EECE 5644 Assignment 1

Eric Grimaldi

2020 October 12

Foreward on Code Libraries

For reference:

The code for this assignment was completed in *Python*. The *NumPy* library is used extensively for linear algebra and random variables. The *SciPy* library is used briefly for a linear algebra function with improved numerical calculation. The *MatPlotLib* and *Seaborn* libraries are used for data visualization. The *Pandas* library is used briefly for a database format compatible with a useful *Seaborn* function.

Question 1

As per the assignment:

X is a four-dimensional random vector with the PDF below:

$$p(x) = p(x|L = 0)P(L = 0) + p(x|L = 1)P(L = 1)$$

$L = l$ signifies the true class label. The class priors are $P(L = 0) = 0.7$ and $P(L = 1) = 0.3$. The class-conditional PDFs are $p(x|L = 0) = g(x|m_0, C_0)$ and $p(x|L = 1) = g(x|m_1, C_1)$ where $g(x|m, C)$ is a multivariate Gaussian PDF with mean m and covariance matrix C. The parameters in this case are

$$m_0 = \begin{bmatrix} -1 \\ -1 \\ -1 \\ -1 \end{bmatrix} C_0 = \begin{bmatrix} 2 & -0.5 & 0.3 & 0 \\ -0.5 & 1 & -0.5 & 0 \\ 0.3 & -0.5 & 1 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix}$$

$$m_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \; C_1 = \begin{bmatrix} 1 & 0.3 & -0.2 & 0 \\ 0.3 & 2 & 0.3 & 0 \\ -0.2 & 0.3 & 1 & 0 \\ 0 & 0 & 0 & 3 \end{bmatrix}$$

For numerical results below, we generate 10000 samples according to this distribution. To do this, we first randomly choose a true label according to the class priors, and then randomly draw a sample data point from the appropriate class-conditional PDF by using $numpy.random.multivariate\_normal()$. Whenever the code is run, an object is initialized which contains the data set generated for for that run. All parts of the question are completed with a single data set in a single run of the code.

Part A: Expected Risk Minimization using True PDFs

As provided in $L02a\_ExpectedRiskMinimizationBasedClassifierDesign.pdf$:

The likelihood-ratio test for the 2-class case of ERM is as follows:

$$\frac{p(x|L=1)}{p(x|L=0)} \underset{D=0}{\overset{D=1}{\gtrless}} \frac{\lambda_{10} - \lambda_{00}}{\lambda_{01} - \lambda_{11}} \frac{P(L=0)}{P(L=1)} = \gamma$$

where λ_{ij} is the loss value for $(D = i|L = j)$. For the minimum expected risk case, λ_{ij} is 0 $\forall i = j$ and is 1 $\forall i \neq j$. This reduces the LRT to :

$$\frac{p(x|L=1)}{p(x|L=0)} \underset{D=0}{\overset{D=1}{\gtrless}} \frac{P(L=0)}{P(L=1)} = \gamma$$

After fully substitution and simplification, this becomes:

$$\frac{|C_1|^{-\frac{1}{2}} e^{-\frac{1}{2}(x-m_1)^T C_1^{-1}(x-m_1)}}{|C_0|^{-\frac{1}{2}} e^{-\frac{1}{2}(x-m_0)^T C_0^{-1}(x-m_0)}} \underset{D=0}{\overset{D=1}{\gtrless}} \frac{0.7}{0.3} = 2.\overline{3} = \gamma$$

As per the assignment, instead of using the theoretically optimal γ, we vary slowly from 0 to ∞. Since we have a finite number of data points x_i, we are able to choose a finite number of γ_i in order to accomplish this. We choose 10001 γ such that $\gamma_0 = 0$, γ_i is halfway between the result of the ith LRT and the $(i+1)$th LRT, and the final γ_{10000} is double the 10000th LRT for good measure.

For each γ_i we classify the entire data set, keeping track of the decision-label pairs, and calculating the confusion matrix quantities for each γ_i (True Positive rate, False Positive rate, True Negative rate, False Negative rate).

Using the False Positive and True Positive rates across all γ_i as (x, y) data we plot the ROC curve for this ERM classifier. See $Figure1$ for an example ROC curve.

