Final Project Report, Subspace Classifier

Tao(Richie) Lin

March 14, 2021

1 Abstract

This experiment is the final project of the Machine learning class at CUNY GC Fall 2020. It applied the subspace classifier on the NYC parking Violation data to predict if the violation ticket can be fully disputed or partially reduced. The real-world dataset is cleaned and transformed into the shape fits the model. The model was built by 1) calculating the class mean and covariance matrix, 2) finding class subspace and projection, 3) calculating the Mahalanobis Distance and 4) making the class assignment. The observation in the training set will go through the process to build a sequence of Mahalanobis Distance for every class, then the same distance will be calculated with every observation in the testing set to the mean of each class. The class assignment will be made by comparing with the training sequence. The result of this experiment is less than ideal. The prediction accuracy of the algorithm is about 20% for the dataset with 4 class, and about 30%'s assignment reserved.

2 Introduction

2.1 Dataset

For this experiment, I am using the Open Parking and Camera Violation dataset on the NYC open portal. It contains the traffic violation records in the open street parking and camera captured offense, such as speeding and failed to stop at the red light. The NYPD traffic department usually issues the summons tickets and usually ends up with a certain amount of fine to the violator if you choose to plea. As the law requires, the summons ticket recipient has the right to apply the disputation by providing more details like evidence or reasoning. Once the disputation was received, the community court judge will review the case and make the ruling whether this case should be dismissed, reduced, or remained the original penalty. This dataset was initially published in May 2016 and is updated to the present. It contains 19 variables and about 62 million records up to date. The variable contains several types of information: the violating vehicle details, such as the plate issuance state, plate type, and plate number; the violation details, such as the summons number, the summons issuance date, issuance agency, precinct and county, the violation time and details; Also the penalty details like fine amount, penalty amount, interest amount, payment amount. For this experiment, we are looking at the violation status, which reveals the summons ticket's disputation process. In this experiment, we focused on four major classes of the violation status variable: Blank (No disputation submitted), hearing held not guilty, hearing hold guilty reduction, and hearing held guilty, trying to predict the outcome of the disputation in the hearing process.

2.2 Algorithm

Building the Mahalanobis Subspace classifier requires following steps:

- calculate class mean and covariance matrix
- Finding class subspace and projection
- Calculating the Mahalanobis distance
- Making the assignment

2.2.1 Calculate class mean and covariance matrix

The training set should be organized by the class first. Let the dimension of the class k training set to be N_k, Z_k to be the count of the observation in the set, the class mean is defined as:

$$\mu_k = \frac{1}{Z_k} \sum_{z=1}^{Z_k} y_{kz}$$

The covariance matrix of the class k is defined as:

$$\Sigma_k = \frac{1}{Z_k - 1} \sum_{z=1}^{Z_k} (y_{kz} - \mu)^{N_k \times 1} (y_{kz} - \mu)'^{1 \times N_k}.$$

So, for each class c, the covariance matrix Σ_c has a eigenvalue eigenvector decomposition:

$$\Sigma_c = T_c \Lambda T'.$$

T_c being the eigenvector and Λ being the eigenvalue. To create a relative subspace projection operator S_c, We need to define the appropriate dimension E_c by selecting a parameter f, having: E_c being the minimum value that satisfy:

$$f < \frac{\sum_{n=1}^{E_c}}{\sum_{n=1}^{N}}.$$

2.2.2 Finding class subspace and projection

Let the S_c being the first E_c column of T_c, the eigenvector of Σ_c. The Projection $y^{E_c \times 1} = S_c' x$.

So we have the covariance matrix of the project y $\Sigma_y = Diagonal(\lambda_1, \lambda_2, \ldots, \lambda_{E_c})$, its inverse should be:

$$\Sigma_y^{-1} = Diagonal(\lambda_1^{-1}, \lambda_2^{-1}, \ldots, \lambda_{E_c}^{-1}).$$

2.2.3 Calculate the Mahalanobis distance

With Σ_y^{-1}, we can calculate the Mahalanobis distance as

$$d_c^2(y) = y' \Sigma_y^{-1} y.$$

This calculate will be applied on the training sequence $< y_{c1}, y_{c2}, \ldots, y_{cZ_c} >$ of class c to measure the Mahalanobis distance of each observation projection to the class mean $< d_c^2(y_{c1}), d_c^2(y_{c2}), \ldots, d_c^2(y_{cZ_c}) >$. After being sorted, the sequence of the Mahalanobis distance $< d_1^2, d_2^2, \ldots, d_{Z_c}^2 >$ will be used in the assignments of the observations in the testing set.

