CS 7641 CSE/ISYE 6740 Homework 5

Yao Xie

Deadline: Sunday April 15, 11:55pm

- Submit your answers as an electronic copy on T-square.
- No unapproved extension of deadline is allowed. Zero credit will be assigned for late submissions. Email request for late submission may not be replied.
- For typed answers with LaTeX (recommended) or word processors, extra credits will be given. If you handwrite, try to be clear as much as possible. No credit may be given to unreadable handwriting.
- Explicitly mention your collaborators if any.

1 Entropy and mutual information [20 points]

In the lecture you became familiar with the concept of entropy for one random variable and mutual information. For a pair of discrete random variables X and Y with the joint distribution $p(x, y)$, the *joint entropy* $H(X, Y)$ is defined as

$$H(X, Y) = -\sum_{x \in X} \sum_{y \in Y} p(x, y) \log p(x, y) \tag{1}$$

which can also be expressed as

$$H(X, Y) = -\mathbb{E}[\log p(X, Y)] \tag{2}$$

Let X and Y take on values $x_1, x_2, ..., x_r$ and $y_1, y_2, ..., y_s$ respectively. Let Z also be a discrete random variable and $Z = X + Y$.

(a) Prove that $H(X, Y) \leq H(X) + H(Y)$

(b) If X and Y are independent, i.e. $P(X, Y) = P(X)P(Y)$, then $H(X, Y) = H(X) + H(Y)$ [4 pts]

(c) Show that $I(X; Y) = H(X) + H(Y) - H(X, Y)$.

(d) Show that $H(Z|X) = H(Y|X)$. Argue that when X, Y are independent, then $H(X) \leq H(Z)$ and $H(Y) \leq H(Z)$. Therefore, the addition of *independent* random variables add uncertainty.

(e) Under what conditions does $H(Z) = H(X) + H(Y)$.

2 Ridge regression and lasso [30 points].

2.1 Recover an image using compressed sensing

In this problem, you will consider choosing the tuning parameters for both ridge regression and the lasso, using 10-fold cross-validation. First download the data cs.mat.

We begin with a true image image of dimension 50×50. You can plot it first. This image is truly sparse, in the sense that 2084 of its pixels have a value of 0, while 416 pixels have a value of 1. You can think of this image as a toy version of an MRI image that we are interested in collecting.

Suppose that, because of the nature of the machine that collects the MRI image, it takes a long time to measure each pixel value individually, but it's faster to measure a linear combination of pixel values. We measure $n = 1300$ linear combinations, with the weights in the linear combination being random, in fact, independently distributed as $\mathcal{N}(0, 1)$. These measurements are given by the entries of the vector `A*double(img(:))` in our MATLAB code. Because the machine is not perfect, we don't get to observe this directly, but we see a noisy version of this. Hence, in terms of our code, we observe `y = A*double(img(:)) + 5*randn(1300,1)` Now the question is: can we model y as a linear combination of the columns of x to recover some coefficient vector that is close to the image? Roughly speaking, the answer is yes. Key points here: although the number of measurements $n = 1300$ is smaller than the dimension $p = 2500$, the true image is sparse, and the weights in a linear combination are i.i.d normal. This is the idea behind the field of *compressed sensing*.

The file model-selection.m is setup to perform ridge regression of y on x, and the lasso of y on x, with the tuning parameter for each method selected by cross-validation. You will fill in the missing pieces. It's helpful to read through the whole file to get a sense of what's to be accomplished. Try to understand all the parts, even if it doesn't seem related to what you have to fill in; this should be good practice for working with MATLAB in the future, etc. You may build your code based on model-selection.m.

Plot the cross-validation error curves for each of ridge regression and the lasso. For both ridge regression and the lasso, what value of λ has a smaller minimum cross-validation error?

Also plot the recovered images under the optimal choices of regularizing parameters for the ridge regression and lasso, respectively. Which one do you think recovers a better image?

2.2 House price dataset.

The HOUSES dataset contains a collection of recent real estate listings in San Luis Obispo county and around it. The dataset is provided in RealEstate.csv.

The dataset contains the following fields:

- MLS: Multiple listing service number for the house (unique ID).
- Location: city/town where the house is located. Most locations are in San Luis Obispo county and northern Santa Barbara county (Santa Maria-Orcutt, Lompoc, Guadelupe, Los Alamos), but there some out of area locations as well.
- Price: the most recent listing price of the house (in dollars).
- Bedrooms: number of bedrooms.
- Bathrooms: number of bathrooms.
- Size: size of the house in square feet.
- Price/SQ.ft: price of the house per square foot.
- Status: type of sale. Thee types are represented in the dataset: Short Sale, Foreclosure and Regular.

*Please ignore the "Location" variable below. It is a categorical variable. (Extra hint: To deal with categorical variable, we typically will convert them using "one-hot" keying and then decide to include "Location" or not, using group lasso. But you are not required to do this for this question.)

1. Fit ridge regression model to predict Price using remaining factors (except Status), for each of the three types of sales: Short Sale, Foreclosure and Regular, respectively. Find optimal λ using cross-validation.

2. Using lasso to select the 2 leading factors for Price, for each of the three type of sales, for each of the three types of sales: Short Sale, Foreclosure and Regular, respectively. Report the factors you found. Find optimal λ using cross-validation.

You may use built in MATLAB functions.

