STAD80 Analysis of Big Data **Assignment: 2**

High Dimensional Regression and Classification

Lecturer: Qiang Sun *Email: qsun@utstat.toronto.edu*

Due Date: This assignment is due on Midnight Sunday, February 23th by 11:59PM.

Submission: Please submit the homework (.rmd, .tex, .pdf) on Quercus. Please also bring a hard copy to Dr. Yicheng Zeng (Office: IC469) at 13:00 on 02/24 (forgrading purposes). He will be there at IC469 until 16:00.

Q1. A Regression by Any Other Form Will Predict Just As Sweetly. (20 points)

Question 1: Constrained and Unconstrained Optimization: Suppose we have $\mathbf{Y} \in \mathbb{R}_{n\times 1}$ and $\mathbf{X} \in \mathbb{R}_{n\times p}$. In this question, we explore the relationship between the following two optimization problems

$$\underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\text{minimize}} \quad \frac{1}{2n}\|\mathbf{Y}-\mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_1 \tag{0.1}$$

and

$$\begin{aligned} &\underset{\boldsymbol{\beta}\in\mathbb{R}^p}{\text{minimize}} && \frac{1}{2n}\|\mathbf{Y}-\mathbf{X}\boldsymbol{\beta}\|_2^2 \\ &\text{subject to} && \|\boldsymbol{\beta}\|_1 \le C \end{aligned} \tag{0.2}$$

Show that for any positive λ in (0.1), there exists a constant C such that (0.2) shares the same minimizer with (0.1). Give comments on this conclusion considering the L1-ball and likelihood level curves you draw for Lasso in Question 1.

(Hint 1: Suppose $\widehat{\boldsymbol{\beta}}$ is a minimizer of (0.1). Choose the constant C to be $C = \|\widehat{\boldsymbol{\beta}}\|_1$. Use a contradiction argument to show that $\widehat{\boldsymbol{\beta}}$ must be the minimizer of (0.2) as well. Let us assume the minimizer to (0.2) is unique.
Hint 2: What must be true if $\widehat{\boldsymbol{\beta}}$ were not a minimizer of (0.2)? How would this lead to a contradiction in your assumptions?)

Question 2: Regularized Regression Consider the elastic-net optimization problem (Find out what it is using google or other resources! Basically elstic-net= ℓ_1 penalty + ℓ_2 penalty,) for pre-given $\mathbf{y}$, $\mathbf{X}$, α and λ:

$$\min_{\boldsymbol{\beta}}\Big\{\|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda[\alpha\|\boldsymbol{\beta}\|_2^2 + (1-\alpha)\|\boldsymbol{\beta}\|_1]\Big\}. \tag{0.3}$$

Show how one can turn this into a standard lasso problem using an augmented version of $\mathbf{X}$ and $\mathbf{y}$. That is, add columns and/or rows to $\mathbf{X}$ and $\mathbf{y}$ with specific values to become $\widetilde{\mathbf{X}}$ and $\widetilde{\mathbf{y}}$ so the optimal value of (0.3) matches that of the lasso optimization problem:

$$\min_{\widetilde{\boldsymbol{\beta}}}\Big\{\|\widetilde{\mathbf{y}}-\widetilde{\mathbf{X}}\boldsymbol{\beta}\|_2^2 + \widetilde{\lambda}\|\boldsymbol{\beta}\|_1\Big\}. \tag{0.4}$$

[This question shows you how to reformulate elastic-net regularization as a special case of lasso regularization.]

Q2. Poisson Regression (15 points)

In logistic regression, the conditional distribution of a binary output Y is Bernoulli:

$$Y|\boldsymbol{X} \sim Ber(\theta(\boldsymbol{X})),$$

where the Bernoulli parameter is related to $\boldsymbol{\beta}^T\boldsymbol{X}$ by the logit transformation:

$$logit(\theta(\boldsymbol{X})) \equiv \log\left(\frac{\theta(\boldsymbol{X})}{1-\theta(\boldsymbol{X})}\right) = \boldsymbol{\beta}^T\boldsymbol{X}.$$

Now consider the case where the output Y is a "count" that can take any non-negative integer value $0, 1, 2, ...$ (for example, the number of daily hits on a web server). Then the standard model for Y is the Poisson. Recall that the probability mass function of the Poisson distribution with parameter $\lambda > 0$ is

$$P(Y = k|\lambda) = \frac{\lambda^k}{k!}e^{-\lambda}.$$

In *Poisson regression* the conditional distribution of Y given $\boldsymbol{X}$ is

$$Y|\boldsymbol{X} \sim Poisson(\lambda(\boldsymbol{X})),$$

where $\log\lambda(\boldsymbol{X}) = \boldsymbol{\beta}^T\boldsymbol{X}$.

(a) For data $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_n, y_n)\}$ where $\mathbf{x}_i \in \mathbb{R}^p$ and $y_i \in \mathbb{N} = \{0, 1, 2, ...\}$, write down the (conditional) log-likelihood function $\ell(\boldsymbol{\beta})$ under the above Poisson regression model.

(b) For logistic regression and Poisson regression, show that at the MLE $\widehat{\boldsymbol{\beta}}$

$$\sum_{i=1}^{n} y_i\mathbf{x}_i = \sum_{i=1}^{n} E_{\widehat{\beta}}[Y|\boldsymbol{X} = \mathbf{x}_i]\mathbf{x}_i.$$

Q3. SVM Theory (25 points)

Suppose we have i.i.d data $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, where $\mathbf{x}_i$ is a d-by-1 numeric vector and $y_i = 1$ or -1. Support Vector Machine (SVM) solves the following optimization problem.

$$\min_{\boldsymbol{\beta},b} \frac{1}{n}\sum_{i=1}^{n}[1 - y_i(\boldsymbol{\beta}^T\mathbf{X}_i - b)]_+ + \lambda\|\boldsymbol{\beta}\|_2^2 \tag{0.5}$$

where $[x]_+ := \max\{x, 0\}$ and $\lambda > 0$. Here, we explore the geometric meaning behind this optimization problem.

