CAP 5625: Programming Assignment 4

Due on Canvas by Friday, December 1, 2023 at 11:59pm

Preliminary instructions

You may consult with other students currently taking CAP 5625 in your section at FAU on this programming assignment. If you do consult with others, then you must indicate this by providing their names with your submitted assignment. However, all analyses must be performed independently, all source code must be written independently, and all students must turn in their own independent assignment. Note that for this assignment, you may *choose* to pair up with one other student in *your section* of CAP 5625 and submit a joint assignment. If you *choose* to do this, then both your names must be associated with the assignment and you will each receive the same grade.

> Though it should be unnecessary to state in a graduate class, I am reminding you that you may **not** turn in code (partial or complete) that is written or inspired by others, including code from other students, websites, past code that I release from prior assignments in this class or from past semesters in other classes I teach, or any other source that would constitute an academic integrity violation. All instances of academic integrity violations will receive a zero on the assignment and will be referred to the Department Chair and College Dean for further administrative action. A second offense could lead to dismissal from the University and any offense could result in ineligibility for Departmental Teaching Assistant and Research Assistant positions.

You may choose to use whatever programming language you want. However, you must provide clear instructions on how to compile and/or run your source code. I recommend using a modern language, such as Python, R, or Matlab as learning these languages can help you if you were to enter the machine learning or artificial intelligence field in the future.

All analyses performed and algorithms run must be written from scratch. That is, you may not use a library that can perform gradient descent, cross validation, ridge regression, logistic (multinomial) regression, optimization, etc. to successfully complete this assignment (though you may reuse your relevant code from Programming Assignments 1, 2, and 3). The goal of this assignment is not to learn how to use particular libraries of a language, but it is to instead understand how key methods in statistical machine learning are implemented. With that stated, I will provide 5% extra credit if you additionally implement the assignment using built-in statistical or machine learning libraries (see Deliverable 7 at end of the document).

Note, credit for deliverables that request graphs, discussion of results, or specific values will not be given if the instructor must run your code to obtain these graphs, results, or specific values.

Brief overview of assignment

In this assignment you will still be analyzing human genetic data from $N = 183$ training observations (individuals) sampled across the world. The goal is to fit a model that can predict (classify) an individual's ancestry from their genetic data that has been projected along $p = 10$ top principal components (proportion of variance explained is 0.2416) that we use as features rather than the raw genetic data, as the training would take too long to complete for this assignment with the raw data. I recognize that we have not covered precisely what principal components are, but we will get to this in Module 10, and for now you should just treat them as highly informative features as input to your classifier. Specifically, you will perform a penalized (regularized) logistic (multinomial) regression fit using ridge regression, with the model parameters obtained by batch gradient descent. Your predictions will be based on $K = 5$ continental ancestries (African, European, East Asian, Oceanian, or Native American). Ridge regression will permit you to provide parameter shrinkage (tuning parameter $\lambda \geq 0$) to mitigate overfitting. The tuning parameter λ will be chosen using five-fold cross validation, and the best-fit model parameters will be inferred on the training dataset conditional on an optimal tuning parameter. This trained model will be used to make predictions on new test data points.

Training data

Training data for these observations are given in the attached `TrainingData_N183_p10.csv` comma-separated file, with individuals labeled on each row (rows 2 through 184), and input features (PC1, PC2, …, PC10) and ancestry label given on the columns (with the first row representing a header for each column).

Test data

Test data are given in the attached `TestData_N111_p10.csv` comma-separated file, with individuals labeled on each row (rows 2 through 112), and input features (PC1, PC2, …, PC10), and ancestry label given on the columns (with the first row representing a header for each column). There are five individuals with Unknown ancestry, 54 individuals with Mexican ancestry, and 52 individuals with African American ancestry. Each of the five Unknown individuals belong to one of the five ancestries represented in the training set, and each ancestry is represented once in the five Unknown individuals. The Mexican and African American individuals have a range of ancestry proportions based on historical mixing of ancestors of diverse ancestry. You will use the class probabilities from logistic (multinomial) regression to predict the ancestry proportion of each of these putatively mixed samples.

Detailed description of the task

Recall that the task of performing a ridge-penalized logistic regression fit to training data $\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$ is to minimize the cost function

$$J(\mathbf{B}, \lambda) = -\log \mathcal{L}(\mathbf{B}) + \lambda \sum_{k=1}^{K} \sum_{j=1}^{p} \beta_{jk}^2$$

where

$$\mathbf{B} = \begin{bmatrix} \beta_{01} & \beta_{02} & \cdots & \beta_{0K} \\ \beta_{11} & \beta_{12} & \cdots & \beta_{1K} \\ \vdots & \vdots & \ddots & \vdots \\ \beta_{p1} & \beta_{p2} & \cdots & \beta_{pK} \end{bmatrix}$$

