Movie Embeddings

Problem :. This homework is about "movie embeddings,"which are like word embeddings but can be used to recommend movies to people who provide one movie, or a number of movies that they like. Movie vectors will be created using a dataset of movie likes of a large number of users. To define movie vectors we use an analog of the *Distributional hypothesis for word meaning*: you can "understand"a movie by looking at people who liked this particular movie, and asking what other movies they like.

We use a very simple embedding. Let $X_{i,j}$ be the number of users that liked both movies i and j. Then we train the vectors $v_1, v_2, \ldots,$ for all the movies using the following objective function, where v_i is the vector for movie i.

$$Cost = \sum_{i=1}^{n} \sum_{j=1}^{n} \gamma_{ij} (v_i \cdot v_j - X_{ij})^2$$

Where $\gamma_{ij} = 1$ if i and j are different, and $= 0$ if $i = j$. Also, $v_i \cdot v_j$ is vector inner product.

The input to this optimization are the X_{ij} counts and it has to find the movie vector v_i for each movie i.

To optimize, we use gradient decent: for each variable y_k in the optimization the update at step t is

$$y_k \to y_k - \eta \cdot \nabla_{t,k}$$

Here $\nabla_{t,k}$ denotes the partial derivative of the objective with respect to variable y_k at iteration t. The variables are coordinates of the movie vectors.

Part 1: implementation

The X_{ij} counts will come from the MovieLens dataset in which 943 users rated 1682 movies. We have preprocessed the dataset to simplify the homework; download it from:

movieratings.csv:
http://www.cs.princeton.edu/courses/archive/fall16/cos402/ex/movieratings.csv.
movies.csv:
http://www.cs.princeton.edu/courses/archive/fall16/cos402/ex/movies.csv.

The file movieratings has 100,000 rows. The three entries in each row represent movie_id, user_id and movie rating, respectively. A rating of 1 indicates the user likes the movie, while a rating of 0 indicates the user does not like the movie. movies.csv has 1682 movies. It maps each movie_id to its title.

In this part, you should write a script named as MovieEmbeddings.py. It should have the following functions:

- `make_cooccurence` : This function creates the co-occurrence matrix. Each entry $X_{i,j}$ is the number of users who like both movie i and j. Because of the way the matrix is created, it should be symmetric.

- `train` : This function trains the movie vectors on the MovieLens dataset using gradient decent. You should first initialize the movie vectors using standard normal distribution. In your code, set learning rate η to 0.00001; the dimension of a movie vector k to 300, and run 200 iterations. K is the dimension of the movie vector, i.e. the matrix of the movie vectors is n by k. There are 1682 movie vectors and each movie vector has 300 elements. (You can try other settings but these worked in our trials.) For debugging purposes, `Train` should print the value of the cost function at the end of each iteration.
- `gradient` : This function does the main work for `train`. It is the inner loop of the optimization which uses gradient descent.
- `recommend1` : This function recommends top 20 movies to a user when given a movie that the user likes. Basically, you need to calculate the cosine similarity score between the given movie vector and all the other movie vectors, and then pick the top 20 movies that have the highest cosine similarity scores.
- `recommend2` : Similar to recommend1, this function gets a list of movies from the user, not just a single movie and return top 20 recommendations. It first computes the average of the vectors for these movies, say v, and recommend other movies based upon cosine similarity with v.

Part 2: Training and Recommendations

- Train the movie vectors with the dataset provided. Set the learing rate η at 0.00001 and the size of each movie vector k at 300. Output the value of the cost function after each iteration of `gradient` for the first 10 iterations.
- Output the top 20 recommended movies returned by `recommend1` given the movie LION KING.
- Output the top 20 recommended movies returned by `recommend2` given the three movies: SLEEPLESS IN SEATTLE, PHILADELPHIA STORY, SEX, LIES, AND VIDEOTAPE

What and how to turn in:

- 1. Turn in hard copies in class on the due date.

  A printout of all your python **scripts**, **movie recommendations** and **value of the objective function in each iteration of gradient for 10 iterations**.

- 2. Upload your code and outputs to CS dropbox by the due date.

  Using this DropBox link,

  http://dropbox.cs.princeton.edu/COS402_F2016/Programming_Assignment3,

  Upload all your python **scripts** and the cooccurrence matrix file: **cooccurrence.csv.gz**. You should only turn in uncompressed .py files. All code should be working and well documented. If appropriate, a readme.txt file explaining briefly how your code is organized, what data structures you are using, or anything else that will help the graders understand how your code works.