Question 1: Geometric Interpretation (5 Points)

The geometric motivation of SVM is to select two parallel hyperplanes in a way that they separate the data and there are no points between them, and then try to maximize their distance. Any two parallel hyperplanes in $\mathbb{R}^d$ can be expressed as two equations below.

$$\boldsymbol{\beta}^T\mathbf{x} - b = 1$$

and

$$\boldsymbol{\beta}^T\mathbf{x} - b = -1,$$

where $\boldsymbol{\beta} \in \mathbb{R}^d$ and b are constants. Show the distance between these two hyperplanes is $2/\|\boldsymbol{\beta}\|_2$.

(Hint 1: Two vectors $\mathbf{y}, \mathbf{z} \in \mathbb{R}^d$ are perpendicular if $\mathbf{y}^T\mathbf{z} = 0$.)
(Hint 2: Note that for a hyperplane $\{\mathbf{x} \in \mathbb{R}^d : \boldsymbol{\beta}^T\mathbf{x} = C\}$, the vector $\boldsymbol{\beta}$ is perpendicular to the hyperplane. We often call $\boldsymbol{\beta}$ the normal vector of the hyperplane.)
(Hint 3: Consider two parallel hyperplanes. Given a point $\mathbf{x}_0$ on one hyperplane, find another point $\mathbf{x}_1$ on the other hyperplane such that $(\mathbf{x}_1 - \mathbf{x}_0)$ is parallel to $\boldsymbol{\beta}$. Then $\|\mathbf{x}_1 - \mathbf{x}_0\|_2$ is the distance between the two hyperplanes.)

Question 2: Simple Reformulation of SVM (10 Points)

(a) We can therefore implement the idea of SVM by solving the following optimization problem.

$$\begin{aligned} \min_{\boldsymbol{\beta}, b} \quad & \frac{1}{2}\|\boldsymbol{\beta}\|_2^2 \\ \text{subject to} \quad & y_i(\boldsymbol{\beta}^T\mathbf{x}_i - b) \geq 1, 1 \leq i \leq n \end{aligned} \tag{0.6}$$

In a few sentences, answer the following questions:

- Why do we wish to minimize the objective function $\frac{1}{2}\|\boldsymbol{\beta}\|_2^2$?
- For each datapoint $(\mathbf{x}_i, y_i)$, why do we use the constraint $y_i(\boldsymbol{\beta}^T\mathbf{x}_i - b) \geq 1$? Analyze the case where $y_i = 1$ and $y_i = -1$ separately.

(b) However, sometimes data points cannot be linearly separated, which means there are no such two hyperplanes that can separate data points without error. Construct such a case where $d = 2$ with at least three data points where (0.6) has no feasible solutions, i.e. there is no $\boldsymbol{\beta}$ and b to satisfy the constraint of (0.6).

Question 3: General Reformulation of SVM (10 Points)

For data that cannot be linearly separated, we introduce non-negative slack variables ζ_i for each data point, which serves as a measure of misclassification degree, and we instead consider the following optimization problem.

$$\begin{aligned} \min_{\boldsymbol{\beta}, b, \{\zeta_i\}_{i=1}^n} \quad & C\sum_{i=1}^n \zeta_i + \frac{1}{2}\|\boldsymbol{\beta}\|_2^2 \\ \text{subject to} \quad & \zeta_i \geq 1 - y_i(\boldsymbol{\beta}^T\mathbf{x}_i - b) \\ & \zeta_i \geq 0 \\ & \text{for all } 1 \leq i \leq n \end{aligned} \tag{0.7}$$

This method is called the soft margin method, which essentially means to optimize a trade-off between a large margin and a small misclassification error.

Assume that, for some given λ parameter, $\widehat{\boldsymbol{\beta}}$ and $\widehat{b}$ optimizes (0.5). Show that there exists some constant C and $\left\{\widehat{\zeta}_i\right\}_{i=1}^n$ such that $\widehat{\boldsymbol{\beta}}$, $\widehat{b}$, and $\left\{\widehat{\zeta}_i\right\}_{i=1}^n$ is a feasible set of solutions that optimizes (0.7).

Q4. Big Data. Big Profit. (30 points)

Behind every successful website, there are the advertisements that paid for its growth. Online advertising represent a major source of traffic as well as income for modern businesses. The product may be exceptional, but it is the advertisements that bring the users and (especially for free services) pay the bills.

At the center of this industry, the advertisement exchanges that auction out advertisement spaces as a website loads and demand side platforms (DSPs) that compete to these spaces up.

Enter you. A young, ambitious data scientist at one of these DSPs: iPinYou. You are ready to show them what you got. Your first assignment? Design a Real-Time Bidding (RTB) algorithm that, for any given ad space, predicts how many clicks it may get and decides how much to bid for it.

You will be using the `glmnet` package with the iPinYou dataset[1], which provides you with information about previous auctions including properties of the space (height, width, etc.), information about the website viewer, and the bids by iPinYou and competing exchanges.

Install the `glmnet` package. Load the bidding data `q1.RData` into R. You can use `ls()` to see the objects that are loaded. Let's get started! You have a career to build!

