Machine Learning	Programming Assignment 6
Comp540 Spring 2015	**due 17 April 2015 at 8 pm**

Introduction

In this problem, you will implement an anomaly detection algorithm and apply it to detect failing servers on a network. In the second part, you will use collaborative filtering to build a recommender system for movies. To get started, please download the code base `pa6.zip` from Owlspace. When you unzip the archive, you will see the following files.

Name	Edit?	Read?	Description
estimateGaussian.m	Yes	Yes	Estimate the parameters of a Gaussian distribution with a diagonal covariance matrix
selectThreshold.m	Yes	Yes	Find a threshold for anomaly detection
cofiCostFunc.m	Yes	Yes	Implement the cost function for collaborative filtering
ex6.m	No	Yes	Matlab script that will run anomaly detection functions
ex6_cofi.m	No	Yes	Matlab script that will run your collaborative filtering functions
ex6data1.mat	No	No	First example dataset for anomaly detection
ex6data2.mat	No	No	Second example dataset for anomaly detection
ex6_movies.mat	No	No	Movie review dataset
ex6_movieParams.mat	No	No	Parameters provided for debugging functions
multivariateGaussian.m	No	Yes	Computes the probability density function for a Gaussian distribution
visualizeFit.m	No	No	2D plot of a Gaussian distribution and a dataset
checkCostFunction.m	No	No	Gradient checking for collaborative filtering
computeNumericalGradient.m	No	No	Numerically compute gradients
fuming.m	No	No	Function minimization routine
loadMovieList.m	No	No	Loads the list of movies into a cell-array
movie_ids.txt	No	No	List of movies
normalizeRatings.m	No	No	Mean normalization for collaborative filtering
pa6_2015.pdf	No	Yes	this document

Throughout the first part of the assignment (anomaly detection) you will be using the script `ex6.m`. For the second part on collaborative filtering, you will use `ex6_cofi.m`. These scripts set up the dataset for the problems and make calls to functions that you will write. You are only required to modify functions in other files, by following the instructions in this assignment.

Problem 1: Anomaly detection (30 points)

In this problem, you will implement an anomaly detection algorithm to detect anomalous behavior in server computers. The features measure the throughput (mb/s) and latency (ms) of response of each server. We have a dataset $\mathcal{D} = \{x^{(1)}, \ldots, x^{(m)}\}$ of unlabeled examples where $m = 307$ and each example $x^{(i)} \in \Re^2$. You suspect that the vast majority of these examples are normal (non-anomalous) examples of the servers operating normally, but there might also be some examples of servers acting anomalously within this dataset.

You will use a Gaussian model to detect anomalous examples in your dataset. You will first start on a 2D dataset that will allow you to visualize what the algorithm is doing. On that dataset you will fit a Gaussian distribution and then find values that have very low probability and hence can be considered anomalies. After that, you will apply the anomaly detection algorithm to a larger dataset with many dimensions. You will be using `ex6.m` for this part of the exercise. The first part of `ex6.m` will visualize the dataset as shown in Figure 1.

