Lauren Simms
Atholton High School
Johns Hopkins University Applied Physics Laboratory
APL ASPIRE Internship

Abstract - My internship during the 2022/2023 school year, I was assigned to use LaTex and create Python examples of the previously explained ideas using Anaconda development environment. So far I have researched machine learning I. Within machine learning I have focused on the Bayes Classifier algorithm and the Mahalanobis equation on the Iris data set.

Before starting my internship, I had experience coding projects with python and java for school assignments, but had zero experience with the LaTeX, or with complicated mathematical equations that make up the machine learning algorithms. Naively I had no idea there were other commonplace typesetting systems besides Microsoft Word, or Docs. I found LaTeX to be incredibly useful for documenting scientific and technical documents and my learning experience with it to be very similar to learning front end design work. The ability to recompile changes like code and visually see the changes immediately, and how to organize components with a specific syntax. The mathematical equations however I had more of a challenge with at the start. Changing my methodology to fully understanding each component of the equation allowed me to write correct code.

Learning how to write my code clearly and intentionally was also a skill I learned when writing my python examples. Previously, I mostly had worked on my own and was graded on my code's ability to complete the task. While writing the Latex for Dr. Rodriguez, my code not only must work, but be clear to anyone who viewed it with varying knowledge of python. These skills helped me navigate through the focus of my internship, the Bayes Classifier, and Mahalanobis distance equations.

In today's world, big data, is absolutely critical and drives everything from the social media posts on your phone, business decisions, to self-driving cars. This big data, as the name implies, is massive and include varies types that must be transformed out of its raw state to be understandable to computers to process. The ETL (Extract, Load, Transform) model is commonly used especially with the use of the cloud removing some of the barriers of storing mass amounts of data. There are often four major steps in the data transformations, identifying data, planning future steps in the transformation, coding how the data is transformed, and finally converting data from original to usable format. Some specifics transformations that the data may need are removing redundancies, combining data, filtering, which may improve the quality of data allowing higher intelligence to be mined. Data transformation often requires high computing capabilities and time specially to transform the large amounts of data that is being recorded.

The Iris data set was first used in 1936 by Ronald Fisher in his paper The Use of Multiple Measurements in Taxonomic Problems including examples of linear discriminant analysis. The data set includes three species of irises, Setosa, Versicolor, and Virginica, with 50 observations of each species. Each observation has four dimensions, the petal length, petal width, sepal length, and sepal width. Machine learning has become the future, a step further than programming machines, but machines adapt and learn from the data they collect. The Bayes Classifier algorithm is named after Thomas Bayes (1701-1761) and his ideas were published and contributed by Richard Price in "An Essay towards solving a Problem in the Doctrine of Chances" in 1763.

The Bayes uses vectors, specifically the mean vector, μ_j, and the covariance matrix, Σ_j. The vector $\mathbf{x}_0$ is a D-dimensional vector of the observed data, $|\Sigma_j|$ is the determinant of the covariance matrix, and Σ_j^{-1} is the inverse of the covariance matrix of the given class j. D is the number of dimensions, or features that the data is including. For the Iris data set the dimensions would be the petal length, petal width, sepal length, and sepal width.

The general multivariate normal density for D dimensions is defined as:

$$p(C_j|\mathbf{x}_0) = \frac{1}{\sqrt{(2\pi)^D|\Sigma_j|}} \exp\Big[-\frac{1}{2}\big(\mathbf{x}_0 - \mu_j\big)^T\Sigma_j^{-1}\big(\mathbf{x}_0 - \mu_j\big)\Big] \tag{1}$$

where the mathematical equation $\frac{1}{\sqrt{(2\pi)^D|\Sigma_j|}}$ is being termed the *normalizing constant* .

The Bayes' classifier (discriminate function) can be expressed in terms of the prior probabilities and conditional probability densities as follows:

$$p(C_j|\mathbf{x}_0)P(C_j) = P(x_0|C_j) = \frac{1}{\sqrt{(2\pi)^D|\Sigma_j|}} \exp\Big[-\frac{1}{2}\big(\mathbf{x}_0 - \mu_j\big)^T\Sigma_j^{-1}\big(\mathbf{x}_0 - \mu_j\big)\Big]P(C_j) \tag{2}$$

where the a priori probabilities $P(C_j)$ are the estimates of $\mathbf{x}_0$ belonging to a class C_j.