Question 1: Click Me! Click Me! (10 points) Our first task is to predict whether a user will click on a certain add. We consider only the `Region`, `City`, `AdX`, `Domain`, `Key_Page`, `Ad_Vis`, `Ad_Form`, `Ad_Width`, `Ad_Height` and `Floor_Price` columns as our features, and we're interested in predicting `Click`. We treat the first 7 relevant columns (`Region` to `Ad_Form`) as categorical features, and we will use multiple indicator columns to express each of these features. Please see the following hint for the specific instructions.

(**Hint:** Indicator columns are columns whose entries can only be ones or zeros. We usually use multiple indicator columns to represent a single categorical feature. If the feature has k different categories, you need $(k-1)$ indicator columns to express it. To illustrate that, consider the feature `Region` that has three categories: Region 1, Region 3, and Region 6. You would use two indicators to represent these three regions: Region 3 would correspond to "10" and Region 6 to "01". The entries "00" would then implicitly correspond to Region 1. The reason why we use indicator columns for categorical data is that unlike numeric features, numbers in categorical features do not represent magnitude but only difference. For example, if we stick to using the original column for the categorical feature `Region`, it does not make any sense that Region 6 is "bigger" than Region 1. In your code, you may choose the implicit category for each categorical variable, but please indicate your choice in the write-up.)

The last three columns (`Ad_Width`, `Ad_Height` and `Floor_Price`) are numeric data (as opposed to categorical data). Standardize all three columns in the following way: for each column, subtract the mean value of that column and then divide by the standard deviation of that column. This will set the transformed column to have a mean of 0 and standard deviation of 1.

You should have 21 training columns when you're done (2 columns for `Region`, 5 columns for `City`, etc. This does not include the `Click` column.) Lastly, convert the `Click` column to have value 1 if there is at least one click, 0 if there are no clicks at all. After this preprocessing step, do the following steps.

- **Part a:** Fit the model on the training data using the `glmnet` function to predict if there is a click or not. You should predict 0 if you think there is not going to be a click, and 1 if you think there will be. Plot the regularization paths of Lasso and Ridge under the Binomial family. An example of the Lasso's regularization path is shown in Figure 1. See Section A for Under-the-Hood details on the objective function that the package `glmnet` aims to optimize under the Binomial family.
(**Note:** In `glmnet` and, later, in `cv.glmnet`, please use `standardize=FALSE` since we already standardized manually.)
- **Part b:** Based on the 2 graphs you just plotted, which features seem to be more important than others?
(**Note:** You may choose your own criteria for how to identify important features. However, you must explain these choices clearly in your write-up.)
- **Part c:** For both Lasso and Ridge regression, use a 5-fold cross validation (via the `cv.glmnet` function) to determine the tuning parameter λ. Mark the corresponding L1-norm on your plot made in Part a[2]. Also plot

[1] This dataset is a slightly simplified version of the one used in the Real-Time Bidding (RTB) algorithm competition held in 2013.

[2] This is a bit statistically meaningless for Ridge regression since Ridge regression constrains the L2-norm, not the L1-norm. Unfortunately, there's no easy way to set the x-axis to be the L2-norm.

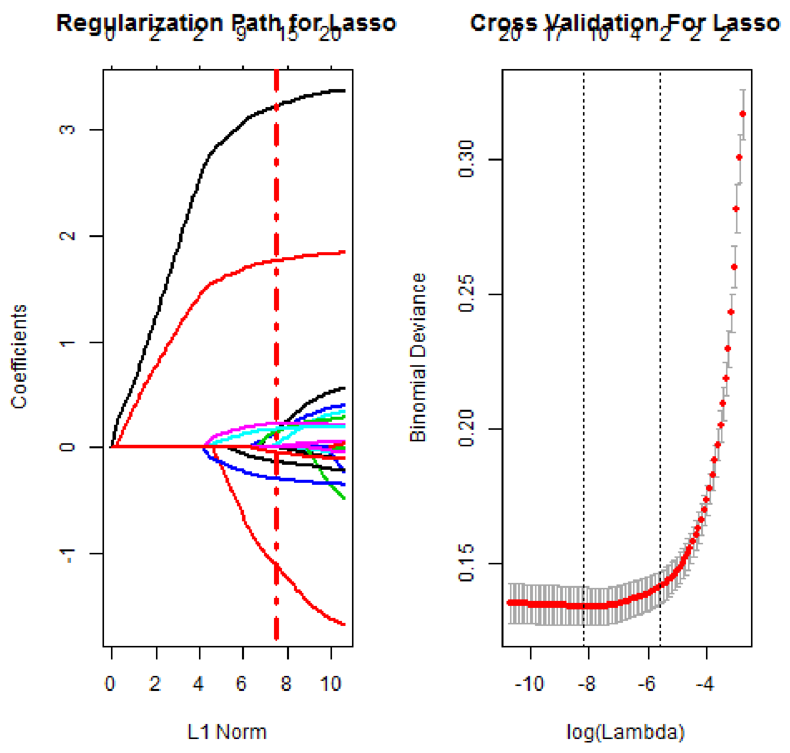


Figure 1: An (truncated) example of what your coefficients (left) and cross-validation (right) might look like. The red dotted line in the coefficient graph marks the L1-norm chosen by cross validation, and left-most lambda in the cross-validation graph should correspond to this L1-norm. Your solution to this problem will have 4 plots similar to the left plot, 2 plots similar to the right plot.

the cross validation. Your graphs will roughly look like the form in Figure 1 above. Using a few sentences, interpret these four graph (from Part a and Part c) and give some intuition on why models with larger degrees of freedom do not necessarily do better in the cross validation.

- **Part d:** Evaluate both models by using the testing data. Record each classification test error by reporting the prediction accuracy for data points where (in truth) $y_i = 1$ and for data points where $y_i = 0$. (For each of the two models, you will be reporting two numbers.)