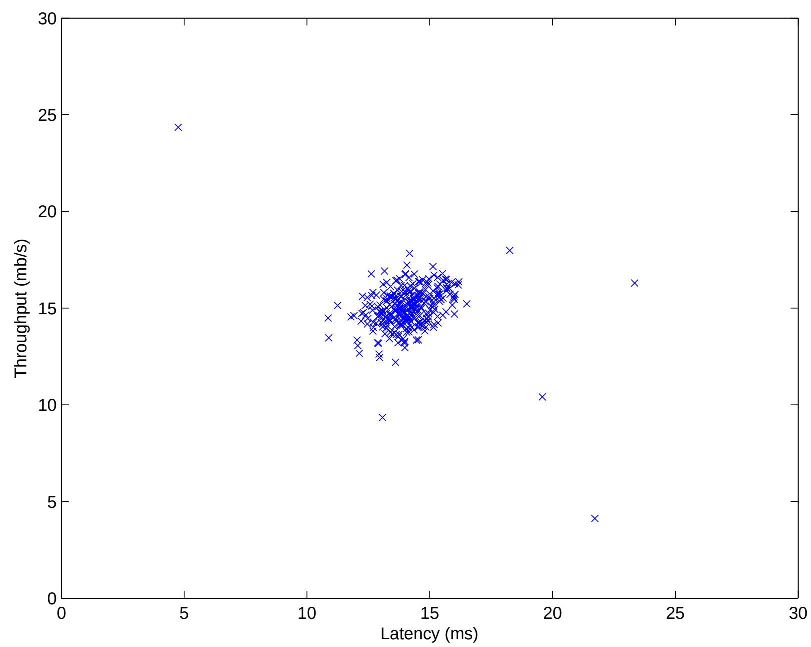

Figure 1: The first dataset

Gaussian distribution

To perform anomaly detection, you will first need to fit a model to the datas distribution. Given a training set $\{x^{(1)}, \ldots, x^{(m)}\}$ (where $x^{(i)} \in \Re^2$), you want to estimate the Gaussian distribution for each of the features x_j. For each feature $j = 1 \ldots d$, you need to find parameters μ_j and σ_j^2, that fit the data in the j^{th} dimension in each example. Recall that a univariate Gaussian distribution is given by

$$p(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} exp(-\frac{(x-\mu)^2}{2\sigma^2})$$

where μ is the mean and σ^2 is the variance.

Estimating parameters of a Gaussian distribution (15 points)

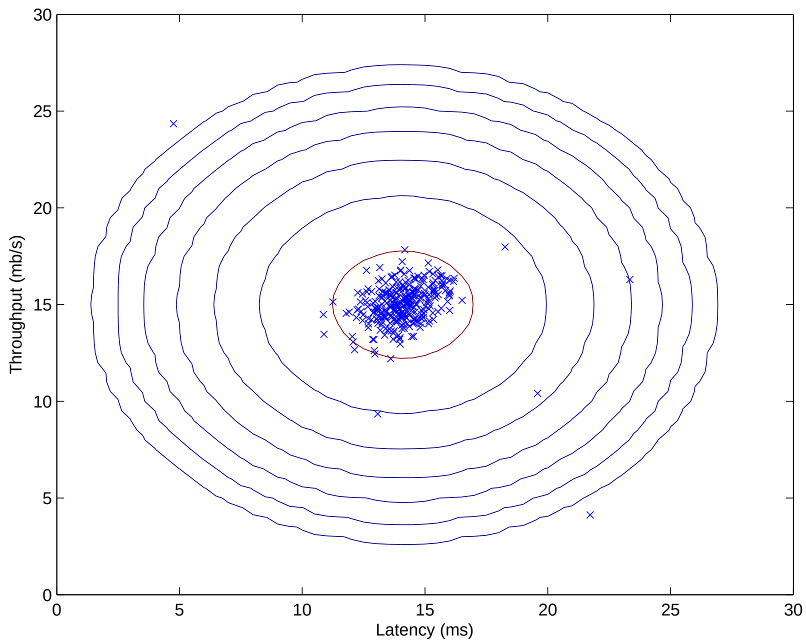

Figure 2: The Gaussian distribution contours of the distribution fit to the dataset

You can estimate the parameters, μ_j and σ_j^2 of the j^{th} feature by using the following equations. To estimate the mean, you will use:

$$\mu_j = \frac{1}{m} \sum_{i=1}^{m} x_j^{(i)}$$

To estimate the variance, you will use:

$$\sigma_j^2 = \frac{1}{m} \sum_{i=1}^{m} (x_j^{(i)} - \mu_j)^2$$

Your task is to complete the code in `estimateGaussian.m`. This function takes as input the data matrix `X` and should output an d-dimension vector `mu` that holds the mean of all the d features and another d-dimension vector `sigma2` that holds the variances of all the features. You should implement this in a vectorized way to be more efficient. Note that in Matlab, the `var` function will (by default) use $\frac{1}{m-1}$ instead of $\frac{1}{m}$ when computing σ_j^2.

Once you have completed the code in `estimateGaussian.m`, the next part of `ex6.m` will visualize the contours of the fitted Gaussian distribution. You should get a plot similar to Figure 2. From your plot, you can see that most of the examples are in the region with the highest probability, while the anomalous examples are in the regions with lower probabilities.

