ISyE 8803 – Topics on High Dimensional Data Analytics - Exam II

- For all questions, you are required to clearly state all assumptions you make and show all necessary details of your solutions.
- You are not allowed to discuss the exam content with your fellow students or receive aid on this exam.
- You are expected to observe the Georgia Tech Honor Code throughout the exam.
- Exam is due on April 26 at 11:59 pm. Please submit your solutions via Canvas.

Question 1 (25 points)

In this question, we want to minimize $\|Ax - b\|_1$ by solving the following optimization problem using alternating direction method of multipliers (ADMM).

$$\text{minimize } \|z\|_1 \quad \text{subject to} \quad Ax - z = b \qquad (1)$$

(a) Equation (1) can be decomposed as a sum of a convex and differentiable function *f* and a convex but not differentiable function *g*. Identify these functions and write the augmented Lagrangian function with $\rho = 1$.

(b) Write the ADMM updates for x_k, z_k and u_k.

(c) In this part, you implement your own regression algorithm using your solution in part b, to perform energy efficiency analysis for different building shapes. There are 620 buildings. The buildings differ with respect to the glazing area, the glazing area distribution, and the orientation, amongst other parameters. The dataset comprises 620 samples and 8 features, and a real valued response "Heating Load". The features can be found in EnergyEfficiency.csv, the corresponding responses can be found in Response.csv. Implement your regression algorithm and report your model coefficients and the MSE of your fitted model. Note that you should standardize the data.

Question 2 (25 points)

In this problem, you build a set of different models for ozone forecasting classification task. There are two classes 1: ozone day, 0: normal day. Data were collected at the Houston, Galveston and Brazoria area. Seventy-two data attributes are extracted. The following are specifications for several most important attributes that are highly valued by Texas Commission on Environmental Quality (TCEQ). Note that you should standardize the data.

O_3- Local ozone peak prediction
Upwind - Upwind ozone background level
EmFactor - Precursor emissions related factor
Tmax - Maximum temperature in degrees F
Tb - Base temperature where net ozone production begins (50 F)
SRd - Solar radiation total for the day
WSa - Wind speed near sunrise (using 09-12 UTC forecast mode)
WSp - Wind speed mid-day (using 15-21 UTC forecast mode)

First, we are going to use logistic regression. The response variable can take the values "1" (ozone day) or "0" (normal day). We observe n independent observations $\left(y_i, x_{1i}, \dots, x_{pi}\right)$ for $i = 1, \dots, n$, where $y_i \in \{1,0\}$. The data comes from the following logistic model:

- $y_i \sim$ Bernoulli (π_i) with $P(y_i = g) = \pi_i = 1 - P(y_i = b)$
- The parameter π_i satisfies:

$$logit(\pi_i) = log\left(\frac{\pi_i}{1 - \pi_i}\right) = \beta_0 + \boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{x}$$

Or equivalently

$$\pi_i = \frac{e^{\beta_0 + \boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{x}}}{1 + e^{\beta_0 + \boldsymbol{\beta}^{\boldsymbol{T}}\boldsymbol{x}}}$$

The estimates of parameters $\boldsymbol{\beta}$ in the logistic regression model are obtained by maximum likelihood. In this case, the likelihood function is:

$$L(\boldsymbol{\beta}) = \prod_{i=1}^{n} (1 - \pi_i)^{1-y_i} \ (\pi_i)^{y_i}$$

a) Write the log-likelihood function associated with this problem in terms of the parameters $\boldsymbol{\beta}$

b) Run a logistic regression for classification on the test set. Present the coefficients obtained and the confusion matrix for the test set.

c) Use Ridge logistic regression for classification on the test set. First, write the loss function that you will minimize. Second, explain why we use Ridge regression. Third, run the model. Present the optimal tuning parameter, obtained using cross-validation, the coefficients for this parameter and the confusion matrix for the test set.

d) Use lasso logistic regression for classification on the test set. First, write the loss function that you will minimize. Second, explain why we use lasso regression. Third, run the model. Present the optimal tuning parameter, obtained using cross-validation, the coefficients for this parameter and the confusion matrix for the test set.

e) Use adaptive lasso logistic regression for classification on the test set. First, write the loss function that you will minimize. Second, explain why we use adaptive lasso regression. Third, run the model. Present the optimal tuning parameter, obtained using cross-validation, the coefficients for this parameter and the confusion matrix for the test set.

f) Finally: Which model will you select? Why?

Question 3 (25 points)

One use case of Robust PCA is separating moving objects from the background in a video sequence. A sequence of 500 images given in "background_removal.mat".

A) Plot image 1 and 350 from the input tensor.

B) Reformat data into a matrix (each image is one column)

C) Implement Robust PCA algorithm on data to separate background of surveillance video from objects moving across train tracks.

D) For image 350, plot the results of Robust PCA. So, plot the background image and the foreground object.

Question 4 (25 points)

Personalized recommendation systems are used in a wide variety of applications such as electronic commerce, social networks, web search, and more. Machine learning techniques play a key role to extract individual preference over items. In this assignment, we explore this popular business application of machine learning, by implementing a simple matrix-factorization-based recommender using gradient descent.

Suppose you are an employee in Netflix. You are given a set of ratings (from one star to five stars) from users on many movies they have seen. Using this information, your job is implementing a personalized rating predictor for a given user on unseen movies. That is, a rating predictor can be seen as a function $f: U \times I \rightarrow R$, where U and I are the set of users and items, respectively. Typically, the range of this function is restricted to between 1 and 5 (stars), which is the the allowed range of the input.