You will have a total of 4 plots for this question (2 for normal `glmnet`, 2 for using cross-validation). [This question helps you understand how `glmnet` works.]

Question 2: Know The Enemy: Unfortunately, there is another DSP bidding for the same spaces as you. To give yourself an edge, you decide to predict how much the other exchange will bid. At the same time, you get to review exactly how the Lasso and Ridge regression "regularizes" our statistical model. Fun!

We consider only the `AdX` (the exchange of original for the ad space) and `iPinYou_Bid` columns (what iPinYou had bid on the space) as our features, and we're interested in predicting `Comp_Bid`[3] (the bid by the competitor). Standardize all three columns in the following way: for each column, subtract the mean value of that column and then

[3]This linear regression does not make complete since `AdX` is a categorical variable, but as we will see, it carries some predictive power if we "pretend" it's a numerical variable.

divide by the standard deviation of that column. This will set the transformed column to have a mean of 0 and standard deviation of 1. After the transformation, our linear relationship is

$$\text{Comp\_Bid} = \beta_1 \text{AdX} + \beta_2 \text{iPinYou\_Bid} + \epsilon$$

for a Gaussian noise term ϵ. Here, β_1, β_2 are two coefficients. Notice that since we standardize our variables, we no longer have an intercept.

Follow these steps:

- Determine the MLE coefficients of the linear regression. That is, use the `lm` function to compute the linear regression coefficients.
- Determine the Lasso coefficients of the linear regression. That is, use the `glmnet` function to compute the linear regression coefficients under a Gaussian model. Pick the Lasso coefficients that have half the L1-norm of the MLE coefficients. See Section A for Under-the-Hood details.
- For both a discretized grid of values for $\beta_1, \beta_2 \in [-.5, 1]$, compute the mean square error (MSE) according to each pair of β_1 and β_2. Generate the discretized grid by

  ```
  beta1 = seq(-.5,1,length.out=100)
  beta2 = seq(-.5,1,length.out=100)
  ```

  and recall that for response y_i and data $\mathbf{x}_i = \{x_{i,1}, x_{i,2}\}$ over n data points, the MSE is calculated with

  $$\text{MSE}(\beta_1, \beta_2) = \sum_{i=1}^{n} (y_i - (\beta_1 x_{i,1} + \beta_2 x_{i,2}))^2. \tag{0.8}$$

  (**Note:** This step takes a large amount of memory. If this causes problems like freezing or crashing, try closing R and restarting.)
- Using the `contour` (built-in) function in `R`, plot level curves of the MSE, and place the MLE coefficients and Lasso coefficients on this plot.
- Draw the L1-ball such that the Lasso coefficients sit on the edge of the L1-ball (diamond-shaped).

Do the same steps but for the Ridge coefficients (and picking the coefficients that have half the L2-norm of the MLE coefficients), and instead of an L1-ball, draw an L2-ball (circle-shaped). Your plots will look like Figure 2.

Answer the following questions: In relationship to the level curves, where do to the MLE coefficients sit? What does this say about the MSE of the MLE coefficients? In relationship to the level curves and the L1-ball (or L2-ball), where do the Lasso (or Ridge) coefficients sit? What does this say about the MSE of the Lasso and Ridge coefficients? Lastly, based on our geometric representation of the L1- and L2-balls, why can we believe that Lasso coefficients favor sparsity over Ridge coefficients? Answer each of the 5 questions with 1-2 sentences. [This question shows you how regularization works geometrically.]

The story begins... What is the next step? For fun (no credit), show us a list of the next steps you would take to design a complete algorithm to bid on advertising spaces as they appear.

Q5. Discover Gravitational Waves in Your Room (20 Points)

The most exciting scientific news recently is the discovery of the gravitational waves after its existence was predicted by Albert Einstein exactly 100 years ago. The device that detect the gravitational wave is called LIGO[4]. LIGO is the

[4]Laser Interferometer Gravitational-Wave Observatory

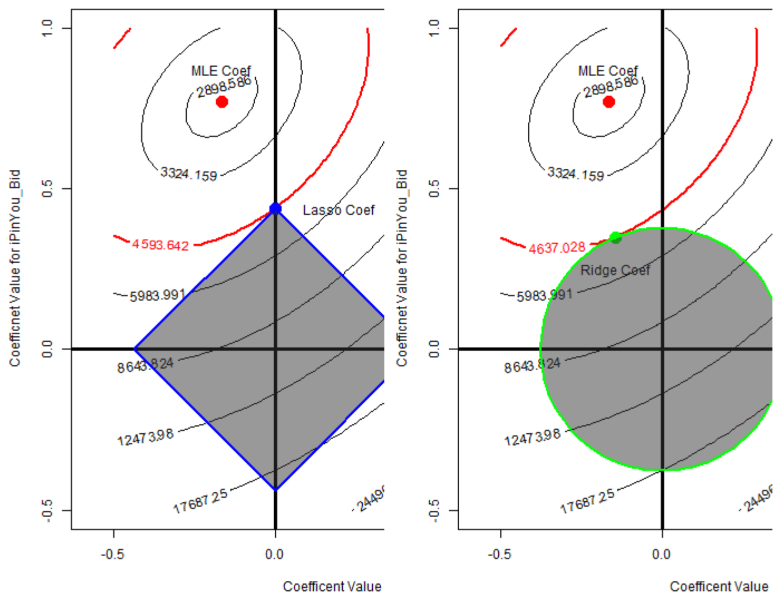
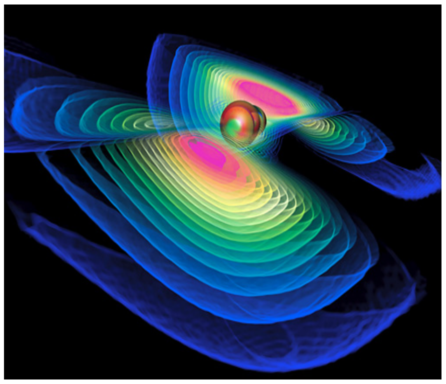
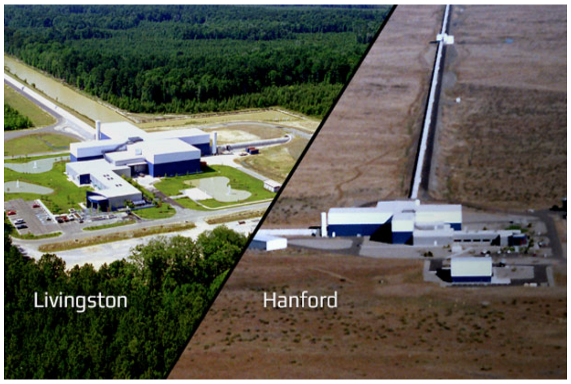


