MSDS 630
HW 2
Gradient Descent for Matrix Factorization
Due: Feb 10, 2025 @ 11:59p

Getting Started

1 Matrix Factorization via Gradient Descent with Momentum (20 pts)

In this part of the assignment you will build a collaborative filtering model to predict ratings that MovieLens users give to movies. We will be using a dataset with 100836 ratings, 610 users, and 9724 movies.

Data can be downloaded using this command line:

```
wget http://files.grouplens.org/datasets/movielens/ml-latest-small.zip
```

1.1 Task:

Build the general architecture of a collaborative filtering learning algorithm, including:

1. Encoding rating data
2. Initializing parameters
3. Calculating the cost function
4. Calculating the gradient
5. Using gradient descent with momentum to fit the Matrix Factorization model
6. Predicting on new data
7. Putting it all together

1.2 Instructions:

- Starting code can be found in `hw2_start.py`. Some functions are pre-written according to their descriptions, others need to be written by you. These functions are indicated with sections like:

```
### BEGIN SOLUTION
### END SOLUTION
```

Note:

The specific directions for each function are found in the function docstring.

- Do not use loops (for/while) in your code, unless the instructions in the starting code explicitly ask you to do so. Extra loop usage will result in a loss of 5 points.

- Do not submit data to Github. Doing so will result in a loss of 5 points.

1.3 Deliverables:

Submit your completed functions as a Python script entitled "mf.py".

1.4 Evaluation:

You can guide your testing with the notebook: `ex-test-hw2.ipynb`. Once you've finished, make sure you can run:

```
pytest test_tiny.py
```

There are 4 tests to pass: one for the encoding, one for calculating the cost, one for calculating the gradient, and one for carrying out gradient descent with momentum. Each pass is worth 5 points. If you fail a test, but your code is on the right track, half credit will be awarded for that test.

2 Matrix Factorization with Bias (5 pts)

We want to extend the Matrix Factorization model discussed in class to add a "bias" or "offset" parameter for each user and another bias/offset parameter for each movie. This parameter is analogous to the intercept coefficient in linear regression. For the problem in class we had the parameters matrices U and V . We are adding the bias term u^* which is a vector of dimension n_u and v^* which is a vector of dimension n_m. Here is the equation for this new model:

$$\hat{y}_{ij} = u_i^* + v_j^* + u_i \cdot v_j$$

Note: u_i^* and v_j^* are scalars, and $u_i \cdot v_j = (UV^T)_{ij}$.

Write the gradient descent updating equations for u_i^*, v_j^*, and the elements of U and V.

2.1 Evaluation:

The following will be evaluated:

- correct loss function (1pt)
- correct updating equations for elements of U and V (2pts)
- correct updating equations for bias parameters (2pts)

3 Logistic Matrix Factorization (5 pts)

Consider a website in which users give ratings of "thumbs up" or "thumbs down" to items. We can encode "thumbs up" by user i on item j as $y_{ij} = 1$ and a "thumbs down" by user i on item j as yij = 0. We can use a version the matrix factorization model called Logistic MF. In this version we use a sigmoid after the dot product. The model is $\hat{y}_{ij} = \sigma(u_i \cdot v_j)$ where $\sigma(x) = (1 + e^{-x})^{-1}$ is the sigmoid function and $y_{ij} = 1$ is either 0 or 1 (or empty). You have n_u users, n_m items and K is the embedding size. $\hat{y}_{ij}$ is interpreted as a probability and the loss function is the binary cross entropy loss:

$$-\frac{1}{N}\left[\sum_{(i,j):r_{ij}=1} y_{ij}\log(\hat{y}_{ij}) + (1-y_{ij})\log(1-\hat{y}_{ij})\right],$$

where N is the number of ratings with y_{ij} either 0 or 1. Here $r_{ij} = 1$ if there is a rating (0 or 1) of item j by user i and 0 otherwise.

1. How many parameters do you have in this model?
2. Given the following test set with predictions, compute the test loss:

user id	item id	rating	$\hat{y}$
0	0	0	0.6
0	3	1	0.4
1	2	0	0.4
1	0	1	0.9
2	1	1	0.8
2	9	0	0.05

3. Suppose you run your optimization and get the following U and V matrices.

$$U = \begin{bmatrix} 0.2 & 2.4 & -0.1 \\ 1.6 & 1.0 & 2.0 \\ 2.6 & -0.6 & 1.5 \\ 0.9 & 0.7 & 0.2 \\ 2.0 & 0.4 & 0.3 \\ -1.9 & 0.5 & 2.3 \\ 0.8 & -0.9 & 0.9 \end{bmatrix}, \quad V = \begin{bmatrix} 0.0 & 3.5 & 1.0 \\ 3.6 & 0.0 & 0.1 \\ 0.0 & 4.0 & 0.9 \\ 2.5 & 0.1 & 0.2 \end{bmatrix}$$

   Write a numeric expression for the value of $\hat{y}_{13}$ in terms of the sigmoid function. Assume matrix indices start at 1.

4. What is the value of K?

5. Think about how the gradient calculations for this model differ from the previous model. Go ahead and write them if you want more practice but no need to submit this part.

3.1 Evaluation:

The following will be evaluated:

- correct number of parameters in model (1pt)
- correct test loss (1pts)
- correct predicted value of y_{13} (2pts)
- correct embedding size (1pt)