ORF 525: Statistical Foundations of Data Science

Jianqing Fan — Frederick L. Moore'18 Professor of Finance

Problem Set #3 Spring 2026

Due Friday, March 6, 2026.

1. Consider the generalized linear model $f(y|\mathbf{X}=\mathbf{x}) = \exp\{\frac{y\theta - b(\theta)}{\phi} + c(y,\phi)\}$ with the canonical link. Let $\ell_n(\boldsymbol{\beta})$ denote the negative log-likelihood of the data $\{(\mathbf{X}_i, Y_i)\}_{i=1}^n$.

   (a) The formula for estimating the variance of the MLE $\widehat{\boldsymbol{\beta}}$ is $\widehat{\text{var}}(\widehat{\boldsymbol{\beta}}) = [-\nabla^2 \ell_n(\widehat{\boldsymbol{\beta}})]^{-1}$. Show that
   $$\widehat{\text{var}}(\widehat{\boldsymbol{\beta}}) = \phi[\sum_{i=1}^n b''(\widehat{\theta}_i)\mathbf{X}_i\mathbf{X}_i^T]^{-1}, \qquad \text{where} \qquad \widehat{\theta}_i = \mathbf{X}_i^T\widehat{\boldsymbol{\beta}}.$$

   (b) Deduce $\widehat{\text{var}}(\widehat{\boldsymbol{\beta}})$ for logistic regression and Poisson regression.

2. Let $\ell_n(\boldsymbol{\beta}) = n^{-1}\sum_{i=1}^n [b(\mathbf{X}_i^T\boldsymbol{\beta}) - Y_i\mathbf{X}_i^T\boldsymbol{\beta}]$ be the (normalized) negative log-likelihood of the generalized linear model with $\phi = 1$. Consider the penalized likelihood estimator
   $$\widehat{\boldsymbol{\beta}} \in \text{argmin}_{\boldsymbol{\beta}}\{\ell_n(\boldsymbol{\beta}) + \lambda_n\|\boldsymbol{\beta}\|_1\}.$$

   (a) Show that $\nabla\ell_n(\boldsymbol{\beta}^*) = n^{-1}\sum_{i=1}^n \varepsilon_i\mathbf{X}_i$, where $\varepsilon_i = b'(\mathbf{X}_i^T\boldsymbol{\beta}^*) - Y_i$.

   (b) Let $\boldsymbol{\Delta} = \widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}^*$ where $\boldsymbol{\beta}^*$ is the true parameter. If $\lambda_n \geq 2\|\nabla\ell_n(\boldsymbol{\beta}^*)\|_\infty$, then for any set $\mathcal{S} \subseteq \{1, 2, \cdots, p\}$ we have
   $$\|\boldsymbol{\Delta}_{\mathcal{S}^c}\|_1 \leq 3\|\boldsymbol{\Delta}_{\mathcal{S}}\|_1 + 4\|\boldsymbol{\beta}^*_{\mathcal{S}^c}\|_1.$$
   Hint: Apply Proposition 5.3.

   (c) Let $\mathcal{S}_0 = \text{supp}(\boldsymbol{\beta}^*)$ and $s_n = |\mathcal{S}_0|$. Suppose that $\|\boldsymbol{\beta}^*_{\mathcal{S}_0^c}\|_1 \lesssim \lambda_n$ (here $p_n \lesssim q_n$ means there exists some constant $C > 0$ such that $p_n \leq Cq_n$ holds for sufficiently large n), and the restricted strong convexity holds with $\tau_L = 0$ and κ_L being a positive constant. Given $\lambda_n \geq 2\|\nabla\ell_n(\boldsymbol{\beta}^*)\|_\infty$, show that
   $$\|\boldsymbol{\Delta}\|_2^2 \lesssim s_n\lambda_n^2 \qquad \text{and} \qquad \|\boldsymbol{\Delta}\|_1 \lesssim s_n\lambda_n.$$
   Hint: Apply Theorem 5.8.

3. Consider the loss $\ell_n(\boldsymbol{\beta}) = n^{-1}\sum_{i=1}^n [b(\mathbf{X}_i^T\boldsymbol{\beta}) - Y_i\mathbf{X}_i^T\boldsymbol{\beta}]$ for the logitstic regression model with $b(\theta) = \log(1+\exp(\theta))$ and normalized covariates $n^{-1}\sum_{i=1}^n X_{ij}^2 = 1$ for $j = 1, \cdots, p$. Assume that Y_i are independent. Show that

   (a) $\varepsilon_i \in [-1, 1]$, where $\varepsilon_i = b'(\mathbf{X}_i^T\boldsymbol{\beta}^*) - Y_i$

   (b) $P(|\sum_{i=1}^n \varepsilon_i X_{ij}| > \sqrt{n}t) \leq 2\exp(-t^2/2)$ for all j.

   (c) $\|\nabla\ell_n(\boldsymbol{\beta}^*)\|_\infty = O_p(\sqrt{\frac{\log p}{n}})$, where $\nabla\ell_n(\boldsymbol{\beta}^*) = n^{-1}\sum_{i=1}^n \varepsilon_i\mathbf{X}_i$, where $\varepsilon_i = b'(\mathbf{X}_i^T\boldsymbol{\beta}^*) - Y_i$.