Figure 2: An (truncated) example of what your geometric pictures (one for Lasso, one for Ridge) for the MSE might look like. Your solution to this problem with have two plots, each similar to one shown here.

most expensive project ever fund by NSF at a total cost of over $620 million.

In this question, we will experience this great moment ourselves by discovering gravitational waves again in your room. Let the gravitational waves fluctuate on your own laptop!





At the time September 14 2015, 09:50:45.39 UTC, a gravitational wave event was detected by the LIGO Hanford and LIGO Livingston observatories. The historical event happened over a billion years ago and traveled many millions of galaxies to our planet and finally can be downloaded from the blackboard.

Load the gravitational wave data worth $620 million: from the LIGO Hanford `LIGO.Hanford.Data.txt`. This is a noise contaminated version of the true signal. We will reproduce the denoising procedure done by the data scientists in LIGO. Please set the seed using the command `set.seed(10)`. (We leave finding the necessary function for loading `.txt` files up to the student. Use Google and other resources to find out how to do this.)

Question 1 Denoise the time series $\mathbf{y}$ from LIGO Hanford through Lasso. (By time series, we simply mean the amplitude of the gravitational wave sampled at different time points.) Denote the length of $\mathbf{y}$ as T. The model for $\mathbf{y}$ we consider here is

$$\mathbf{y} = \mathbf{C}\mathbf{w}^* + \boldsymbol{\epsilon},$$

where $\mathbf{w}^*$ is a length-T vector of sparse coefficients, $\boldsymbol{\epsilon}$ is the noise, and $\mathbf{C} \in \mathbb{R}^{T\times T}$ is the inverse discrete cosine transform matrix[5] (which we will use as the design matrix):

$$\mathbf{C}_{jk} = \begin{cases} \sqrt{\frac{1}{T}}, & \text{for } j = 1, 1 \le k \le T; \\ \sqrt{\frac{2}{T}} \cos\left(\frac{\pi(2k-1)(j-1)}{2T}\right), & \text{for } 2 \le j \le T, 1 \le k \le T. \end{cases}$$

Solve the Lasso $\widehat{\mathbf{w}} = \operatorname{argmin}_{\mathbf{w}} \|\mathbf{y} - \mathbf{C}\mathbf{w}\|_2^2 + \lambda\|\mathbf{w}\|_1$ and use 10-fold cross-validation to choose λ. Plot the spectrum estimator $\widehat{\mathbf{w}}$ and your reconstructed signal $\widehat{\mathbf{y}} = \mathbf{C}\widehat{\mathbf{w}}$. Note that $\widehat{\mathbf{y}}$ is an estimator of the time series that tries to eliminate the effects of the noise, $\boldsymbol{\epsilon}$. Which is more sparse? ($\widehat{\mathbf{w}}$ or $\mathbf{C}^{-1}\mathbf{y}$?) Give us an explanation of why this might be?

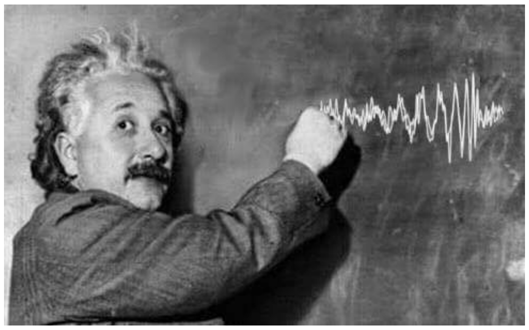


Figure 3: Your data should be similar to what Einstein draws.

Question 2 The LIGO device is super sensitive. It is also possible have some spike noise, e.g., a lightning strike in Africa or an earthquake in the Pacific Rim[6]. We also want to remove noise of this kind if it exists. Now our model is

$$\mathbf{y} = \mathbf{w}_1^* + \mathbf{C}\mathbf{w}_2^* + \boldsymbol{\epsilon},$$

where $\mathbf{w}_1^*$ is a sparse vector that models the spike noise and $\mathbf{w}_2^*$ is a sparse vector of strengths of waves in different frequencies. Let the design matrix $\boldsymbol{\Psi} = \left[\mathbf{I}_T, \mathbf{C}\right] \in \mathbb{R}^{T\times 2T}$, where $\mathbf{I}_T \in \mathbb{R}^{T\times T}$ is an identity matrix. Consider the Lasso

$$\widetilde{\mathbf{w}} = \operatorname*{argmin}_{\mathbf{w}} \|\mathbf{y} - \boldsymbol{\Psi}\mathbf{w}\|_2^2 + \lambda\|\mathbf{w}\|_1,$$

where $\widetilde{\mathbf{w}}, \mathbf{w} \in \mathbb{R}^{2T}$. Using the logic from Q3.1, explain and implement an algorithm to denoise the time series using the Lasso estimator. Use 10-fold cross-validation to choose λ just as Q3.1 and plot your reconstructed signal. Compare the results in Q3.1 and Q3.2 with the gravitational wave generated by theoretical simulation in `LIGO.Hanford.Theory.txt`. Which method is more meaningful and why? You can listen your preferred estimator through the function `play()` in the package `audio`.