Now, let's think about the data representation. Suppose we have m users and n items, and a rating given by a user on a movie. We can represent this information as a form of matrix, namely rating matrix M. Suppose rows of M represent users, while columns do movies. Then, the size of matrix will be $m \times n$. Each cell of the matrix may contain a rating on a movie by a user. In $M_{15,47}$, for example, it may contain a rating on the item 47 by user 15. If he gave 4 stars, $M_{15,47} = 4$. However, as it is almost impossible for everyone to watch large portion of movies in the market, this rating matrix should be very sparse in nature. Typically, only 1% of the cells in the rating matrix are observed in average. All other 99% are missing values, which means the corresponding user did not see (or just did not provide the rating for) the corresponding movie. Our goal with the rating predictor is estimating those missing values, reflecting the user's preference learned from available ratings.

Our approach for this problem is matrix factorization. Specifically, we assume that the rating matrix M is a low-rank matrix. Intuitively, this reflects our assumption that there is only a small number of factors (e.g, genre, director, main actor/actress, released year, etc.) that determine like or dislike. Let's define r as the number of factors. Then, we learn a user profile $U \in R^{m \times r}$ and an item profile $V \in R^{n \times r}$. (Recall that m and n are the number of users and items, respectively.) We want to approximate a rating by an inner product of two length r vectors, one representing user profile and the other item profile. Mathematically, a rating by user u on movie i is approximated by

$$M_{u,i} \approx \sum_{k=1}^{r} U_{u,k} V_{i,k} \quad (1)$$

We want to fit each element of U and V by minimizing squared reconstruction error over all training data points. That is, the objective function we minimize is given by

$$E(U,V) = \sum_{(u,i)\in M} (M_{u,i} - U_u^T V_i)^2 = \sum_{(u,i)\in M} (M_{u,i} - \sum_{k=1}^{r} U_{u,k} V_{i,k})^2 \quad (2)$$

where U_u is the uth row of U and V_i is the ith row of V. We observe that this looks very similar to the linear regression. Recall that we minimize in linear regression:

$$E(\theta) = \sum_{i=1}^{m} (Y^i - \theta^T x^i)^2 = \sum_{i=1}^{m} (Y^i - \sum_{k=1}^{r} \theta_k x_k^i)^2 \quad (3)$$

where m is the number of training data points. Let's compare (2) and (3). $M_{u,i}$ in (2) corresponds to Y^i in (3), in that both are the observed labels. $U_u^T V$ in (2) corresponds to $\theta^T x^i$ in (3), in that both are our estimation with our model. The only difference is that both U and V are the parameters to be learned in (2), while only θ is learned in (3). This is where we personalize our estimation: with linear regression, we apply the same θ to any input x^i, but with matrix factorization, a different profile U_u are applied depending on who is the user u. As U and V are interrelated in (2), there is no closed form solution, unlike linear regression case. Thus, we need to use gradient descent:

$$U_{v,k} \leftarrow U_{v,k} - \mu \frac{\partial E(U,V)}{\partial U_{v,k}}$$

$$V_{j,k} \leftarrow V_{j,k} - \mu \frac{E(U,V)}{\partial V_{j,k}} \quad (4)$$

where μ is a hyper-parameter deciding the update rate. It would be straightforward to take partial derivatives of $E(U,V)$ in (2) with respect to each element $U_{v,k}$ and $V_{j,k}$. Then, we update each element of U and V using the gradient descent formula in (4).

(a) Derive the update formula in (4) by solving the partial derivatives.

(b) To avoid overfitting, we usually add regularization terms, which penalize for large values in U and V . Redo part (a) using the regularized objective function below.

$$E(U,V) = \sum_{(u,i)\in M} (M_{u,i} - \sum_{k=1}^{r} U_{u,k} V_{i,k})^2 + \lambda \sum_{u,k} U_{u,k}^2 + \lambda \sum_{i,k} V_{i,k}^2$$

(λ is a hyper-parameter controlling the degree of penalization.)

(c) Implement myRecommender.m by filling the gradient descent part.

You are given a skeleton code **myRecommender.m**. Using the training data **rateMatrix**, you will implement your own recommendation system of rank **lowRank**. The only file you need to edit and submit is **myRecommender.m**. In the gradient descent part, repeat your update formula in (b), observing the average reconstruction error between your estimation and ground truth in training set. You need to set a stopping criteria, based on this reconstruction error as well as the maximum number of iterations. You should play with several different values for μ and λ to make sure that your final prediction is accurate. Formatting information is here:

Input

• **rateMatrix**: training data set. Each row represents a user, while each column an item. Observed values are one of {1,2,3,4,5}, and missing values are 0.

• **lowRank**: the number of factors (dimension) of your model. With higher values, you would expect more accurate prediction.

Output

• U: the user profile matrix of dimension user count × low rank.

• V: the item profile matrix of dimension item count × low rank.

Note

• Upload your **myRecommender.m** implementation file. (Do not copy and paste your code in your report. Be sure to upload your **myRecommender.m** file.)

• To test your code, try to run **recommender_rmse.m**. You may have noticed that the code prints both training and test error, in RMSE (Root Mean Squared Error), defined as follows: $\sum_{(u,i)\in M} (M_{u,i} - f(u,i))^2$ where $f(u,i)$ is your estimation, and the summation is over the training set or testing set, respectively.

• Note that we provide **recommender_rmse.m** just to help you evaluate your code easily. You are not expected to alter or submit this to us. In other words, we will not use this file when we grade your submission. The only file we take care of is **myRecommender.m**.

(d) Report

In your report, show the performance (**RMSE**) both on your training set and test set, with varied **lowRank**. (The default is set to 1, 3, and 5, but you may want to vary it further.) Discuss what you observe with varied low rank. Also, briefly discuss how you decided your hyper-parameters (μ, λ).