Selecting the threshold ϵ (15 points)

Now that you have estimated the Gaussian parameters, you can investigate which examples have a very high probability given this distribution and which examples have a very low probability. The low probability examples are more likely to be the anomalies in our dataset. One way to determine which examples are anomalies is to select a threshold based on a cross validation set. In this part of the assignment, you will implement an algorithm to select the threshold ϵ using the F1 score on a cross validation set.

You should now complete the code in `selectThreshold.m`. For this, we will use a validation set $\{(x_v^{(1)}, y_v^{(1)}), \ldots, (x_v^{(m_v)}, y_v^{(m_v)})\}$, where the label $y = 1$ corresponds to an anomalous example, and $y = 0$ corresponds to a normal example. For each example in the validation set, we will compete $p(x_v^{(i)})$. The vector of these probabilities $p(x_v^{(1)}), \ldots, p(x_v^{(m_v)})$ is passed to `selectThreshold.m` in the vector `pal`. The corresponding set of labels $y_v^{(1)}, \ldots, y_v^{(m_v)}$ is passed to the same function in the vector `yval`.

The function `selectThreshold.m` should return two values; the first is the selected threshold ϵ. If an example x has a low probability, i.e., $p(x) < \epsilon$, then it is considered to be an anomaly. The function should also return the F1 score, which tells you how well you are doing on finding the ground truth anomalies given a certain threshold. For many different values of ϵ, you will compute the resulting F1 score by computing how many examples the current threshold classifies correctly and incorrectly.

The F1 score is computed using precision (*prec*) and recall (*rec*):

$$F_1 = \frac{2 \times prec \times rec}{prec + rec}$$

You compute precision and recall by:

$$\begin{aligned} prec &= \frac{tp}{tp + fp} \\ rec &= \frac{tp}{tp + fn} \end{aligned}$$

where

- *tp* is the number of true positives: the ground truth label says it is an anomaly and our algorithm correctly classified it as an anomaly.
- *fp* is the number of false positives: the ground truth label says it is not an anomaly, but our algorithm incorrectly classified it as an anomaly.
- *fn* is the number of false negatives: the ground truth label says it is an anomaly, but our algorithm incorrectly classified it as not being anomalous.

In the provided function `selectThreshold.m`, there is already a loop that will try many different values of ϵ and select the best ϵ based on the F1 score. You can implement the computation of the F1 score using a for-loop over all the validation examples (to compute the values *tp*, *fp*, *fn*). You should see a value for epsilon of about $8.99 * 10^{-5}$.

Implementation hints: In order to compute *tp*, *fp* and *fn*, you should use a vectorized implementation rather than loop over all the examples. This can be implemented by Matlabs equality test between a vector and a single number. If you have several binary values in an n-dimensional binary vector $v \in \{0, 1\}^n$, you can find out how many values in this vector are 0 by using: `sum(v == 0)`. You can also apply a logical **and** operator to such binary vectors. For instance, let `vPredictions` be a binary vector of the size of the validation set, where the i^{th} element is 1 if your algorithm considers $x_v^{(i)}$ an anomaly, and 0 otherwise. You can then, for example, compute the number of false positives using: `fp = sum((vPredictions == 1) & (yval == 0))`.