Q6. Upright Human Detection in Photos (25 points)

In this section, we are going to create a human detector that tells us whether there is a upright human in a given photo. We treat this as a classification problem with two classes: having humans or not. You are provided with two datasets[7]

[5]In essence, we are decomposing the wave into component cosine waves of different frequencies. The j-th column of $\mathbf{C}$ corresponds to a cosine curve with frequency $(2k-1)\pi/(2T)$.

[6]Gravitational Waves Exist, New Yorker, 2015.

[7]Photos in `POS` have 160×96 pixels, while photos in `NEG` have larger sizes in `POS`. All photos are stored as `png` files.

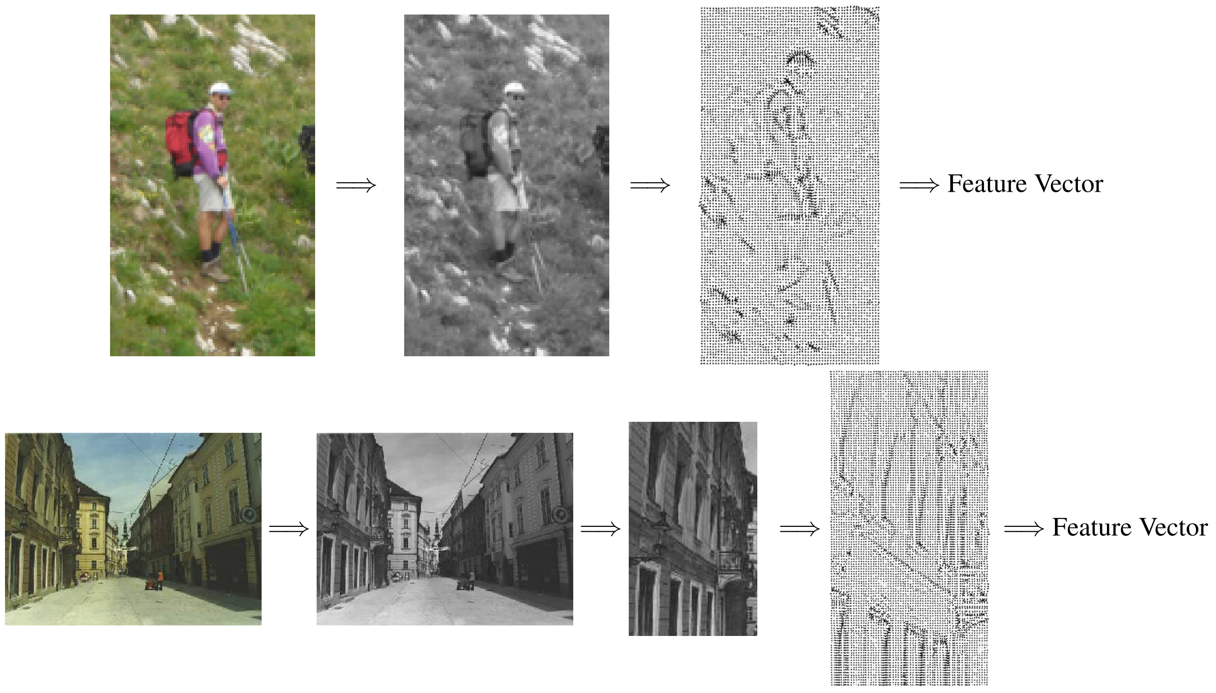


Figure 4: Illustration of feature extraction. The first row corresponds to a positive example from `POS`, while the second row corresponds to a negative one from `NEG`.

`POS` and `NEG` that have photos with and without upright humans respectively.

Part I: Preprocessing

(a) Randomly pick out one image in `NEG` and one in `POS`. For each of the two images, implement each step below to vectorize and extract useful information. We provide functions that will be used in the following steps in a R-script `functions.r`. Please CAREFULLY read the appendix for the detailed introductions of these functions.

1. Download and install the package `png`, and use the function `readPNG` to load photos[8].
2. Use the function `rgb2gray`[9] to transform original photos to the black and white version.
3. Since photos in `NEG` have bigger sizes than those in `POS`, we need to crop them to keep consistency in dimensions. So for all photos `NEG`, use the function `crop.r` to randomly crop a 160×96 picture[10] from the original one.
4. Use the function `grad` to obtain the gradient field of the center 128×64 part of the grayscale matrix.
5. Use the function `hog` (Histograms of Oriented Gradient) to extract a feature vector from the gradient field obtained in the previous step. Partition the height and width into 4 partitions each. Partition the angles into 6 intervals. (Your feature vector should then have $4 \times 4 \times 6 = 96$ components.) Please see the appendix for parameter configuration of this function.

[8]Note that the output of `readPNG` is a three-dimensional array. The first two dimensions identify the position of pixels, and the third dimension identifies the channels.

[9]The output of `rgb2gray` will be a grayscale matrix.

[10]A image with height h and width w is denoted $h \times w$

For each of the two images, provide a picture showing each step above, i.e., the original picture → the black and white picture → the cropped picture (for `NEG` only) → the gradient field → the feature vector. An example of the procedure is illustrated in Fig.4. For the feature vector, report its first six components.

