{ui} = \bar{r} + b_u + b_i$$

$$b^* = \left(A^T A\right)^{-1} A^T c$$
$$b^* = (A^T A + \lambda I)^{-1} A^T c$$

Linear Regression

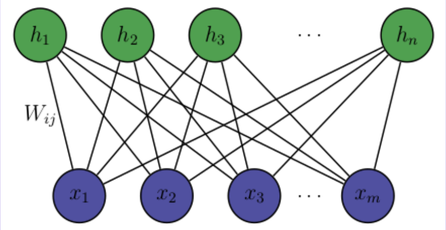
Neighborhood Predictor

$$d_{AB} = \frac{r_A^T r_B}{||r_A||_2 ||r_B||_2}$$

$$\hat{r}_{ui}^N = (\bar{r} + b_u + b_i) + \frac{\sum_{j \in \mathcal{L}_i} d_{ij} \tilde{r}_{uj}}{\sum_{j \in \mathcal{L}_i} |d_{ij}|}$$

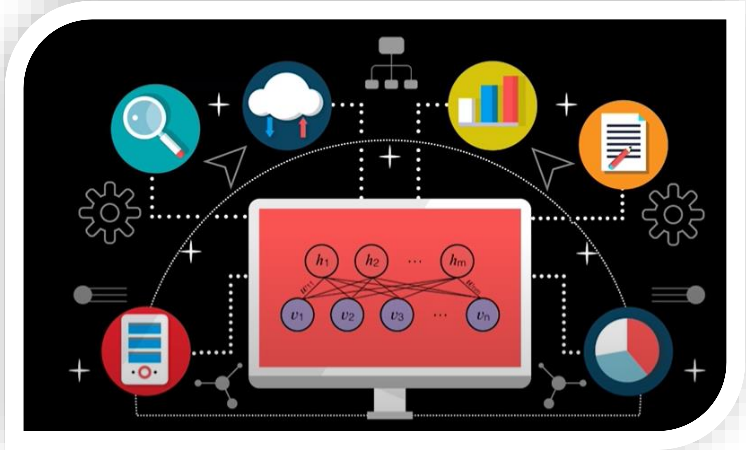
Interactions among movies and users

Restricted Boltzmann Machines (RBM)



undirected graphical model

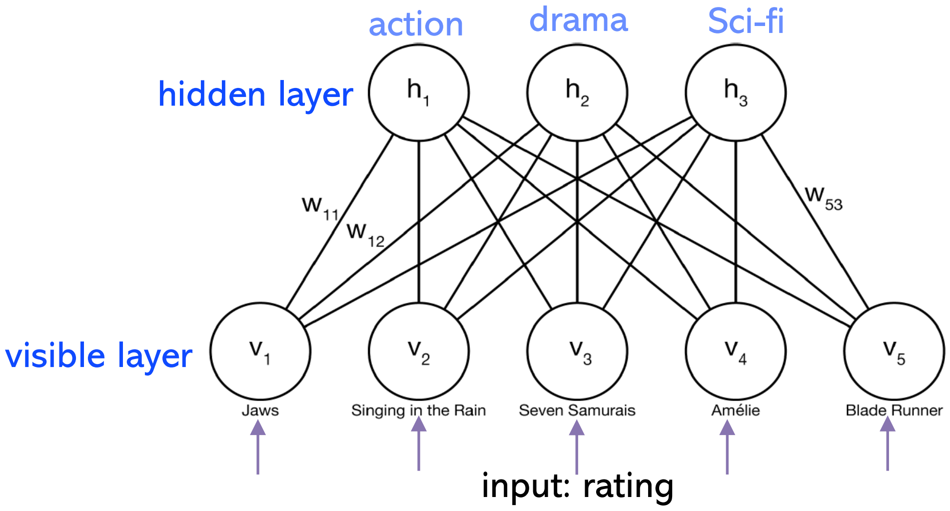
Part 2	Restricted Boltzmann Machines (RBM)



rbm.py, mainRBM.py

RMB model

- Visible layer V
  - Each node is corresponding to a movie
  - Receives input (e.g., user's ratings)
- Hidden layer H
  - Each node represents a feature (e.g., whether user likes action movie or not)



v_i,h_j are binary, i.e. {0,1}

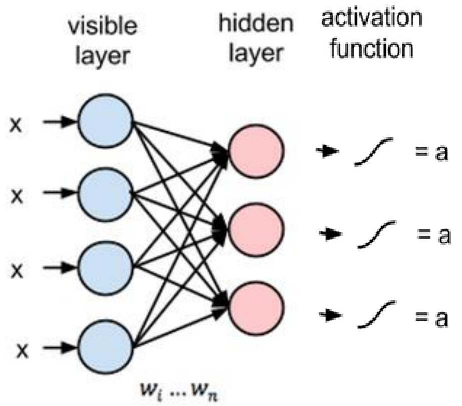
✓ Find the best weights W to represent our data