is the $(p+1) \times K$ matrix of model parameters, where

$$\log \mathcal{L}(\mathbf{B}) = \sum_{i=1}^{N} \left[\sum_{k=1}^{K} y_{ik} \left(\beta_{0k} + \sum_{j=1}^{p} x_{ij} \beta_{jk} \right) - \log \left(\sum_{\ell=1}^{K} \exp \left(\beta_{0\ell} + \sum_{j=1}^{p} x_{ij} \beta_{j\ell} \right) \right) \right]$$

is the likelihood function, where y_{ik}, $i = 1,2, \ldots, N$ and $k = 1,2, \ldots, K$, is an indicator variable defined as

$$y_{ik} = \begin{cases} 1 & \text{if } y_i = k \\ 0 & \text{if } y_i \neq k \end{cases}$$

and where the input p features are standardized (*i.e.*, centered and divided by their standard deviation). Moreover, recall that batch gradient descent updates each parameter β_{jk}, $j = 0,1, \ldots, p$ and $k = 1,2, \ldots, K$, as follows:

$$\beta_{jk} \coloneqq \beta_{jk} - \alpha \frac{\partial}{\partial \beta_{jk}} J(\mathbf{B}, \lambda)$$

where α is the learning rate and where the partial derivative of the cost function with respect to parameter β_{jk} is

$$\frac{\partial}{\partial \beta_{jk}} J(\mathbf{B}, \lambda) = \begin{cases} -\sum_{i=1}^{N} [y_{ik} - p_k(x_i; \mathbf{B})] & \text{if } j = 0 \\ -\sum_{i=1}^{N} x_{ij} [y_{ik} - p_k(x_i; \mathbf{B})] + 2\lambda \beta_{jk} & \text{if } j > 0 \end{cases}$$

where

$$p_k(x_i; \mathbf{B}) = \frac{\exp\left(\beta_{0k} + \sum_{j=1}^{p} x_{ij} \beta_{jk}\right)}{\sum_{\ell=1}^{K} \exp\left(\beta_{0\ell} + \sum_{j=1}^{p} x_{ij} \beta_{j\ell}\right)}$$

To implement this algorithm, depending on whether your chosen language can quickly compute vectorized operations, you may implement batch gradient descent using either Algorithm 1 or Algorithm 2 below (choose whichever you are more comfortable implementing). Note that in languages like R, Python, or Matlab, Algorithm 2 (which would be implemented by several nested loops) may be much slower than Algorithm 1. Also note that if you are implementing Algorithm 1 using Python, use numpy arrays instead of Pandas data frames for computational speed. For this assignment, assume that we will reach the minimum of the cost function within a fixed number of steps, with the number of iterations being 10,000.

You may need to explore different learning rate values to identify one that is not too large and not too small, such that it is likely for the algorithm to converge in a reasonable period of time. I would consider a learning rate of $\alpha = 10^{-5}$, though I encourage you to explore how your model trains for smaller and larger learning rates as well. For this assignment, assume that we will reach the minimum of the cost function within a fixed number of steps, with the number of iterations being 10,000 (a large number as we have many parameters). Keep in mind that due to this large number of iterations, it could take a long time to train your classifier.

Algorithm 1 (vectorized):

Step 1. Choose learning rate α and fix tuning parameter λ

Step 2. Generate $N \times (p+1)$ augmented design matrix

$$\mathbf{X} = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1p} \\ 1 & x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N1} & x_{N2} & \cdots & x_{Np} \end{bmatrix}$$

where column $k+1$ has been centered and standardized such that feature k, $k = 1,2,\ldots,p$, has mean zero and standard deviation one, and generate $N \times K$ indicator response matrix

$$\mathbf{Y} = \begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1K} \\ y_{21} & y_{22} & \cdots & y_{2K} \\ \vdots & \vdots & \ddots & \vdots \\ y_{N1} & y_{N2} & \cdots & y_{NK} \end{bmatrix}$$

where $y_{ik} = 1$ if observation i is from class k, and 0 otherwise.

Step 3. Initialize the $(p+1) \times K$-dimensional parameter matrix

$$\mathbf{B} = \begin{bmatrix} \beta_{01} & \beta_{02} & \cdots & \beta_{0K} \\ \beta_{11} & \beta_{12} & \cdots & \beta_{1K} \\ \vdots & \vdots & \ddots & \vdots \\ \beta_{p1} & \beta_{p2} & \cdots & \beta_{pK} \end{bmatrix}$$

to all zeros, so that initially each class has the same probability.

Step 4. Compute $N \times K$ unnormalized class probability matrix as

$$\mathbf{U} = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1K} \\ u_{21} & u_{22} & \cdots & u_{2K} \\ \vdots & \vdots & \ddots & \vdots \\ u_{N1} & u_{N2} & \cdots & u_{NK} \end{bmatrix} = \exp(\mathbf{XB})$$

where $\exp(\mathbf{XB})$ indicates exponentiation of each element of the $\mathbf{XB}$ matrix, and not the matrix exponential of $\mathbf{XB}$.