(Hint: You can use `writePNG(X, target = 'filename.png')` to write some image array `X` to a png file. For exporting the plot generated by `grad`, you can use the following code:

```
setEPS()
postscript("test.eps")
g=grad(X, 128, 64, T)
dev.off()
```

The code above puts the 128×64 gradient field in the object `g`, and also saves the plot in a new image file `test.eps`.)

(b) Now, apply the above procedure to obtain feature vectors for each image in the dataset. Concatenate the feature vectors together into the rows of a dataframe. Add an additional column indicating whether each row is in `POS` or `NEG`. This will be the dataset you use for Part II.

(Hint: Use the `dir()` function to get the names of all the files in a directory. The functions `rbind()` and `cbind()` combine vectors together row-wise and column-wise, respectively.)

Part II:

We will apply Logistic regression to train the model and investigate its performance. Download the package `glmnet`. Use the function `glmnet` to train the model and plot out the regularization path. In addition, use the function `cv.glmnet` to do the cross validation with the option `type.measure="class"`. Plot the cross validation errors versus the tuning parameter λ.

Question 7. Sentiment Analysis on Amazon Product Reviews (30 points)

Sentiment analysis (also known as opinion mining) refers to the use of natural language processing, text analysis and computational linguistics to identify and extract subjective information in source materials. Sentiment analysis is widely applied to reviews and social media for a variety of applications, ranging from marketing to customer service. In this homework, we'll explore product reviews and ratings from Amazon and build a sentiment analysis classifier based on regularized logistic regression.

(a) First, load in the data `Amazon_SML.RData`. Answer the following basic questions. What are the column names? How many reviews are there? How many unique products are shown? Which product has the most '5' ratings (and how many 5 ratings are there for this product)? Which product has the most '1' ratings (and how many 1 ratings are there for this product)? We suggest to use the `dlply` function in the `plyr` R package for this.

(b) How many reviews of each rating value are there in the entire dataset? Our goal is to build a classifier that reads the review and classifies whether the review was "good" (rating = 5) or "bad" (rating = 1). What is the best performance of a "constant classifier", a classifier that ignores the review and blindly assigns a constant classification?

Now let's build the term-document matrix `dtm.mat` using the script `tdMat.R`. This will help build a matrix where each row represents one document and each column represents a different word. Element (i,j) of the matrix then represents how many times word j appeared in document i (the ith review).

(c) Now we'll run regularized logistic regression on our dataset. We're going to split the `dtm.mat` into two parts: the training set and testing set using the script `splitData.R`. Fit a regularized logistic regression model by `glmnet`. The regularization parameter should be chosen as the `lambda.1se` (1 standard error above the minimizing lambda) calculated by `cv.glmnet`. Please use the random seed and lambda sequence as specified here so that we can check your results.

```
set.seed(10)
lambda<-exp(seq(-20, -1, length.out = 99))
cvfit<-cv.glmnet(x,y,family="binomial",type.measure="class",lambda=lambda)
```

How many covariates have non-zero coefficients in the model selected by lambda.1se? List the twenty words with the most positive coefficients and twenty words with most negative coefficients.

(d) We would like to investigate some of the words that our model selected. We will focus on the words with the most positive and most negative weights that appeared in more than 10 documents. What are these two words? How many documents using either of these two words had a rating of 1 or 5? Print out the first document (according to train.tag) that used this most positive word and the first document that used this most negative word.

(e) Now run the fitted logisitic model on our testing data and report the misclassification rate. How does this compare with the "constant classifier" we originally discussed before (i.e., is it better or worse)?

Q8. Identifying Risk Factors for the Heart Disease (15 points)

The risk factors for the cardiovascular disease had not been well understood until the late 1940s. When President Franklin Delano Roosevelt died on April 12, 1945, his blood pressure was 300/190 and his physician said it was just normal for a man of his age. The Framingham Heart Study, which began in 1948, is the first long-term study for cardiovascular disease designed to identify the risk factors for the heart disease. You can find a detailed description of the dataset `framingham.csv` on the website

`https://biolincc.nhlbi.nih.gov/static/studies/teaching/framdoc.pdf`

In this problem, we want to find out the risk factors that cause coronary heart disease (TenYearCHD). CHD is a disease of the blood vessels supplying the heart. This is one type of heart disease, which has been the leading cause of death worldwide since 1921. In 2008, 7.3 million people died from CHD.

The dataset `framingham.csv` includes several demographic risk factors: the sex of the patient, the age of the patient in years, the education level coded as either 1 for some high school, 2 for a high school diploma or GED, 3 for some college or vocational school, and 4 for a college degree. The dataset also recorded behavioral risk factors associated with smoking: whether or not the patient was a smoker and the number of cigarettes that the person smoked on average in one day. Medical history risk factors were also included. These were whether or not the patient was on blood pressure medication, whether or not the patient had previously had a stroke, whether or not the patient was hypertensive, and whether or not the patient had diabetes. Lastly, the data set had risk factors from the first physical examination of the patient: the total cholesterol level, systolic blood pressure, diastolic blood pressure, Body Mass Index, or BMI, heart rate and glucose.

Part I. Statistical Inference. (5 points) Load the dataset into R and run logistic regression using the function `glm`. Use the `summary` command to find out the `p-value` for all the variables and report the statistically significant ones (the ones with `p-value` smaller than 0.05).

Part II. Prediction Analysis. (5 points) Randomly and evenly split the entire dataset into five subsets using the function `sample`. Use four subsets as the training data and the other subset as the testing data. Report the prediction error of logistic regression. Please use `set.seed(100)` before splitting the data so that your answers should be the same always.