Question 2.1

- When visible layer receives input, the input is transformed to the hidden layer and passed through an activation function to produce the node's output:

$$\mathbb{P}(h_j = 1|\mathbf{v}) = \sigma\left(\sum_{i \in V} v_i W_{ij}\right), \text{ where } \sigma(x) = \frac{1}{1+e^{-x}} \, .$$

the probability of feature j to be active.

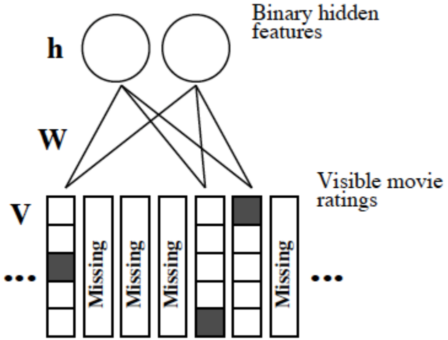


- In the *rbm.py* file, complete the function *sig*.

Question 2.2

❖ RBM requires the input data to be binary form. To adapt to the requirement, we encode each rating as a binary vector

- The rating from a user can be one of the five options, K = {1, 2, 3, 4, 5}
- If a user gives a rating 3, that piece of data will be encoded by the vector (0, 0, 1, 0, 0)



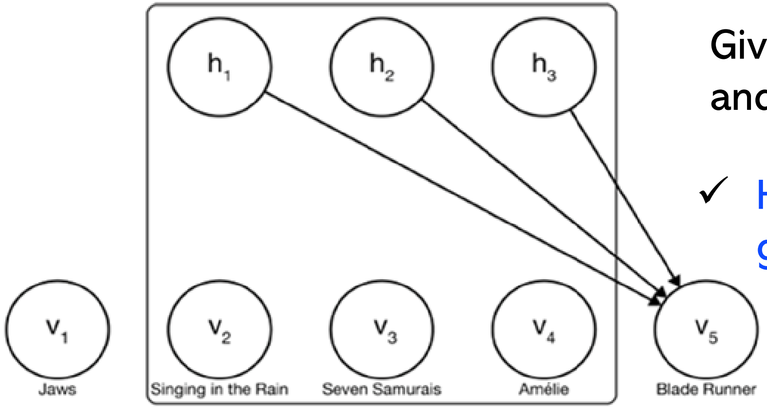
From visible states to the hidden ones:

$$\mathbb{P}(h_j = 1|\mathbf{v}) = \sigma\Big(\sum_{k\in K}\sum_{i\in V} v_i^k W_{ij}^k\Big)$$

❖ Write a function to propagate the visible input (binary vectors of ratings) to the hidden layer, in *visibleToHiddenVec*

Question 2.3

- After we compute the probability of features (hidden nodes), we then compute the predicted results of the user from hidden nodes to visible nodes



$\boldsymbol{v_i}$

Given binary vector of $\boldsymbol{h}$ and weight matrix $\boldsymbol{W}$

✓ Hidden activations flow down to give a distribution over the ratings

- Implement the *hiddenToVisible* function, taking as input a binary vector of hidden units and the edge weights.

Question 2.4

- Implement the function *getPredictedDistribution* to get the predicted distribution over the ratings for a movie

$$\mathbb{P}(v_i^k = 1|\mathbf{h}) = softmax\Big(\sum_{j\in J} h_j W_{ij}^k\Big), \text{ where } softmax(x_k) = \frac{e^{x_k}}{\sum_{l\in K} e^{x_l}}$$

e.g. P_i = (0.1, 0.2, 0.4, 0.3, 0), the probability that rating = 3 is 0.4

K = {1, 2, 3, 4, 5}

- ✓ Output is a distribution over all the possible K states, i.e., the probability that each of the K binary variables takes the value 1
- ✓ We need to transform the probability distribution of user's rating at different scores to a standard rating