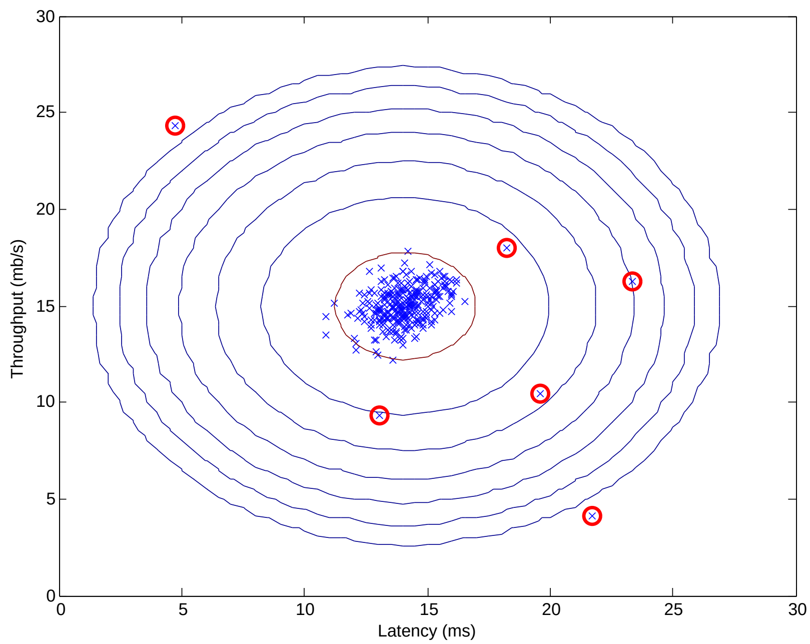

Figure 3: The classified anomalies

Once you have completed the code in `selectThreshold.m`, the next step in `ex6.m` will run your anomaly detection code and circle the anomalies in the plot as in Figure 3.

High dimensional dataset

The last part of the script `ex6.m` will run the anomaly detection algorithm you implemented on a more realistic and much harder dataset. In this dataset, each example is described by 11 features, capturing many more properties of the compute servers. The script will use your code to estimate the Gaussian parameters, evaluate the probabilities for both the training data `X` from which you estimated the Gaussian parameters, and do so for the the validation set `Xval`. Finally, it will use `selectThreshold` to find the best threshold ϵ. You should see a value epsilon of about $1.38 * 10^{-18}$, and 117 anomalies should be found.

Problem 2: Recommender systems (70 points)

In this part of the assignment, you will implement the collaborative filtering learning algorithm and apply it to a dataset of movie ratings. This dataset, called the MovieLens 100k Dataset, consists of user ratings of movies on a scale of 1 to 5. The dataset 943 users, and 1682 movies. For this part of the assignment, you will be working with the script `ex6_cofi.m`.

In the next parts of this assignment, you will implement the function `cofiCostFunc.m` that computes the collaborative filtering objective function and gradient. After implementing the cost function and gradient, you will use `fmincg.m` to learn the parameters for collaborative filtering.

Movie ratings dataset

The first part of the script `ex6_cofi.m` will load the dataset `ex6_movies.mat`, providing the variables `Y` and `R` in your Matlab environment. The matrix `Y` (a num_movies ? num_users matrix) stores the ratings $y^{(i,j)}$ for user j on movie i. The ratings are integers from 1 to 5. The matrix `R` is an binary-valued indicator matrix, where `R(i, j) = 1` if user j gave a rating to movie i, and `R(i, j) = 0` otherwise. The objective of collaborative filtering is to predict movie ratings for the movies that users have not yet rated, that is, the entries with `R(i, j) = 0`. This will allow us to recommend the movies with the highest predicted ratings to the user.

To help you understand the matrix `Y`, the script `ex6_cofi.m` will compute the average movie rating for the first movie (Toy Story) and output the average rating to the screen. Throughout this part of the problem, you will also be working with the matrices, `X` and `Theta`:

$$X = \begin{bmatrix} —(x^{(1)})^T— \\ —(x^{(2)})^T— \\ \dots \\ —(x^{(n_m)})^T— \end{bmatrix}, \Theta = \begin{bmatrix} —(\theta^{(1)})^T— \\ —(\theta^{(2)})^T— \\ \dots \\ —(\theta^{(n_u)})^T— \end{bmatrix}$$

The i^{th} row of `X` corresponds to the feature vector $x^{(i)}$ for the i^{th} movie, and the j^{th} row of `Theta` corresponds to the parameter vector $\theta^{(j)}$, for the j^{th} user. Both $x^{(i)}$ and $\theta^{(j)}$ are n-dimensional vectors. For the purposes of this exercise, you will use $n = 10$, and therefore, $x^{(i)} \in \Re^{10}$ and $\theta^{(j)} \in \Re^{10}$. Correspondingly, `X` is a $n_m \times 10$ matrix and `Theta` is a $n_u \times 10$ matrix.

Collaborative filtering learning algorithm

Now, you will start implementing the collaborative filtering learning algorithm. You will start by implementing the cost function (without regularization). The collaborative filtering algorithm in the setting of movie recommendations considers a set of n-dimensional parameter vectors $x^{(1)}, \dots, x^{(n_m)}$ and $\theta^{(1)}, \dots, \theta^{(n_u)}$, where the model predicts the rating for movie i by user j as $y^{(i,j)} = (\theta^{(j)})^T x^{(i)}$. Given a dataset that consists of a set of ratings produced by some users on some movies, you wish to learn the parameter vectors $x^{(1)}, \ldots, x^{(n_m)}, \theta^{(1)}, \ldots, \theta^{(n_u)}$ that produce the best fit (minimize the squared error).