Step 5. Compute $N \times K$ normalized class probability matrix as

$$\mathbf{P} = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1K} \\ p_{21} & p_{22} & \cdots & p_{2K} \\ \vdots & \vdots & \ddots & \vdots \\ p_{N1} & p_{N2} & \cdots & p_{NK} \end{bmatrix}$$

where

$$p_k(x_i; \mathbf{B}) = p_{ik} = \frac{u_{ik}}{\sum_{\ell=1}^{K} u_{i\ell}}$$

Step 6. For ease of vectorization, generate $(p+1) \times K$ intercept matrix

$$\mathbf{Z} = \begin{bmatrix} \beta_{01} & \beta_{02} & \cdots & \beta_{0K} \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

Step 7. Update the parameter matrix as

$$\mathbf{B} := \mathbf{B} + \alpha[\mathbf{X}^T(\mathbf{Y} - \mathbf{P}) - 2\lambda(\mathbf{B} - \mathbf{Z})]$$

Step 8. Repeat Steps 4 to 7 for 10,000 iterations

Step 9. Set the last updated parameter matrix as $\widehat{\mathbf{B}}$

Algorithm 2 (non-vectorized):

Step 1. Choose learning rate α and fix tuning parameter λ

Step 2. Generate $N \times p$ design matrix

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{Np} \end{bmatrix}$$

where column k has been centered and standardized such that feature k, $k = 1,2, \ldots, p$, has mean zero and standard deviation one, and generate $N \times K$ indicator response matrix

$$\mathbf{Y} = \begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1K} \\ y_{21} & y_{22} & \cdots & y_{2K} \\ \vdots & \vdots & \ddots & \vdots \\ y_{N1} & y_{N2} & \cdots & y_{NK} \end{bmatrix}$$

where $y_{ik} = 1$ if observation i is from class k, and 0 otherwise.

Step 3. Initialize the $(p+1) \times K$-dimensional parameter matrix

$$\mathbf{B} = \begin{bmatrix} \beta_{01} & \beta_{02} & \cdots & \beta_{0K} \\ \beta_{11} & \beta_{12} & \cdots & \beta_{1K} \\ \vdots & \vdots & \ddots & \vdots \\ \beta_{p1} & \beta_{p2} & \cdots & \beta_{pK} \end{bmatrix}$$

to all zeros, so that initially each class has the same probability.

Step 4. Create temporary $(p+1) \times K$-dimensional parameter matrix

$$\mathbf{B}_{\text{temp}} = \begin{bmatrix} \beta_{01}^{\text{temp}} & \beta_{02}^{\text{temp}} & \cdots & \beta_{0K}^{\text{temp}} \\ \beta_{11}^{\text{temp}} & \beta_{12}^{\text{temp}} & \cdots & \beta_{1K}^{\text{temp}} \\ \vdots & \vdots & \ddots & \vdots \\ \beta_{p1}^{\text{temp}} & \beta_{p2}^{\text{temp}} & \cdots & \beta_{pK}^{\text{temp}} \end{bmatrix}$$

Step 5. Compute $N \times K$ unnormalized class probability matrix as

$$\mathbf{U} = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1K} \\ u_{21} & u_{22} & \cdots & u_{2K} \\ \vdots & \vdots & \ddots & \vdots \\ u_{N1} & u_{N2} & \cdots & u_{NK} \end{bmatrix}$$

where

$$u_{ik} = \exp\left(\beta_{0k} + \sum_{j=1}^{p} x_{ij}\beta_{jk}\right)$$

Step 6. Compute $N \times K$ normalized class probability matrix as

$$\mathbf{P} = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1K} \\ p_{21} & p_{22} & \cdots & p_{2K} \\ \vdots & \vdots & \ddots & \vdots \\ p_{N1} & p_{N2} & \cdots & p_{NK} \end{bmatrix}$$

where

$$p_k(x_i; \mathbf{B}) = p_{ik} = \frac{u_{ik}}{\sum_{\ell=1}^{K} u_{i\ell}}$$

Step 7. For each j, $j = 0,1, \ldots, p$, and k, $k = 1,2, \ldots, K$, find next value for parameter j for class k as

$$\beta_{jk}^{\text{temp}} := \begin{cases} \beta_{jk} + \alpha\left(\sum_{i=1}^{N}[y_{ik} - p_{ik}]\right) & \text{if } j = 0 \\ \beta_{jk} + \alpha\left(\sum_{i=1}^{N} x_{ij}[y_{ik} - p_{ik}] - 2\lambda\beta_{jk}\right) & \text{if } j > 0 \end{cases}$$

Step 8. Update the parameter matrix as $\mathbf{B} = \mathbf{B}_{\text{temp}}$

Step 9. Repeat Steps 5 to 8 for 10,000 iterations

Step 10. Set the last updated parameter matrix as $\widehat{\mathbf{B}}$

Effect of tuning parameter on inferred regression coefficients

You will consider a discrete grid of nine tuning parameter values $\lambda \in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3}, 10^{4}\}$ where the tuning parameter is evaluated across a wide range of values on a log scale. For each tuning parameter value, you will use batch gradient descent to infer the best-fit model.

