6240 Spring 2017 HW #2: Recommender Systems & Deep Learning
Out: 3/06 Due: 3/27

Problem 1 [40 points]

- Read papers [2,3]
- Download the movielens 100K ratings dataset from here
- The dataset consists of 943 users and 1682 movies in which each user has rated at least 20 movies. Choose ua.base/test and ub.base/test for result comparison between different training sets and testing sets; and ua.base/test for model comparison between different algorithms. The training and testing data are pre-processed as follows: we apply the database to build a U by I matrix A, where U is the number of users, and I is the number of rated movies. Each element A_{ui} denotes the rating scored by the u-th user for the i-th movie.
    a. Plot the histogram of the variable #movie_ratngs = number of ratings for a movie.
    b. Plot the histogram of the variable #user_ratings = number of ratings for a user
- Build a recommender system where the baseline predictor for a (u,i) pair is : $b_{ui}=\mu+b_u+b_i$ as described in [2] section 4.1.
    a. Estimate μ, b_u, b_i empirically and compute the means square error on the test set according to the following formula

$$\text{RMSE} = \sqrt{\frac{1}{|S_{\text{test}}|}\sum_{(u,i)\in S_{\text{test}}}(R_{ui}-T_{ui})^2}$$

    b. Plot the RMSE per user x movie rating. For example plot the RMSE for all the test user x movies that have test rating 1,2,3,4,5. You have to deliver 5 plots.
    c. Plot the RMSE per user. For example, bin your user in 5 different categories based on their average rating (you get that from your training data). For example, if the average ratings for user range from 3 to 5, then create 5 intervals between 3 and 5. For every bin plot the RMSE for all the users in that bin.
- Build a recommender system based on [2] section 4.2. For a (u,i) pair compute the rating by using the item similarity function:

$$\text{Similarity}(i_1, i_2) = \frac{\sum_{u \in U(i_1) \cap U(i_2)} (R_{u,i_1} - \bar{R}_u)(R_{u,i_2} - \bar{R}_u)}{\sqrt{\left(\sum_{u \in U(i_1) \cap U(i_2)} (R_{u,i_1} - \bar{R}_u)^2\right)\left(\sum_{u \in U(i_1) \cap U(i_2)} (R_{u,i_2} - \bar{R}_u)^2\right)}}$$

For a (u,i) pair find the k most similar items to i that the user u has rated and compute the average rating as indicated in [3] section 3.2.1

a. Compute the test RMSE for k=1,2,3,5,10 and find the optimal one.
b. Plot (for k=2 and k=5) the RMSE per user x movie rating. For example plot the RMSE for all the test user x movies that have training rating 1,2,3,4,5. You have to deliver 5 plots.
c. Plot the RMSE per user. For example, bin your user in 5 different categories based on their average rating (you get that from your training data). For example, if the average ratings for user range from 3 to 5, then create 5 intervals between 3 and 5. For every bin plot the RMSE for all the users in that bin.

Problem 2 [20 points]

- Download the MNIST dataset. It's available in many places, including Kaggle
- Create a modified training set
    a. For each image in the train set, generate two images where the probability of a pixel value being flipped is 0.03
    b. Your new training set size should be 120K
- Review the blog on keras and autoencoders, https://blog.keras.io/building-autoencoders-in-keras.html
- Implement a convolutional autoencoder (four times)
    a. Vary the CNN architecture in four ways
    b. Optimize as best as possible during training
- Plot the test set error (mean L2 error) as a function of the number of samples used to train each autoencoder
- Plot the best- and worst set of digits as measured by error (original image, reconstructed image)

Problem 3 [40 points]

- Download the imagenet collection from http://image-net.org/. The imagenet collection is very big for the scope of the class. We will pick few categories so that we can keep the computations tractable. As you can see

http://image-net.org/explore the categories are not balanced. For the scope of this class we will use the images from:

a. Plant, flora, plant life (1271)
b. Geological formation (1808)
c. Fungus (1207)
d. Sport (1888)
e. Person (1242)
f. Animal (1571)

- Convert the images to vectors:
  a. Autoencoder (unsupervised): You can pick your model details, but justify them (loosely)
  b. Singular Value Decomposition: Represent each image as a vector by unfolding it columnwise. If you have N images, and each image is 256 x 256 that will give you a matrix of N x 2^16. You can do SVD on that matrix and transform each image as a k dimensional vector. You may choose to resample images to be uniform size beforehand. You can pick the number of singular values to retain (k), but provide a justification.
  c. Histograms: For every image,
     i. Create RGB histograms and combine to create a single feature vector by concatenating RGB vectors.
     ii. Create HSV histograms and combine to create a single feature vector by concatenating HSV vectors.
- Compare classification results. Each of the four techniques in #2 above convert images into a vector. For each of the four techniques:
  a. Use Euclidean distance and Pearson correlation coefficient to find the 5 nearest neighbors of each image.
  b. Use majority vote of the 5 NNs to determine the category of each image.
  c. Create a 6x6 confusion matrix for each distance metric whose rows index the true categories and whose columns index the predicted category values. Compute counts for each cell, and the overall accuracy of the system (percentage of correctly classified images).

Deliverables

Present a detailed report on your experiment experiment that explains your design choices and results, shows examples of outputs and inputs, etc.. Are the results reasonable? How did you test to assure so? Reports should include code in python or ipython notebooks.