# Homework 3: (100 points)

## 1 Collaborative Filtering [50 points]

- Read the paper Empirical Analysis of Predictive Algorithms for Collaborative Filtering linked on the course web page. You need to read up to Section 2.1, and are encouraged to read further if you have time.
- The dataset we will be using is a subset of the movie ratings data from the Netflix Prize. You need to download it from the course web page. It contains a training set, a test set, a movies file, a dataset description file, and a README file. The training and test sets are both subsets of the Netflix training data. You will use the ratings provided in the training set to predict those in the test set. You will compare your predictions with the actual ratings provided in the test set. The evaluation metrics you need to use are the Mean Absolute Error and the Root Mean Squared Error. The dataset description file further describes the dataset, and will help you get started. The README file is from the original set of Netflix files, and has been included to comply with the terms of use for this data.
- Implement the collaborative filtering algorithm described in Section 2.1 of the paper (Equations 1 and 2; ignore Section 2.1.2) for making the predictions. You may program in C, C++, Java or Python.

## 2 Neural Networks, Gradient Boosting and SVMs [30 points]

- For this part, you will use scikit learn.
- Download the MNIST dataset available at `http://yann.lecun.com/exdb/mnist/`. The dataset has a training set of 60,000 examples, and a test set of 10,000 examples where the digits have been centered inside 28x28 pixel images. You can also use scikit-learn to download and rescale the dataset using the following code:

```
from sklearn.datasets import fetch_openml
# Load data from https://www.openml.org/d/554
X, y = fetch_openml('mnist_784', version=1, return_X_y=True)
X = X / 255.

# rescale the data, use the traditional train/test split
# (60K: Train) and (10K: Test)
X_train, X_test = X[:60000], X[60000:]
y_train, y_test = y[:60000], y[60000:]
```

- Use the SVM classifier in scikit learn and try different kernels and values of penalty parameter. **Important:** Depending on your computer hardware, you may have to carefully select the parameters (see the documentation on scikit learn for details) in order to speed up the computation. Report the error rate for at least 10 parameter settings that you tried (see how it is reported on `http://yann.lecun.com/exdb/mnist/`). Make sure to precisely describe the parameters used so that your results are reproducible.
- Use the MLPClassifier in scikit learn and try different architectures, gradient descent schemes, etc. Depending on your computer hardware, you may have to carefully select the parameters of MLPClassifier in order to speed up the computation. Report the error rate for at least 10 parameters that you tried. Make sure to precisely describe the parameters used so that your results are reproducible.
- Use the GradientBoostingClassifier in scikit learn and try different parameters (see the documentation for details). Again depending on your computer hardware, you may have to carefully select the parameters in order to speed up the computation. Report the error rate for at least 10 parameters that you tried. Make sure to precisely describe the parameters used so that your results are reproducible.
- What is the best error rate you were able to reach for each of the three classifiers?

## 3 K-means clustering on images [30 points]

In this problem, you will use K-means clustering for image compression. We have provided you with two images.

- Display the images after data compression using K-means clustering for different values of K (2, 5, 10, 15, 20).
- What are the compression ratios for different values of K? Note that you have to repeat the experiment multiple times with different initializations and report the average as well as variance in the compression ratio.
- Is there a tradeoff between image quality and degree of compression. What would be a good value of K for each of the two images?

We have provided you a java template "KMeans.java" which implements various image input/output operations. You have to implement the function kmeans in the template. If you want to use python, you will have to replicate the code in KMeans.java in python (will take roughly 10-15 minutes).

## What to turn in for this homework:

In a single zip file:

- A PDF report containing your write up for parts 1, 2 and 3.
- Your source code for collabortive filtering and kmeans algorithm.

Note that your program must compile and we should be able to replicate your results. Otherwise no credit will be given.