4. Consider the framework for marginal screening. Letting $\widehat{L}_0 = \min_{\beta_0} n^{-1}\sum_{i=1}^n L(Y_i, \beta_0)$, define
   $$\widehat{L}_j = \widehat{L}_0 - \min_{\beta_0,\beta_j} n^{-1}\sum_{i=1}^n L(Y_i, \beta_0 + X_{ij}\beta_j).$$
   and $(\widehat{\beta}_{0j}^M, \widehat{\beta}_j^M) = \text{argmin}_{\beta_0,\beta_j} n^{-1}\sum_{i=1}^n L(Y_i, \beta_0 + X_{ij}\beta_j)$, the marginal regression coefficient.

(a) For the square loss $L(y,x)=(y-x)^2$, show that $\widehat{L}_j=\widehat{r}_j^2\widehat{L}_0$ and $\widehat{\beta}_j=\widehat{r}_j\widehat{L}_0^{1/2}$, assuming the j^{th} variable has been standardized to have sample mean 0 and variance 1.

(b) Give an example where marginal correlation screening will miss important variables (false negatives) and select many unimportant variables (false positives).

(c) Let $\mathbf{X}$ be jointly normal with $X_i\sim N(0,1)$, $\text{cov}(X_i,X_j)=0.7, \forall i\neq j<p$ and $\text{cov}(X_i,X_p)=0$ for $i<p$. Consider the linear combination $\mathbf{X}^T\boldsymbol{\beta}$ with $\beta_1=1$, $\beta_2=2$, $\beta_3=3$, $\beta_j=0$ for $5\leq j<p$, and $\beta_p=0.2$, what value of β_4 makes $\text{cov}(X_4,\mathbf{X}^T\boldsymbol{\beta})=0$? With this choice of β_4, compare $\text{cov}(\mathbf{X}^T\boldsymbol{\beta},X_j)$ for $5\leq j<p$ with $\text{cov}(\mathbf{X}^T\boldsymbol{\beta},X_p)$.

(d) Now suppose the response $Y=1$ with probability $p(X)=\frac{\exp(\mathbf{X}^T\boldsymbol{\beta})}{1+\exp(\mathbf{X}^T\boldsymbol{\beta})}$. Can the marginal screening pick up X_4 if $\text{cov}(X_4,\mathbf{X}^T\boldsymbol{\beta})=0$? Why?

5. Upright Human Detection in Photos

Go to the instructor's class website, download the image data `pictures.zip` and its associated preliminary codes `human.r`. In this problem, 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 in a photo. You are provided with two datasets `POS` and `NEG` that have photos with and without upright humans respectively.

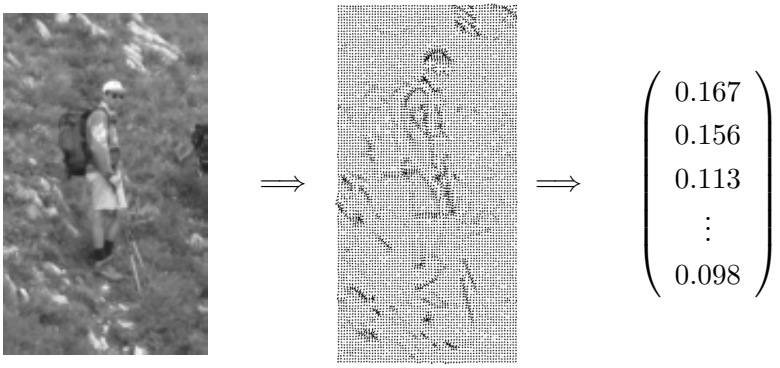
(a) Load Pictures and Extract Features (The code has already written for you)

The tutorial below that explains the data loading and feature extraction in `human.r`. If you do not want to read, you can just execute the code to get the extracted features. Remember to install the package `png` by using `install.packages("png")` and to change the working directory to yours.

i. Install the package `png` by using `install.packages("png")`, and use the function `readPNG` to load photos. The function `readPNG()` will return the grayscale matrix of the picture.

ii. Use the function `grad` to obtain the gradient field of the central 128×64 part of the grayscale matrix.

iii. Use the function `hog` (Histograms of Oriented Gradient) to extract a feature vector from the gradient field obtained in the previous step. Your feature vector should have 96 components. Please see the appendix for parameter configuration of this function.



Original Picture Gradient Field Feature Vector

Figure 1: Illustration of feature extraction for a positive example from `POS`.

(b) Logistic Regression (You are responsible for writing the code).

In this question, we will apply logistic regression to the data and use the cross validation to test the classification accuracy of the fitted model. Please finish the following steps:

i. Set the random seed to be 525, i.e., type in `set.seed(525)` in your code. Randomly divide your whole data into five parts. Let the first four parts be the training data and the rest be your testing data.

ii. Feed the training data to the function `glm` and store the fitted model as `fitted1` in R.

iii. Use the function `predict` to apply your fitted model `fitted1` to do classification on the testing data and report the misclassification rate (prediction error).

iv. Let us now select the features using the stepwise selection, by issuing the R-function `step(fitted1)`. Let us call the selected model (the last model in the output) `fitted2`. Report the model `fitted2` and the misclassification rate of `fitted2`.

v. Report the misclassification rate by adding a Lasso penalty with λ chosen by 10 fold cross-validation. (You can use the function `glmnet`.)

Appendix

A. 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.

B. Useful Functions

(a) `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.

(b) `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()`.

(c) `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.

(d) `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).