2.2.4 Make the assignment

Now let the calculated Mahalanobis distance of a projected observation from the testing set be $d_c^2(y)$, then

- if $d_c^2(y) < d_(1)^2$, set the p-value $p_c(y) = \frac{1}{Z_c}$
- if $d_c^2(y) > d_{(Z_c)}^2$, set the p-value $p_c(y) = 1$
- if $d_{(Z-1)}^2 < d_c^2(y) < d_Z^2$, set the p-value $p_c(y) = \frac{z - .5}{Z_c}$

Here we need to set a threshold $p_0 > 0.5$. if $p_c(y) > p_0$, then y can't be assigned to class c. The minimal $p_c(y)$ of class c for y should be assigned.

3 Experiment Process

3.1 Data preparation

3.1.1 Initialization and exploration

The dataset is downloaded as a csv file from the NYC open data portal website. There are about 62 million rows in the dataset. Considering the RAM of my machine, I decided to index data into chunks of 20k rows and randomly pick from them for multiple rounds of experiments. There are a total of 19 variables in the dataset. The Summons number as the summons ticket's unique identifier does not provide any reasonable information in the model, so we decided to remove it. Along with the violation image as we are using only the categorical variables and not considering images. To further reduce the computational burden, I removed all the records that do not contain the four violation status class: Blank as no disputation submitted, hearing held and guilty, hearing held and guilty reduction, and hearing held and not guilty. Also, to make the time variable usable, the observations missing violation time are removed. After these steps, about 2

3.1.2 Data cleaning

The dataset needs to be further cleaned and transformed to be in shape for modeling. Here are a couple of things I did:

- The registration state of the violating vehicle's plate is recoded into "NY" and "NON-NY."
- The month and year of the summons issuance date were separated out, hoping to introduce some seasonality factors.
- The time of the violation was transformed from a timestamp to the categorical variable, containing values like: "Late night" from midnight to 6 am, "Morning" from 6 am to 12 pm, "Afternoon" from 12 pm to 7 pm, and "Night" from 7 pm to the midnight.
- For the monetary variables with a clear pattern in the amount, like the fine amount, penalty amount, which usually aligns with the type of the violation, it will be recorded to a series of ascending numerical values.
- The monetary variables that do not have clear patterns in the amounts will be categorized into eight intervals from the minimum to the maximum amount, such as the interest amount and the payment amount.
- The violation detail variable was grouped and categorized into several major types of violations: driving in the bus lane, expired parking meter, failure to display parking receipt, photo captured for speeding in school zones, etc.

3.2 Modeling preparation

After the cleaning, all of the variables have been converted to categorical. The labeled value of the variables will be replaced with a series of ascending numbers to fit into the model. There could be more labeling numbers for some variables with too many classes than those with fewer classes. Because the number will be calculated in the model, having more labeling numbers could skew the distribution as larger numbers having a higher impact. To eliminate the imbalance in the weights in the model, all of the variables were normalized. So every variable has a mean of 0 and a standard deviation of 1. Then the data were separated into two halves, the training set and the testing set as directed in the project slides. Then the training set was further divided into four parts based on the classes. In such a way, the algorithm that is built as the function can apply directly to these subsets. After all of these cleaning, the training set has about 17k observation and 11 variables:

- Plate's state of registration

- License type, passenger, or commercial
- summons's issuance month and year
- Violation period, late night, morning, afternoon, or night.
- Interest amount categorized into intervals
- Reduction amount categorized into
- Intervals
- summons's an issuance police precinct
- summons's issuance county
- Violation details in category

3.3 Building algorithm

The algorithm was built by defining functions and reference the result of one another:

- Get_z: this function is made to retrieve the number of observations in the given class set.
- Get_mu: this function is made to calculate each feature's mean by summing the total value of each dimension of all observations, then divided by the total number of observations by calling the get_z function. $\mu_k = \frac{1}{Z_k} \sum_{z=1}^{Z_k} y_{kz}$
- Get_sigma: this function is made to get the covariance of the training set of the given class, referencing function get_mu for the mean of each dimension, and the function get_z for the total number of observation. $\Sigma_k = \frac{1}{Z_k - 1} \sum_{z=1}^{Z_k} (y_{kz} - \mu)^{N_k \times 1} (y_{kz} - \mu)'^{1 \times N_k}$.
- Get_lambda: this function is used to perform the eigenvector - eigenvalue decomposition to calculate the covariance matrix's eigenvalue lambda. It is referencing the function get_sigma for the covariance matrix. $\Sigma_c = T_c \Lambda T'$.
- Get_EC: this function is used to get the Ec value, which represents the smallest number that satisfies (Ec n1 cn °N n1 cn ¥ f). it is referencing function Get_lambda for the sequence of the eigenvalues.
- Get_Sc: this function is used to get the first Ec column of the eigenvector Tc as the orthogonal projection operator to the subspace of Ec. It is referencing the function get_Ec and get_sigma for the covariance matrix. $f < \frac{\sum_{n=1}^{E_c}}{\sum_{n=1}^{N}}$.
- Get_y: this function is used to create the orthogonal projection y into the subspace Sc. It references the get_Sc function for the projection operator Sc. $y^{E_c \times 1} = S_c' x$
- Get_dcsq: this function is used to calculate the Mahalanobis distance: dc squared. It references function get_mu to calculate the center of the class, the function get_sigma for the covariance matrix, and get_Sc for the subspace orthogonal project operator. $d_c^2(y) = y' \Sigma_y^{-1} y$.
- Get_tranining_seq: this function is made to build the sequence of the Mahalanobis distance between each x-tuple in training set to the center of the class. It references the previously built functions to get the parameters for the calculation: get_mu for the class center, get_sc for the orthogonal projection operator, and get_sigma for the covariance matrix for the class. Then the function loop through every x-tuple in the training set, using the function get_dcsq to calculate the Mahalanobis distance and record it in a one-dimensional array.
- Get_tested: This function takes the x-tuples in the testing set and the class training sequence to calculate the p-value of the x-tuple belongs to the class. It references the get_training sequence function for the training sequence of the given class.
- Get_assignment: This function is used to determine the assignment class for the

given x-tuple in the testing set. It references the get_tested function for the p-values of all available classes and generally assigns to the class having the minimum p-value. It also has a threshold of rejection P0; if the calculated p-value PC(y) ¿ P0, they are not allowed in class c. if there is no available class for the x tuple, it will be assigned to the reserved class.

With the functions built in such referencing structure, the experiment will have the training set of each class, f value, and p0 value as the input and the class assignment as the final output. In the last step of the process, we built a for loop to test every observation in the test set and record the assignment. The assigned class will be paired with the true class in a data frame and used to calculate the performance values: assignment accuracy and the reserved class ratio. This assignment process will also be replicated with a progressively increasing value of p0 starting from 0.5 to 1 with a step as 0.05. The accuracy and the reserved ratio are also stored and compared for the conclusive analysis.

3.4 Result

The accuracy of this experiment is unexpectedly low. The f was set equals 0.4 for the first run to reduce the processing time. As the p0 value increased, the accuracy increased from 16% to 27%, along with the reserved class assignment ratio from 19% to 35%.

```
> p0_rate
     p0      accur  reserved
1  0.50 0.1616953 0.1946332
2  0.55 0.1808131 0.2233099
3  0.60 0.1941725 0.2455373
4  0.65 0.2112173 0.2743291
5  0.70 0.2157089 0.2805482
6  0.75 0.2291835 0.2985143
7  0.80 0.2461131 0.3223540
8  0.85 0.2598180 0.3372106
9  0.90 0.2702983 0.3475757
10 0.95 0.2716803 0.3487274
11 1.00 0.2736588 0.3507944
```

Figure 1: Building the function the calculate the prior probability of given dataset

To further test the algorithm, I did another two runs with the f equals 0.7 and 0.9. It surprises the result returns exactly the same as when f equals 0.4.

4 Experiment result and conclusions

The experiment does not look successful to me, as the accuracy is low and the f value has no effects on the result. There must be some problem with the logic and the implementation. It might be a coding error or misinterpretation in the building process. However, the most important lesson I have learned is about time management. It took me three months longer than expected to finish the project. For someone having a fresh start in learning computer science in theory, it was much hard than expected to make sense of every step of a complex algorithm. It gets even harder when you are multiple weeks behind the deadline and having drastic changes in the family and daily

job that made you shift the focus. I was ashamed to ask for help from the professor and my classmate, that I had to dive into the concept myself and read the slides time after time. Though the professor, you kindly mentioned that we could take as long as we needed to make this project professional. I think it is important to have a presentable product after such a long time. The experiment is not running as expected, and I am happy to continuously improve on this project and willing to take whatever score the professor believes is appropriate. I would say it should be over 100 hours I have spent on this project almost every weekend during the last three months. In this fulfilling study process, I learned a ton by repeatedly reading through the slides, watching the class recording, and reviewing the mathematical concepts I learned from my college time. I will continue focusing on this field as this is where my passion is.