Deliverable 1: Illustrate the effect of the tuning parameter on the inferred ridge regression coefficients by generating five plots (one for each of the $K = 5$ ancestry classes) of 10 lines (one for each of the $p = 10$ features), with the y-axis as $\hat{\beta}_{jk}$, $j = 1,2, \ldots, 10$ for the graph of class k, and x-axis the corresponding log-scaled tuning parameter value $\log_{10}(\lambda)$ that generated the particular $\hat{\beta}_{jk}$. Label both axes in all five plots. Without the log scaling of the tuning parameter, the plot will look distorted.

Choosing the best tuning parameter

You will consider a discrete grid of nine tuning parameter values $\lambda \in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3}, 10^{4}\}$ where the tuning parameter is evaluated across a wide range of values on a log scale. For each tuning parameter value, perform five-fold cross validation and choose the λ value that gives the smallest

$$\mathrm{CV}_{(5)} = \frac{1}{5}\sum_{m=1}^{5} \mathrm{CategoricalCrossEntropy}_m$$

where

$$\mathrm{CategoricalCrossEntropy}_m = -\frac{1}{N_m}\sum_{i \in \text{ Validation Set } m}\left(\sum_{k=1}^{K} y_{ik} \log_{10} p_k(x_i; \widehat{\mathbf{B}})\right)$$

is a measure of the cost for a classifier with K classes in fold m and where N_m is the number of observations in Validation set m.

Note that during the five-fold cross validation, you will hold out one of the five sets (here either 36 or 37 observations) as the Validation Set and the remaining four sets (the other 147 or 146 observations) will be used as the Training Set. On this Training Set, you will need to standardize (center and divided by the standard deviation across samples) each feature. These identical values used for standardizing the input will need to be applied to the corresponding Validation Set, so that the Validation set is on the same scale. Because the Training Set changes based on which set is held out for validation, each of the five pairs of Training and Validation Sets will have different standardization parameters.

Deliverable 2: Illustrate the effect of the tuning parameter on the cross validation error by generating a plot with the y-axis as $\mathrm{CV}_{(5)}$ error, and the x-axis the corresponding log-scaled tuning parameter value $\log_{10}(\lambda)$ that generated the particular $\mathrm{CV}_{(5)}$ error. Label both axes in the plot. Without the log scaling of the tuning parameter λ, the plots will look distorted.

Deliverable 3: Indicate the value of λ value that generated the smallest $\mathrm{CV}_{(5)}$ error.

Deliverable 4: Given the optimal λ, retrain your model on the entire dataset of $N = 183$ observations to obtain an estimate of the $(p + 1) \times K$ model parameter matrix as $\widehat{\mathbf{B}}$ and

make predictions of the probability for each of the $K = 5$ classes for the 111 test individuals located in `TestData_N111_p10.csv`. That is, for class k, compute

$$p_k(X; \widehat{\mathbf{B}}) = \frac{\exp(\hat{\beta}_{0k} + \sum_{j=1}^{p} X_j \hat{\beta}_{jk})}{\sum_{\ell=1}^{K} \exp\left(\hat{\beta}_{0\ell} + \sum_{j=1}^{p} X_j \hat{\beta}_{j\ell}\right)}$$

for each of the 111 test samples X, and also predict the most probable ancestry label as

$$\hat{Y}(X) = \underset{k \in \{1,2,\ldots,K\}}{\arg\max}\ p_k(X; \widehat{\mathbf{B}})$$

Report all six values (probability for each of the $K = 5$ classes and the most probable ancestry label) for all 111 test individuals.

Deliverable 5: How do the class label probabilities differ for the Mexican and African American samples when compared to the class label probabilities for the unknown samples? Are these class probabilities telling us something about recent history? Explain why these class probabilities are reasonable with respect to knowledge of recent history.

Deliverable 6: Provide all your source code that you wrote from scratch to perform all analyses (aside from plotting scripts, which you do not need to turn in) in this assignment, along with instructions on how to compile and run your code.

Deliverable 7 (extra credit): Implement the assignment using statistical or machine learning libraries in a language of your choice. Compare the results with those obtained above, and provide a discussion as to why you believe your results are different if you found them to be different. This is worth up to 5% additional credit, which would allow you to get up to 105% out of 100 for this assignment.