3 Ridge Regression [10 pts]

For linear regression, it is often assumed that $y = \theta^\top \mathbf{x} + \epsilon$ where $\theta, \mathbf{x} \in \mathbb{R}^m$ by absorbing the constant term, and $\epsilon \sim \mathcal{N}(0, \sigma^2)$ is a Gaussian random variable. Given n i.i.d samples $(\mathbf{x}^1, y^1), ..., (\mathbf{x}^n, y^n)$, we define $\mathbf{y} = (y^1, ..., y^n)^\top$ and $X = (\mathbf{x}^1, ..., \mathbf{x}^n)^\top$. Thus, we have $\mathbf{y} \sim \mathcal{N}(X\theta, \sigma^2 I)$. Show that the ridge regression estimate is the mean of the posterior distribution under a Gaussian prior $\theta \sim \mathcal{N}(0, \tau^2 I)$. Find the explicit relation between the regularization parameter λ in the ridge regression estimate of the parameter θ, and the variances σ^2, τ^2.

4 Programming: Recommendation System [40 pts]

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 $f : \mathcal{U} \times \mathcal{I} \to \mathbb{R}$, where $\mathcal{U}$ and $\mathcal{I}$ 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

$$M \in \mathbb{R}^{m \times n}$$

is a low-rank matrix. (Recall that m and n are the number of users and items, respectively.) 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 $U \in \mathbb{R}^{m \times r}$ and an item profile $V \in \mathbb{R}^{n \times r}$:

$$M = UV^T.$$

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

$$M_{u,i} \approx \sum_{k=1}^{r} U_{u,k} V_{i,k}. \tag{3}$$

We want to fit each element of U and V by minimizing squared reconstruction error over all training data points. Suppose we only observe entries at a subset of locations $\mathcal{A}$. Denote the uth row of U as U_u^T (hence U_u is a r-dimensional column vector), and the ith row of V as V_i^T (hence V_i is also a r-dimensional vector), $u = 1, 2, \ldots, m$, $i = 1, \ldots, n$. That is, the objective function we minimize is given by

$$E(U, V) = \sum_{(u,i)\in\mathcal{A}} (M_{u,i} - U_u^T V_i)^2 = \sum_{(u,i)\in\mathcal{A}} (M_{u,i} - \sum_{k=1}^{r} U_{u,k} V_{i,k})^2. \tag{4}$$

We observe that this looks very similar to the linear regression. Recall that we minimize in linear regression:

$$E(\theta) = \sum_{i=1}^{m} (Y^i - \theta^T x^i)^2 = \sum_{i=1}^{m} (Y^i - \sum_{k=1}^{r} \theta_k x_k^i)^2 \tag{5}$$

where m is the number of training data points. Let's compare (4) and (5). $M_{u,i}$ in (4) corresponds to Y^i in (5), in that both are the observed labels. $U_u^T V_i$ in (4) corresponds to $\theta^T x^i$ in (5), 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 (4), while only θ is learned in (5). 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 (4), there is no closed form solution, unlike linear regression case. Thus, we need to use gradient descent:

$$U_{v,k} \leftarrow U_{v,k} - \mu \frac{\partial E(U,V)}{\partial U_{v,k}}, \qquad V_{j,k} \leftarrow V_{j,k} - \mu \frac{\partial E(U,V)}{\partial V_{j,k}}, \tag{6}$$

where μ is a hyper-parameter deciding the update rate. It would be straightforward to take partial derivatives of $E(U, V)$ in (4) 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 (6).

(a) Derive the update formula in (6) by solving the partial derivatives. [10 pts]

(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. [5 pts]

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

(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 $\times$ low rank.
- **V**: the item profile matrix of dimension item count $\times$ low rank.

Evaluation

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 `homework3.m`. You may have noticed that the code prints both training and test error, in RMSE (Root Mean Squared Error), defined as follows:

$$\sum_{(u,i)\in\Omega} (M_{u,i} - f(u,i))^2$$

where $f(u,i)$ is your estimation, and the summation is over the training set or testing set, respectively. For the grading, we will use another set-aside testing set, which is not released to you. If you observe your test error is less than 1.00 without cheating (that is, training on the test set), you may expect to see the similar performance on the unseen test set as well.

Note that we provide `homework3.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 `myRecommender3.m`.

Grading criteria:

- Your code should output U and V as specified. The dimension should match to the specification.
- We will test your output on another test dataset, which was not provided to you. The test RMSE on this dataset should be at least 1.05 to get at least partial credit.
- We will measure elapsed time for learning. If your implementation takes longer than 3 minutes for rank 5, you should definitely try to make your code faster or adjust parameters. Any code running more than 5 minutes is not eligible for credit.
- Your code should not crash. Any crashing code will be not credited.

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).

Note

- Do not print anything in your code (e.g, iteration 1 : err=2.4382) in your final submission.
- Do not alter input and output format of the skeleton file. (E.g, adding a new parameter without specifying its defalut value) Please make sure that you returned all necessary outputs according to the given skeleton.
- Please do not use additional file. This task is simple enough that you can fit in just one file.
- Submit your code with the best parameters you found. We will grade without modifying your code. (Applying cross-validation to find best parameters is fine, though you do not required to do.)
- Please be sure that your program finishes within a fixed number of iterations. Always think of a case where your stopping criteria is not satisfied forever. This can happen anytime depending on the data, not because your code is incorrect. For this, we recommend setting a maximum number of iteration in addition to other stopping criteria.

Grand Prize

Similar to the Netflix competition held in 2006 to 2009, the student who achives the lowest RMSE on the secret test set will earn the Grand Prize. We will give extra 10 bonus points to the winner, and share the student's code to everyone. (Note that the winner should satisfy all other grading criteria perfectly, including answer sanity check and timing requirement. Otherwise, the next student will be considered as the winner.)