You will complete the code in `cofiCostFunc.m` to compute the cost function and gradient for collaborative filtering. Note that the parameters to the function (i.e., the values that you are trying to learn) are `X` and `Theta`. In order to use an off-the-shelf minimizer such as `fmincg`, the cost function has been set up to unroll the parameters into a single vector `params`.

Collaborative filtering cost function

The collaborative filtering cost function (without regularization) is given by

$$J(x^{(1)}, \ldots, x^{(n_m)}, \theta^{(1)}, \ldots, \theta^{(n_u)}) = \frac{1}{2} \sum_{(i,j):R(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right)^2$$

You should now modify `cofiCostFunc.m` to return this cost in the variable `J`. Note that you should be accumulating the cost for user j and movie i only if `R(i, j) = 1`. After you have completed the function, the script `ex6_cofi.m` will run your cost function. You should expect to see an output of 22.22.

We strongly encourage you to use a vectorized implementation to compute `J`, since it will later by called many times by the optimization package `fmincg`. As usual, it might be easiest to first write a non-vectorized implementation (to make sure you have the right answer), and the modify it to become a vectorized implementation (checking that the vectorization steps dont change your algorithms output). To come up with a vectorized implementation, the following tip might be helpful: You can use the `R` matrix to set selected entries to 0. For example, R .* M will do an element-wise multiplication between `M` and `R`; since `R` only has elements with values either 0 or 1, this has the effect of setting the elements of `M` to 0 only when the corresponding value in `R` is 0. Hence, `sum(sum(R.*M))` is the sum of all the elements of `M` for which the corresponding element in `R` equals 1.

Collaborative filtering gradient

Now, you should implement the gradient (without regularization). Specifically, you should complete the code in `cofiCostFunc.m` to return the variables `X_grad` and `Theta_grad`. Note that `X_grad` should be a matrix of the same size as `X` and similarly, `Theta_grad` is a matrix of the same size as `Theta`. The gradients of the cost function are given by:

Note that the function returns the gradient for both sets of variables by unrolling them into a single vector. After you have completed the code to compute the gradients, the script `ex6_cofi.m` will run a gradient check (`checkCostFunction`) to numerically check the implementation of your gradients. If your implementation is correct, you should find that the analytical and numerical gradients match up closely.

$$\frac{\partial J}{\partial x_k^{(i)}} = \sum_{j:R(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)}) \theta_k^{(j)}$$

$$\frac{\partial J}{\partial \theta_k^{(j)}} = \sum_{i:R(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)}) x_k^{(i)}$$

Vectorizing this computation is very important for subsequent problems. You can preserve the loop over each movie and the loop over each user without adverse consequences on time. However, you need to come up with a way to all derivatives associated with the components of $x^{(i)}$ at the same time. The derivative for the feature vector of the i^{th} movie is:

$$(X_{grad}(i,:))^T = \begin{bmatrix} \frac{\partial J}{\partial x_1^{(i)}} \\ \frac{\partial J}{\partial x_2^{(i)}} \\ \dots \\ \frac{\partial J}{\partial x_n^{(i)}} \end{bmatrix} = \sum_{j:R(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)}) \theta^{(j)}$$

To vectorize the above expression, you can start by indexing into `Theta` and `Y` to select only the elements of interests (that is, those with `R(i, j) = 1`). Intuitively, when you consider the features for the i^{th} movie, you only need to be concerned about the users who had given ratings to the movie, and this allows you to remove all the other users from `Theta` and `Y`.

Concretely, you can set `idx = find(R(i, :)==1)` to be a list of all the users that have rated movie i. This will allow you to create the temporary matrices `Theta_temp = Theta(idx, :)` and `Y_temp = Y(i, idx)` that index into `Theta` and `Y` to give you only the set of users which have rated the i^{th} movie. This will allow you to write the derivatives as:

$$Xgrad(i,:) = (X(i,:) * Theta\_temp^T ? Y\_temp) * Theta\_temp$$

The vectorized computation above returns a row-vector After you have vectorized the computations of the derivatives with respect to $x^{(i)}$, you should use a similar method to vectorize the derivatives with respect to $\theta^{(j)}$ as well.