The Bayes classifier can be expressed in terms of the prior probabilities, $P(C_i)$, and posterior probability of class membership as follows:

$$P(C_j|x_0) = \frac{P(C_j)\frac{1}{\sqrt{(2\pi)^D|\Sigma_j|}} \exp\Big[-\frac{1}{2}\big(\mathbf{x}_0 - \mu_j\big)^T\Sigma_j^{-1}\big(\mathbf{x}_0 - \mu_j\big)\Big]}{\sum_{i=1}^{c} P(C_i)\frac{1}{\sqrt{(2\pi)^D|\Sigma_i|}} \exp\Big[-\frac{1}{2}\big(\mathbf{x}_0 - \mu_i\big)^T\Sigma_i^{-1}\big(\mathbf{x}_0 - \mu_i\big)\Big]} \tag{3}$$

where the a priori probabilities $P(C_j)$ are the estimates of belonging to a class and under the assumption that $\Sigma_j = \Sigma$ for $\forall j$. The classification decision assigns the the input observation $\mathbf{x}_0$ to the class C_j class with maximal posterior probability according to the inequality by a largest value returned to $C_{j,0}$. Lets consider a two class case where the classes are C_1 and C_2 as follows:

$$\begin{aligned} &\text{if} \\ &P(C_1|\mathbf{x}_0) > P(C_2|\mathbf{x}_0) \rightarrow \mathbf{x}_0 \text{ belongs to } C_1 \\ &\text{elseif} \\ &P(C_1|\mathbf{x}_0) \leq P(C_2|\mathbf{x}_0) \rightarrow \mathbf{x}_0 \text{ belongs to } C_2 \end{aligned} \tag{4}$$

If you have a multiclass problem you can calculate $P(C_j|\mathbf{x}_0)$ by taking the larger return value.

My first step in breaking down these equations was mathematically finding the components required, the mean, covariance matrix, and determinant.

The mean is average value of data given by

$$\mu = \frac{1}{n}\sum_{i=i}^{n} x_i \tag{5}$$

The covariance is the relationship of two random variables. A positive covariance means that the variables tend to vary in the same direction, while a negative covariance would vary in opposite directions. Very similar to the correlation of two variables, except the covariances are not bounded between -1 and 1. Covariances are often organized into a matrix in which the determinant can be calculated. The determinant is a scalar value that indicative of how a matrix behaves and its properties.

The covariance is given by

$$\mathbf{C} = \frac{1}{N-1}\sum_{n=1}^{N}(\mathbf{X}_n - \bar{\mathbf{X}})(\mathbf{X}_n - \bar{\mathbf{X}})^T \tag{6}$$

Determinant of Covariance Matrix

Numbers within the matrix are denoted by $|C_{ij}|$ with the first as the row, second the column, e.g., $|C_{2,2}|$ would be a covariance of size $[2 \times 2]$.

The determinant of a 2x2 matrix is given by

$$\det(C) = \begin{vmatrix} a & b \\ c & d \end{vmatrix} = (a * d) - (b * c)$$

However, for matrices of n x n size, can be determined with the Leibniz formula as follows:

$$\det(A) = \sum_{\sigma \in \mathbf{S}_n} (sgn(\sigma) \prod_{i=1}^{n} a_{i\sigma_i})$$

$\prod$ is the product of sequence of products within the indexes. Sgn or sign (σ) is the number of permutations required for the matrix to return to the column order from least to greatest. If the number of permutation is even, then the sign of the product is positive. If the number of permutations is odd, then the sign of the product is negative.

After understanding each component I computed the equations by hand. Below is an example of the Bayes Classifier equations on the first observation of Setosa with all four dimensions.

