1. Introduction

Your task is to build a mixture model for collaborative filtering. You are given a data matrix containing movie ratings made by users where the matrix is extracted from a much larger Netflix database. Any particular user has rated only a small fraction of the movies so the data matrix is only partially filled. The goal is to predict all the remaining entries of the matrix.

You will use mixtures of Gaussians to solve this problem. The model assumes that each user's rating profile is a sample from a mixture model. In other words, we have K possible types of users and, in the context of each user, we must sample a user type and then the rating profile from the Gaussian distribution associated with the type. We will use the Expectation Maximization (EM) algorithm to estimate such a mixture from a partially observed rating matrix. The EM algorithm proceeds by iteratively assigning (softly) users to types (E-step) and subsequently re-estimating the Gaussians associated with each type (M-step). Once we have the mixture, we can use it to predict values for all the missing entries in the data matrix.

Setup:

As with the last project, please use Python's **NumPy** numerical library for handling arrays and array operations; use **matplotlib** for producing figures and plots.

1. *Note on software:* For all the projects, we will use python 3.6 augmented with the **NumPy** numerical toolbox, the **matplotlib** plotting toolbox. In this project, we will also use the typing library, which is already included in the standard library (no need to install anything).

2. Download netflix.tar.gz and untar it in to a working directory. The archive contains the following python files:

- `kmeans` where we have implemented a baseline using the K-means algorithm
- `naive_em.py` where you will implement a first version of the EM algorithm (tabs 3-4)
- `em.py` where you will build a mixture model for collaborative filtering (tabs 7-8)
- `common.py` where you will implement the common functions for all models (tab 5)
- `main.py` where you will write code to answer the questions for this project
- `test.py` where you will write code to test your implementation of EM for a given test case

Additionnally, you are provided with the following data files:

- `toy_data.txt` a 2D dataset that you will work with in tabs 2-5
- `netflix_incomplete.txt` the netflix dataset with missing entries to be completed
- `netflix_complete.txt` the netflix dataset with missing entries completed
- `test_incomplete.txt` a test dataset to test for you to test your code against our implementation
- `test_complete.txt` a test dataset to test for you to test your code against our implementation
- `test_solutions.txt` a test dataset to test for you to test your code against our implementation **updated on August 8th**

Tip: Throughout the whole online grading system, you can assume the NumPy python library is already imported as `np`. In some problems you will also have access to other functions you've already implemented. Look out for the "Available Functions" Tip before the codebox, as you did in the previous projects.

This project will unfold both on MITx and on your local machine. However, we encourage you to first implement the functions locally and run them to validate basic functionality. Think of the online graders as a submission box to submit your code when it is ready. You should not have to use the online graders to debug your code.

Discussion

Show Discussion

Topic: Unit 4 Unsupervised Learning (2 weeks) :Project 4: Collaborative Filtering via Gaussian Mixtures / 1. Introduction