(Hint: `sample` can be used to obtain a random permutation of the data)

Part III. Regularized or Not (5 points) Use `glmnet` to apply logistic regression with LASSO penalty on the data and plot cross validation errors against tuning parameters with `nfolds` $= 5$ and `type.measure="class"`. What is the shape of the curve? More specifically, is it a typical U-curve that first goes down and then goes up? Do you think we need to use regularization here?

(Hint: Drop the sample if it has `NA` values)

A Under-the-Hood Details

Generally speaking, the `glmnet` package minimizes the sum of negative log-likelihood and ℓ_1-penalization of β. Under different model families, the objective function has different specific forms.

Additional Details for Question 1 in Q4: Here, we want to form predictions that follow a Binomial distribution. We do not need to worry about this ourselves, but just so we're aware of what's going on, `glmnet` uses a regularized logistic regression to handle a binomial response. The statistical model utilizes a conditional probability of the following form

$$\mathbb{P}_{\beta_0,\boldsymbol{\beta}}(Y_i = y_i \mid \mathbf{X}_i = \mathbf{x}_i) = \left(\frac{1}{1+e^{-\beta_0-\mathbf{x}_i^\intercal\boldsymbol{\beta}}}\right)^{y_i}\left(1-\frac{1}{1+e^{-\beta_0-\mathbf{x}_i^\intercal\boldsymbol{\beta}}}\right)^{(1-y_i)} \tag{A.1}$$

where β_0 is our intercept, $\boldsymbol{\beta}$ is our vector of p coefficients, n is the number of data points, y_i is the Bernoulli variable indicating a click or not for data point i, $\mathbf{x}_i$ is a vector of 21 features for data point i. Notice that this looks like the form $p_i^{y_i}(1-p_i)^{(1-y_i)}$ for some p_i. The objective function `glmnet` is solving is

$$\min_{(\beta_0,\boldsymbol{\beta})\in\mathbb{R}^{p+1}} -\frac{1}{n}\sum_{i=1}^{n}\left[y_i(\beta_0+\mathbf{x}_i^\intercal\boldsymbol{\beta}) - \log(1+e^{(\beta_0+\mathbf{x}_i^\intercal\boldsymbol{\beta})})\right] + \lambda[(1-\alpha)||\boldsymbol{\beta}||_2^2 + \alpha||\boldsymbol{\beta}||_1] \tag{A.2}$$

where α is a parameter.

Additional Details for Question 2 in Q4: Here, we want to form predictions that follow a Gaussian distribution. In this setting, `glmnet` uses a regularized linear regression. The statistical model is

$$\mathbb{P}_{\beta_0,\boldsymbol{\beta},\sigma^2}(Y_i = y_i \mid \mathbf{X}_i = \mathbf{x}_i) = N(y_i - \beta_0 - \mathbf{x}_i^\intercal\boldsymbol{\beta}, \sigma^2) \tag{A.3}$$

where we're only interested in estimating β_0 and $\boldsymbol{\beta}$. The objective function `glmnet` is solving is

$$\min_{(\beta_0,\boldsymbol{\beta})\in\mathbb{R}^{p+1}} \frac{1}{2n}\sum_{i=1}^{n}(y_i-\beta_0-\mathbf{x}_i^\intercal\boldsymbol{\beta})^2+\lambda[(1-\alpha)||\boldsymbol{\beta}||_2^2+\alpha||\boldsymbol{\beta}||_1]. \tag{A.4}$$

B Useful Functions in Q6

1. `rgb2gray(X)` transforms colored pictures to their black and white versions. `X` is a three dimensional array, where the first two dimensions index the position of the pixel and the last dimension denotes the four channels (R, G, B and A). Search the key word "RGBA" if you are interested in image data representation. The output will be a grayscale matrix that corresponds to the black and white version of the original picture.

2. `crop.r(X, h, w)` randomly crops a sub-picture that has height h and width w from `X`. The output is therefore a sub-matrix of `X` with `h` rows and `w` columns.

3. `crop.c(X, h, w)` crops a sub-picture that has height h and width w at the center of `X`. The output is therefore a sub-matrix of `X` with `h` rows and `w` columns. This function helps the `hog(...)` function. For, the cropping in your assignment, use `crop.r()`.

4. `grad(X, h, w, pic)` yields the gradient field at the center part of the given grayscale matrix `X`. The center region it examines has height `h` and width `w`. It returns a list of two matrices `xgrad` and `ygrad`. The parameter `pic` is a boolean variable. If it is `TRUE`, the generated gradient filed will be plotted. Otherwise the plot will be omitted.

5. `hog(xgrad, ygrad, hn, wn, an)` returns a feature vector in the length of `hn*wn*an` from the given gradient field. `(xgrad[i,j], ygrad[i,j])` gives the grayscale gradient at the position `(i,j)`. `hn` and `wn` are the partition number on height and width respectively. `an` is the partition number on the angles (or the interval $[0, 2\pi)$ equivalently).

Histogram of Oriented Gradient

Here we give a brief introduction of what `hog(xgrad, ygrad, hn, wn, an)` does. First of all, it uniformly partitions the whole picture into `hn*wn` small parts with `hn` partitions on the height and `wn` partitions on the width. For each small part, it counts the gradient direction whose angle falls in the intervals $[0, 2\pi/\texttt{an})$, $[2\pi/\texttt{an}, 4\pi/\texttt{an})$, ..., $[2(\texttt{an}-1)\pi/\texttt{an}, 2\pi)$ respectively. So `hog` can get `an` frequencies for each small picture. Applying the same procedure to all the small parts, `hog` will have `hn*wn*an` frequencies that constitute the final feature vector for the given gradient field.