Setosa[1] = [5.1, 3.5, 1.4, 0.2]
Setosa mean = [5.006, 3.428, 1.5, 0.2]

Setosa Covariance matrix

$$\Sigma = \begin{bmatrix} 0.404343 & 0.093763 & 0.303290 & 0.049094 \\ 0.093763 & 0.104004 & 0.071380 & 0.047629 \\ 0.303290 & 0.071380 & 0.304588 & 0.048824 \\ 0.049094 & 0.047629 & 0.048824 & 0.075433 \end{bmatrix}$$

Setosa inverse covariance matrix

$$\Sigma_j^{-1} = \frac{1}{\begin{bmatrix} 0.404343 & 0.093763 & 0.303290 & 0.049094 \\ 0.093763 & 0.104004 & 0.071380 & 0.047629 \\ 0.303290 & 0.071380 & 0.304588 & 0.048824 \\ 0.049094 & 0.047629 & 0.048824 & 0.075433 \end{bmatrix}}$$

$p(C_j|\mathbf{x}_0)P(C_j) = P(x_0|C_j) = \frac{1}{\sqrt{(2\pi)^D|\Sigma_j|}} \exp\Big[-\frac{1}{2}\big(\mathbf{x}_0 - \mu_j\big)^T\Sigma_j^{-1}\big(\mathbf{x}_0 - \mu_j\big)\Big]P(C_j)$

$p(Setosa|setosa[1])\frac{1}{3} = P(setosa[1]|Setosa) = \frac{1}{\sqrt{(2\pi)^D|\Sigma_j|}} \exp\Big[-\frac{1}{2}\big(\mathbf{x}_0 - \mu_j\big)^T\Sigma_j^{-1}\big(\mathbf{x}_0 - \mu_j\big)\Big]P(C_j)$

$p(Setosa|setosa[1])\frac{1}{3} = P(setosa[1]|Setosa) = \frac{1}{\sqrt{(2\pi)^4|\Sigma_j|}} \exp\Big[-\frac{1}{2}\big([0.094, 0.072, -0.1, 0]\big)^T\Sigma_j^{-1}\big(\mathbf{x}_0 - \mu_j\big)\Big]\frac{1}{3}$

P(setosa[1]—Setosa) = 1.0
P(setosa[1]—Versicolor) = 2.4622784712161353e-19
P(setosa[1]—Virginica) = 1.5926333409107877e-35

After understanding how the equations worked, I used python in the spyder IDE to put them into code to efficiently run on the Iris dataset. Collaborating with other students in my group, I utilized the covariance code to create my own python code for the Bayes Classifier Equations. Running this code I was able to find the exact probabilities of the input observation $\mathbf{x}_0$ belonging to each class.

```python
import numpy as np
from numpy import linalg as la

class Bayes:
    def makeList(data, row):
        npList = data[row, :]
        return npList.tolist()

    def meanCalc(data, row):
        """ Find the mean of a dataset

        Keyword arguments:
        data -- the 2d numpy array
        row -- feature of data to calculate mean on

        """
        dataList = Bayes.makeList(data, row)

        sum = 0
        for item in dataList:
            sum += item
        n = len(dataList)
        mean = sum/n
        return mean

    def covarianceCalc(data, row1, row2):
        """ Find the covariance matrix of a 2d numpy array

        Keyword arguments:
        data -- the 2d numpy array
        row1, row2 -- the features of the dataset to calculate covariance on

        """

        dataList1 = Bayes.makeList(data, row1)
        dataList2 = Bayes.makeList(data, row2)

        mean1 = Bayes.meanCalc(data, row1)
        mean2 = Bayes.meanCalc(data, row2)

        sum = 0
        for i, item in enumerate(dataList1):
            zeroMean1 = dataList1[i] - mean1
            zeroMean2 = dataList2[i] - mean2

            sum += zeroMean1 * zeroMean2

        # population covariance, N, not N-1
        n = len(dataList1)

        covariance = sum/n

        return covariance

```

Listing 1: Python mean example

```
def covarianceMatrix(data):
        """ Find the covariance matrix of a 2d numpy array

        Keyword arguments:
        data -- the 2d numpy array

        """

        numRows = np.shape(data)[0] #

        covarianceMatrix = np.empty(shape = [numRows, numRows])

        for i, row in enumerate(covarianceMatrix):
            for j, col in enumerate(row):
                covarianceMatrix[i][j] = Bayes.covarianceCalc(data, i, j)

        return covarianceMatrix

```

Listing 2: Python example 1

The Bayes Classifier correctly identifies the Iris data set bar two observations. Versicolor[33] as Virginica and Virginica[33] as Versicolor.

