CS189: Introduction to Machine Learning

Homework 7

Due: 11:59 p.m.Sunday May 3rd, 2015

1   K-means Clustering

Run k-means on MNIST with $k = 5, 10, 20$ cluster centers and visualize the centers. You must implement the k-means algorithm yourself.

1. Use the image data at `mnist_data/images.mat`. There 60,000 unlabeled images. Each image contains 28×28 pixels.

2. Visualize the cluster centers, viewing each coordinate as a pixel.

3. Repeat the algorithm for multiple times with random initialization. Does the k-mean loss vary in different runs?

2   Joke Recommender System

You will build a personalized joke recommender system. There are $m = 100$ jokes and $n = 24,983$ users. As historical data, every user read a subset of jokes and rated them. The goal is to recommend more jokes, such that the recommended jokes match the individual user's sense of humour.

2.1   Data Format

The historical rating is represented by a matrix $R \in \mathbb{R}^{n \times m}$. The entry R_{ij} represents the user i's rating on joke j. The rating is a real number in $[-10, 10]$: a higher value represents that the user is more satisfied with the joke. If the joke is not rated, then the corresponding entry value is `NaN`.

The directory `joke_data/jokes/` contains the text of all 100 jokes. Read them before you start! In addition, you are provided with three files at `joke_data/`:

- `train.mat` is given as the training data. It contains the matrix R specified above.

- `validation.txt` contains user-joke pairs that doesn't appear in the training set. Each line takes the form "`i, j, s`", where `i` is the user index, `j` is the joke index, `s` indicates whether the user likes the joke. More specifically, `s = 1` if and only if the user gives a positive rating to the joke.

- `query.txt` contains user-joke pairs that are neither in the training set nor in the validation set. Each line takes the form "`id,i,j`". You are asked to predict if user `i` likes joke `j`. The integer `id` is a unique id for the user-joke pair. Use it to submit the prediction to Kaggle (see Section 2.4).

2.2 Warm-up

Let's start with the simplest recommender system: recommend the joke by its average rating in the training set. In particular, assert that any user will like the joke if its average rating is positive. Report your prediction accuracy on the validation set.

Next, we turn to a more advanced system where personal preferences are taken into account. Replace all missing values of R by zero, then measure the distance of any two users by the Euclidean distance between their rating vectors. For each user, find the k nearest neighbors of him/her, then make the prediction by averaging the ratings of these neighbors. Report your prediction accuracy on the validation set with $k = 10$, $k = 100$ and $k = 1,000$. How are the accuracies compare to the simple system?

2.3 Latent Factor Model

Latent factor model is the state-of-the-art method for personalized recommendation. It learns a vector representation $u_i \in \mathbb{R}^d$ for each user and a vector representation $v_j \in \mathbb{R}^d$ for each joke, such that the inner product $\langle u_i, v_j \rangle$ approximates the rating R_{ij}. You will build a simple latent factor model.

1. Replace all missing values by zero. Then use principle component analysis (PCA) to learn vector representation for users and jokes. Note that you are NOT allowed to use any inbuilt functions for PCA in MATLAB.

2. Evaluate the learnt vector representations by mean squared error:

$$\text{MSE} = \sum_{(i,j)\in S} (\langle u_i, v_j \rangle - R_{ij})^2 \quad \text{where } S := \{(i,j) : R_{ij} \neq \texttt{NaN}\}. \tag{1}$$

   Try $d = 2, 5, 10, 20$. How does the MSE vary as a function of d? Use the inner product $\langle u_i, v_j \rangle$ to predict if user i likes joke j. Report prediction accuracies on the validation set.

3. For sparse data, replacing all missing values by zero is not a completely satisfying solution. A missing value means that the user has not read the joke, but doesn't mean that the rating should be zero. A more reasonable choice is to minimize the MSE only on rated joke. Let's define a loss function:

$$L\Big(\{u_i\}, \{v_j\}\Big) := \sum_{(i,j)\in S} (\langle u_i, v_j \rangle - R_{ij})^2 + \lambda \sum_{i=1}^{n} \|u_i\|_2^2 + \lambda \sum_{j=1}^{m} \|v_j\|_2^2,$$

where set S has the same definition as in equation (1) and $\lambda > 0$ is the regularization coefficient. Implement an algorithm to learn vector representations by minimizing the loss function $L(\{u_i\}, \{v_j\})$.

Hint: you may want to employ an alternating minimization scheme. First, randomly initialize $\{u_i\}$ and $\{v_j\}$. Then minimize the loss function with respective to $\{u_i\}$ by treating $\{v_j\}$ as constant vectors, and minimize the loss function with respect to $\{v_j\}$ by treating $\{u_i\}$ as constant vectors. Iterate these two steps until both $\{u_i\}$ and $\{v_j\}$ converge. Note that when one of $\{u_i\}$ or $\{v_j\}$ is given, minimizing the loss function with respect to the other part has closed-form solutions.

4. Compare the resulting MSE and the prediction error with Step 2. Note that you need to tune the hyper-parameter λ to optimize the performance.

2.4 Recommending Jokes

1. Using the methods you have implemented to predict the users' preference on unread jokes. For each line "id, i, j" in the query file, output a line "id, s" to a file named kaggle_submission.txt. Here, s $= 1$ means that user i will give a positive rating to joke j, while s $= 0$ means that the user will give a non-positive rating. The first line of kaggle_submission.txt should be the field titles: "Id, Category". Submit kaggle_submission.txt to Kaggle.

2. (Optional) Build a joke recommender system for your friends: randomly display 5 jokes to receive their ratings, then recommend 5 best jokes and 5 worst jokes. Report your friends' ratings on the jokes that the system recommends.

3 (Optional) Eigenfaces

This exercise is optional. You will implement the Eigenfaces algorithm to find a common bases between celebrity faces and student faces. You are provided three files as part of the exercise:

- CelebrityDatabase.zip contains images of (aligned) faces of celebrities from which you will construct eigenfaces.

- StudentDatabase.zip contains images of aligned students' faces for which you will reconstruct from the eigenfaces.

- mask.mat contains a boolean mask that zeroes out the background in the face images.

To load an image file to MATLAB, use im $=$ imread(image_filename). In order to keep only the relevant pixels in a face image and convert it into a vector, execute:

unmasked_pixels = find(mask); im_vector = im(unmasked_pixels);

In order to obtain a masked image from im_vector, use:

```
full_im = zeros(size(binary_mask)); full_im(unmasked_pixels) = im_vector;
```

You need to do the following steps to find eigenfaces and the best matches.

1. Compute eigenfaces from the celebrity faces (after applying the binary mask). Visualize the top 10 eigenfaces. What kind of variations do the top eigenfaces seem to correspond to?

2. Reconstruct 5 celebrity faces from only the top 10 eigenfaces and visualize some of them with the original image and the reconstructed image side-by-side. Plot the average ℓ_2-error as a function of the number of eigenfaces used to reconstruct the original image.

3. Pick 5 student faces and reconstruct them with top 20 eigenfaces. Visualize these reconstructions as you did above in your write up. Plot the average ℓ_2-error as a function of the number of eigenfaces to reconstruct the original image.

Include all the figures, numbers and answers to the questions in your report.