Question 2.5

- Implement the two functions *predictRatingMax* and *predictRatingExp*, which implement the two options below

e.g., A user's probabilities for a movie i at 5 different score options:

P_i = (0.1, 0.2, 0.4, 0.3, 0), the probability that rating = 3 is 0.4

K = {1, 2, 3, 4, 5}

- ✓ Predict the rating with the highest probability. In the above example, the prediction =3.
- ✓ Predict the rating as weighted average of probabilities

$$prediction = \sum_{k \in K} p_i^k k \ .$$

In the above example:

$$prediction = 0.1 \times 1 + 0.2 \times 2 + 0.4 \times 3 + 0.3 \times 4 + 0 \times 5 = 2.9$$

Find weights

RBM $\{r_{ui}\} \longrightarrow$ **predictor** w $\longrightarrow \{\hat{r}_{ui}\}$

- Find the weights that maximize the probability that our model would generate the data it has been trained on, which is called log-likelihood $\max \log P(\mathbf{v})$

$$\frac{\partial \log p(\mathbf{v})}{\partial w_{ij}} = \langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model}$$

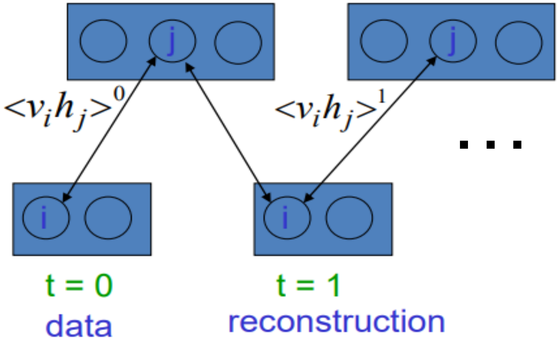
- Use a gradient ascent method.
  - Start from a random weight
  - Update weights by adding the gradient $\nabla W_{i,j}^{k}$

$$W \leftarrow W + \epsilon \nabla W_{i,j}^{k}$$

Gradient ascent method

- **Learning**

  - We start with the data $\mathbf{v}$, consisting by all the $K - elements$ vectors v_i, $i \in M_u$, of all the movies M_u that a particular user u has rated. This part is implemented by the function `getV` in the `rbm.py` file.
  - Compute $\mathbb{P}(h_j = 1|\mathbf{v})$ with `visibleToHiddenVec`.
  - Call positive gradient $(PG)_{i,j,k} = \mathbb{P}(h_j = 1|\mathbf{v}) \cdot v_i^k$ (computed by `probProduct`).



- **Unlearning:**

  - Sample the states of the hidden units according to $\mathbb{P}(h_j = 1|\mathbf{v})$.
  - Use `hiddenToVisible` to compute *"negative" data* $\overline{\mathbf{v}}$.
  - Essentially repeat the steps of the Learning part, this time with "negative" data, i.e compute $\mathbb{P}(h_j = 1|\overline{\mathbf{v}})$ and call "negative" gradient $(NG)_{i,j,k} = \mathbb{P}(h_j = 1|\overline{\mathbf{v}}) \cdot \overline{v}_i^k$.

Gradient ascent method

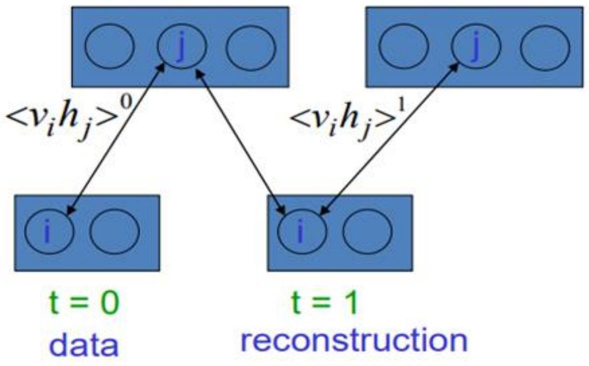
- **Synthesis:** Update only the weights for the movies that the user has rated.