Versicolor Misclassification
Versicolor[33] = [6.0, 2.7, 5.1, 1.6]
Versicolor Means = [5.936, 2.77, 4.26, 1.326]
Versicolor Mins = [4.9, 2, 3, 1]
Versicolor Maxes = [7, 3.4, 4.8, 1.8]

Versicolor Covariance matrix

$$\Sigma = \begin{bmatrix} 0.266433 & 0.085184 & 0.182898 & 0.055780 \\ 0.085184 & 0.098469 & 0.082653 & 0.041204 \\ 0.182898 & 0.082653 & 0.220816 & 0.073102 \\ 0.055780 & 0.041204 & 0.073102 & 0.039106 \end{bmatrix}$$

$|\Sigma_j|$ = Versicolor inverse covariance matrix

$$\Sigma_j^{-1} = \frac{1}{\begin{bmatrix} 0.266433 & 0.085184 & 0.182898 & 0.055780 \\ 0.085184 & 0.098469 & 0.082653 & 0.041204 \\ 0.182898 & 0.082653 & 0.220816 & 0.073102 \\ 0.055780 & 0.041204 & 0.073102 & 0.039106 \end{bmatrix}}$$

$$p(C_j|\mathbf{x}_0)P(C_j) \quad = \quad P(x_0|C_j) \quad = \quad \frac{1}{\sqrt{(2\pi)^D|\Sigma_j|}} \exp\Big[\; - \; \frac{1}{2}\big(\mathbf{x}_0 \; - \; \mu_j\big)^T\Sigma_j^{-1}\big(\mathbf{x}_0 \; - \; \mu_j\big)\Big]P(C_j)$$

$p(Versicolor|versicolor[33])\frac{1}{3} = P(versicolor[33]|Versicolor) = \frac{1}{\sqrt{(2\pi)^D|\Sigma_j|}} \exp\Big[-\frac{1}{2}\big(\mathbf{x}_0-\mu_j\big)^T\Sigma_j^{-1}\big(\mathbf{x}_0-\mu_j\big)\Big]P(C_j)$

```
import numpy as np
import math

def classify(x, mu, matrix):
    size = len(x)
    det = np.linalg.det(matrix) #determinant of matrix
    xmu = np.matrix(x - mu)
    t = xmu.transpose()
    inv = np.linalg.inv(setM)
    normConst = (1.0/(math.sqrt((2*math.pi)** size)*det))
    exp = (-1/2)*(xmu * inv * t)
    print(normConst)
    print(exp)
    print(normConst * exp)

def classify3():
    sum = 0
    prob_C1 = 50/150
    prob_C2 = 50/150
    prob_C3 = 50/150
    sumSet = 0
    sumVer = 0
    sumVir = 0
    for x in range(50):
        sumSet += prob_C1 * classify((np.array([setosa['petal_length'][x], setosa['petal_width'][x],
                                    setosa['sepal_length'][x], setosa['sepal_width'][x]]),
                        meanArrSet, setM)
        sumVer += prob_C2 * classify((np.array([versicolor['petal_length'][x], versicolor['petal_width'][x],
                                    versicolor['sepal_length'][x], versicolor['sepal_width'][x]])),
                        meanArrVer, verM)
        sumVir += prob_C3 * classify((np.array([virginica['petal_length'][x], virginica['petal_width'][x],
                                    virginica['sepal_length'][x], virginica['sepal_width'][x]])),
                        meanArrVir, virM)
    sum = sumSet + sumVer + sumVir

    return sum

print("Bayes classifier:",classify(dataSet, meanArrSet, setM)/classify3())
```

Listing 3: Python example 1

$p(Versicolor|versicolor[1])\frac{1}{3} = P(versicolor[1]|Versicolor) = \frac{1}{\sqrt{(2\pi)^4|\Sigma_j|}} \exp\Big[-\frac{1}{2}\big([0.064, -0.07, 0.84, 0.274]\big)^T \Sigma_j^{-1}$

P(versicolor[33]—Setosa) = 7.752265894279429e-116
P(versicolor[33]—Versicolor) = 0.3257957802868816
P(versicolor[33]—Virginica) = 0.6742042197131184

The Bayes Classifier Equations calculated that there is a 67.4% chance that the data belonged to Virginica class , and a 32.6% chance that the data belonged to the Versicolor class.