Regularized cost function

The cost function for collaborative filtering with regularization is given by

$$J(x^{(1)}, \dots, x^{(n_m)}, \theta^{(1)}, \dots, \theta^{(n_u)}) = \frac{1}{2} \sum_{(i,j):R(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})^2 + \left(\frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^{n} (\theta_k^{(j)})^2 \right) + \left(\frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^{n} (x_k^{(i)})^2 \right)$$

You should now add regularization to your original computations of the cost function, `J`. After you are done, the script `ex6_cofi.m` will run your regularized cost function, and you should expect to see a cost of about 31.34.

Regularized gradient

Now that you have implemented the regularized cost function, you should proceed to implement regularization for the gradient. You should add to your implementation in `cofiCostFunc.m` to

```
Top recommendations for you:
Predicting rating 8.5 for movie Star Wars (1977)
Predicting rating 8.4 for movie Titanic (1997)
Predicting rating 8.4 for movie Shawshank Redemption, The (1994)
Predicting rating 8.3 for movie Raiders of the Lost Ark (1981)
Predicting rating 8.2 for movie Schindler's List (1993)
Predicting rating 8.1 for movie Good Will Hunting (1997)
Predicting rating 8.1 for movie Empire Strikes Back, The (1980)
Predicting rating 8.1 for movie Usual Suspects, The (1995)
Predicting rating 8.0 for movie Wrong Trousers, The (1993)
Predicting rating 8.0 for movie Braveheart (1995)


Original ratings provided:
Rated 4 for Toy Story (1995)
Rated 3 for Twelve Monkeys (1995)
Rated 5 for Usual Suspects, The (1995)
Rated 4 for Outbreak (1995)
Rated 5 for Shawshank Redemption, The (1994)
Rated 3 for While You Were Sleeping (1995)
Rated 5 for Forrest Gump (1994)
Rated 2 for Silence of the Lambs, The (1991)
Rated 4 for Alien (1979)
Rated 5 for Die Hard 2 (1990)
Rated 5 for Sphere (1998)
```

Figure 4: Movie recommendations

return the regularized gradient by adding the contributions from the regularization terms. Note that the gradients for the regularized cost function is given by:

$$\frac{\partial J}{\partial x_k^{(i)}} = \sum_{j:R(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})\theta_k^{(j)} + \lambda x_k^{(i)}$$

$$\frac{\partial J}{\partial \theta_k^{(j)}} = \sum_{i:R(i,j)=1} ((\theta^{(j)})^T x^{(i)} - y^{(i,j)})x_k^{(i)} + \lambda \theta_k^{(j)}$$

After you have completed the code to compute the gradients, the script `ex6_cofi.m` will run another gradient check (`checkCostFunction`) to numerically check the implementation of your gradients.

Learning movie recommendations

After you have finished implementing the collaborative filtering cost function and gradient, you can now start training your algorithm to make movie recommendations for yourself. In the next part of the ex8 cofi.m script, you can enter your own movie preferences, so that later when the algorithm runs, you can get your own movie recommendations! We have filled out some values according to our own preferences, but you should change this according to your own tastes. The list of all movies and their number in the dataset can be found listed in the file `movie_idx.txt`.

Recommendations

After the additional ratings have been added to the dataset, the script will proceed to train the collaborative filtering model. This will learn the parameters `X` and `Theta`. To predict the rating of movie i for user j, you need to compute $(\theta^{(j)})^T x^{(i)}$. The next part of the script computes the ratings for all the movies and users and displays the movies that it recommends (Figure 4), according to ratings that were entered earlier in the script. Note that you might obtain a different set of the predictions due to different random initializations.

What to turn in

Please zip up all the files in the archive (including files that you did not modify) and submit it as `pa6_netid`.zip on Owlspace before the deadline. Include a PDF file in the archive that presents your plots and your discussion of results from the problems above.

Acknowledgment

These problems are from Andrew Ng's exercise on anomaly detection and collaborative filtering.