$$W \leftarrow W + \epsilon \nabla W_{i,j}^{k}$$

Negative statistics

where $\nabla W_{i,j}^{k} = (PG)_{i,j,k} - (NG)_{i,j,k}$

Positive statistics for the connection between visible node i and hidden node j



RMB model

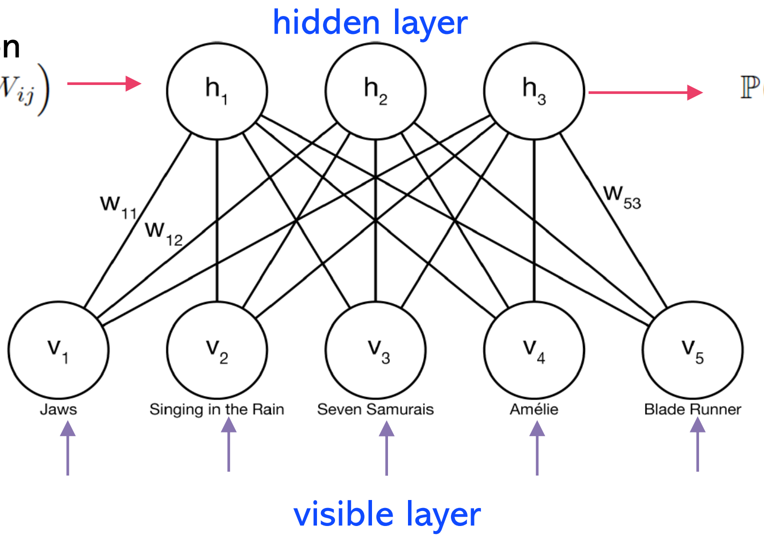
activation function

$$\mathbb{P}(h_j = 1|\mathbf{v}) = \sigma\Big(\sum_{i \in V} v_i W_{ij}\Big)$$

W

Binary vector $\boldsymbol{v}_i$

input: rating



$$\mathbb{P}(v_i^k = 1|\mathbf{h}) = softmax\Big(\sum_{j \in J} h_j W_{ij}^k\Big)$$

Distribution over rating

Max/Exp

rating

Gradient ascent method

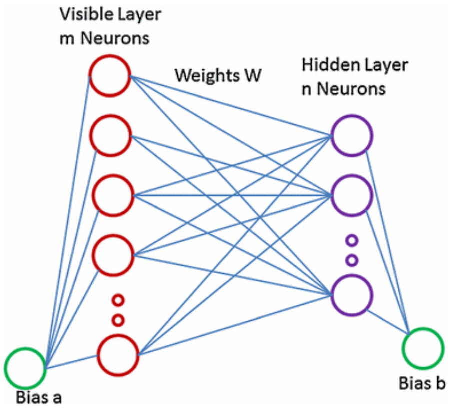
$\{v\}$ → **predictor** W → $\{\hat{v}\}$

max log$P(\mathbf{v})$

Part 3

Some extensions by online references

- Momentum
- Adaptive learning rates
- Early stopping
- Regularization
- Mini Batch
- Biases



Your algorithms

- Two algorithms
  - ✓ Linear regression (λ)
  - ✓ RBM (F, ϵ)

- It is important to train your model on the training set, and then see how well it performs on previously unseen data contained in the validation set

- You can change the hyperparameters to make the algorithms more performant

Test data

- Test your program on test data (*test.csv*)
- When you produce the rating file (for all movies and for all users), name it in the following format: <Group ID>_v<version number>.txt
- After receiving your file, we will compute the RMSE obtained on the test set and give you its value

Bonus

- Submit result once every week from week 9 to week 12 (Based on RBM, not Regression)
- At the end of each week, the current best RMSE (over all previous weeks) will be broadcasted to the class, for you to know which is the current goal to beat. Also, each week, the team with the best outstanding RMSE over the test set will get a bonus of 5 points
- At most 10-bonus score for each group

Week	Dates
9	Monday, March 22 - Sunday, March 28 (11:59 PM)
10	Monday, March 29 - Sunday, April 4 (11:59 PM)
11	Monday, April 5 - Sunday, April 11 (11:59 PM)
12	Monday, April 12 - Sunday, April 18 (11:59 PM)

Submission and Grading

- Project Group: Same with homework group.
- Submission: Source code + Report (Deadline: Week13)
- Grading:
  - (25%) Part 1: Linear regression (*linearRegression.py, projectLib.py*)
  - (25%) Part 2: Basic RBM (*rbm.py, mainRBM.py*)
  - (25%) Part 3: Extensions of the RBM
  - (25%) Part 4: Report
  - 5 bonus points for the group submitting the best results each week
  - 10 bonus points for the best results overall

Thanks!

Do you have any questions?