Virginica Misclassification
Virginica [33] = [6.3, 2.8, 5.1, 1.5]
Virginica Means = [6.588, 2.974, 5.552, 2.026]
Virginica Mins = [4.9, 2.5, 4.5, 1.5]
Virginica Maxes = [7.9, 3.8, 6.9, 2.5]

Virginica Covariance matrix

$$\Sigma = \begin{bmatrix} 0.404343 & 0.093763 & 0.303290 & 0.049094 \\ 0.093763 & 0.104004 & 0.071380 & 0.047629 \\ 0.303290 & 0.071380 & 0.304588 & 0.048824 \\ 0.049094 & 0.047629 & 0.048824 & 0.075433 \end{bmatrix}$$

Virginica inverse covariance matrix

$$\Sigma_j^{-1} = \frac{1}{\begin{bmatrix} 0.404343 & 0.093763 & 0.303290 & 0.049094 \\ 0.093763 & 0.104004 & 0.071380 & 0.047629 \\ 0.303290 & 0.071380 & 0.304588 & 0.048824 \\ 0.049094 & 0.047629 & 0.048824 & 0.075433 \end{bmatrix}}$$

$p(C_j|\mathbf{x}_0)P(C_j) = P(x_0|C_j) = \frac{1}{\sqrt{(2\pi)^D|\Sigma_j|}} \exp\Big[-\frac{1}{2}\big(\mathbf{x}_0 - \mu_j\big)^T \Sigma_j^{-1}\big(\mathbf{x}_0 - \mu_j\big)\Big] P(C_j)$

$p(Virginica|virginica[33])\frac{1}{3} = P(virginica[33]|Virginica) = \frac{1}{\sqrt{(2\pi)^D|\Sigma_j|}} \exp\Big[-\frac{1}{2}\big(\mathbf{x}_0 - \mu_j\big)^T \Sigma_j^{-1}\big(\mathbf{x}_0 - \mu_j\big)\Big] P(C_j)$

$p(Virginica|virginca[33])\frac{1}{3} = P(virginica[33]|Virginica) = \frac{1}{\sqrt{(2\pi)^4|\Sigma_j|}} \exp\Big[-\frac{1}{2}\big([-0.288, -0.174, -0.452, -0.526]\big) \Sigma_j^{-1}\big([-0.288, -0.174, -0.452, -0.526]\big)\Big]\frac{1}{3}$

P(virginica[33]—Setosa) = 8.018713522096487e-113
P(virginica[33]—Versicolor) = 0.8021542652147471
P(virginica[33]—Virginica) = 0.19784573478525286

The Bayes Classifier Equations calculated that there is a 80.2% chance that the data belonged to Versicolor class , and a 19.8% chance that the data belonged to the Virginica class.

To further analyze the misclassifications, I used Mahalanobis distance, which utilizes the same metrics as the Bayes classifier, mean and covariance, to explain why these observations were misclassified.

Here the Mahalanobis distance is calculated for each observation within a class using the class mean μ by feature and the class inverse covariance matrix. Σ^{-1}. The Mahalanobis distance is included in the exponenet of the Bayes Classifier equations, using the same metrics, the mean and covariance.

$$D_i = \left((x_i - \mu)\Sigma^{-1}(x_i - \mu)^T\right)^{1/2} \tag{7}$$

In the figure below the Setosa species is shown in blue, Versicolor species is shown in red, and the Virginica species is shown in green. The x-axis is the sepal length and the y-axis is the petal width for this figure.

The Mahalanobis distance can be used to identify the observations furthest from the mean by feature while accounting for the variance of each feature and the rotation of the data with the use of the covariance matrix. In this approach, the values returned from the Mahalanobis distance are sorted in descending with the value with the largest distance analyzed as a potential outlier. It should be apparent that the three observations for each class circled in black appear to be identified as outliers. These observations can be removed with the potential benefit of having a tighter standard deviation of the data being analyzed.

The two misclassifications are defined as outliers by the Mahalanobis distance

The distance for the versicolor misclassification is : 2.844105292479585
The distance for a versicolor observation close to the mean is : 1.1866961563856047

The distance for the virginica misclassification is : 2.069951927774073
The distance for a virginica observation close to the mean is : 1.251419518563993

Throughout the duration of my APL ASPIRE internship, I have learned invaluable skills and information under the mentorship of Dr. Benjamin Rodriguez. Certain things I did not expect to have to adjust to such as learning how to document using Latex and collaborating with other students virtually allowed me to grow as a student and computer scientist. This introduction to machine learning by comprehending complex mathematical equations into python algorithms, and analyzing them with other equations gave me an insight into a field that I understood in abstract, but not in practice. I am incredibly grateful for this opportunity and all those who have helped me this year.

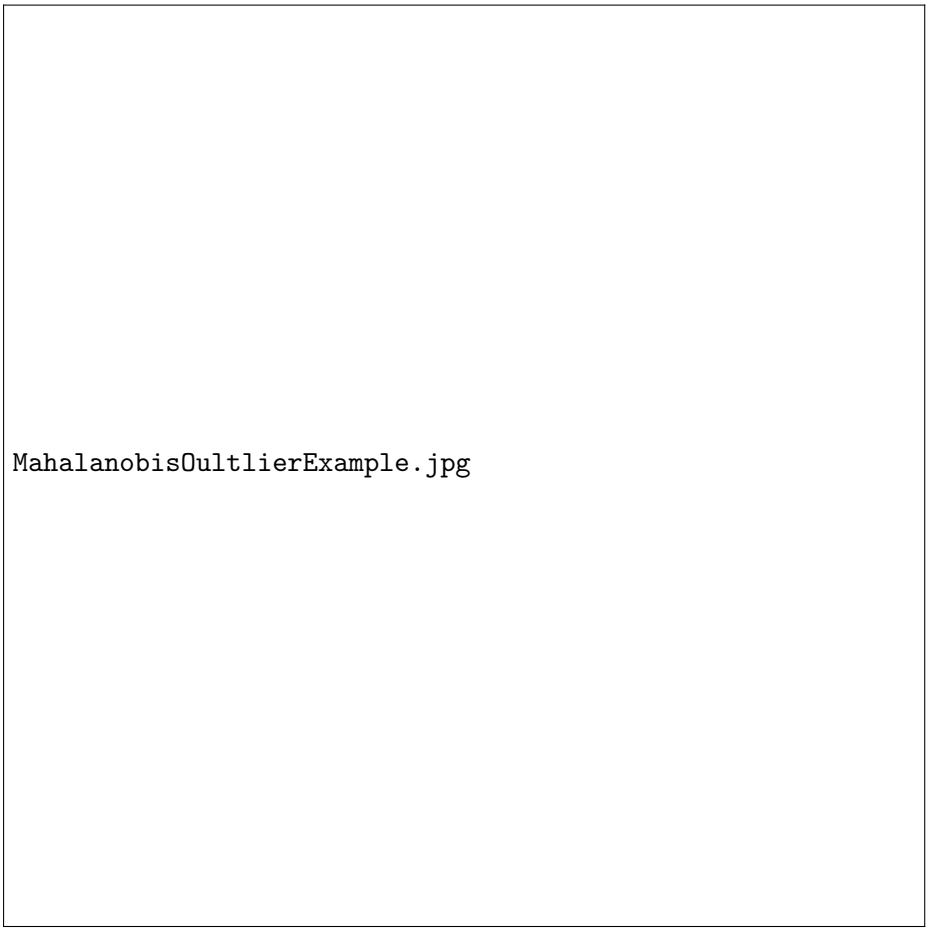

Figure 1: Outlier Identification - Here the Iris data by flower type has the observation with the largest observation identified based on the Mahalanobis distance.

The rings represent the covariance, with the colored ring as the mean of each species The misclassifications are the dots outlined in black

References

[1] Rodriguez, Benjamin; Chen, Mei-Chin; Saeed, Amir; "Algorithms for Data Science", JHU.