We also empirically determine $\gamma_{\hat{P}_e}$ which minimizes the probability of error by simply keeping track of the lowest P_e as we move through γ for the ROC curve. $\gamma_{\hat{P}_e}$ is superimposed on the ROC curve, as you can see in $Figure1$, along with the relevant $\hat{P}_e$, TP, and TN.

The exact $\gamma_{\hat{P}_e}$ and $\hat{P}_e$ varies depending on the data set generated. Typically we see $\gamma_{\hat{P}_e}$ hover around 2.1-2.5, with occasional jumps as far as 1.8 or 2.8. $\hat{P}_e$ hovers around 2.6-3.3%. Since the code is somewhat slow to run, these range estimates should be taken with a large pinch of salt. Regardless of our uncertainty due to the small sample size of runs, our empirically estimated $\gamma_{\hat{P}_e}$ seems reasonably near our theoretically optimal $\gamma_{\hat{P}_e}$ of $2.\overline{3}$.

Part B: ERM using a Naive Bayesian Assumption

We repeat the previous analysis under an incorrect Naive Bayesian assumption.

For this analysis we use the correct priors and means from the previous part, however C_0 and C_1 are replaced with:

$$C_{0NB} = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix} C_{1NB} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 3 \end{bmatrix}$$

For this part, our analysis methods are fully the same. Note, however, that we do choosing a second set of γ_i for the NB case, since our LRT computations will have somewhat different results. The method for these choices are identical, however.

In order to analyze the impact of this modelling assumption, the resulting ROC curve, $\gamma_{\hat{P}_e}$, and $\hat{P}_e$ are superimposed on the plot of the previous analysis. A typical case can be seen in $Figure1$. The Naive Bayesian assumption negatively impacts our classifier's performance. The ROC curve is slightly worse near the perfect detection point of (0,1). $\hat{P}_e$ is typically 1-2% worse. $\gamma_{\hat{P}_e}$ farther from the theoretically optimal point of $2.\overline{3}$, and is strangely prone to hovering around 1.3-1.7.

Part C: Fisher Linear Discriminant Analysis

We repeat the analysis a third time using a Fisher LDA classifier instead of an ERM classifier.

As provided in $L01b\_FisherLDA.pdf$, the general form of an LDA classifier is as follows:

$$y_{LDA} = w_{LDA}^T x \underset{D=0}{\overset{D=1}{\gtrless}} \tau$$

Where w_{LDA} is the eigen vector with the largest eigen value of the generalized eigendecomposition of the form $S_B w = (\frac{w^T S_B w}{w^T S_W w}) w S_W$ where S_B is the scattering between classes $S_B = (m1 - m0)(m1 - m0)^T$ and S_W is the scattering within classes $S_W = C_1 + C_2$.

As per the assignment, we estimate the mean vectors and covariance matrices using sample averages of the data set. From these, we calculate w_{LDA}. From that, we calculate the Linear Discriminant Score for each data point, for the purpose of classifying the points by comparing them to a threshold τ. Note, we use a similar method to choose 10001 τ as we did for γ in the previous parts - though in this case τ varies from $-\infty$ to ∞. Finally, we use a similar method to classify the entire data set for each τ, and to use these decision-label pairs to produce an ROC curve, $\tau_{\hat{P}_e}$, and $\hat{P}_e$ for our LDA classifier.

In order to compare the performance of this classifier to our previous two classifiers, the resulting ROC curve, $\tau_{\hat{P}_e}$, and $\hat{P}_e$ are superimposed on the plot of the previous analyses. A typical case can be seen in $Figure1$. The LDA classifer performs much worse than both the true ERM classifier and the Naive Bayesian ERM classifier. The LDA ROC curve is very obviously worse. $\hat{P}_e$ is typically 10-20%. $\tau_{\hat{P}_e}$ doesn't seem strictly worthwhile to compare to $\tau_{\hat{P}_e}$, but for reference it was typically a small negative number, perhaps -1.5 or -0.5. The negative impact of the data compression inherent to LDA seems apparent.

Question 2

As per the assignment:

X is a three-dimensional random vector that takes values from a mixture of four Gaussians. Each Gaussian pdf is the class-conditional pdf for one of four labels $L \in \{1, 2, 3, 4\}$. Class priors are $P(L = 1) = 0.2$, $P(L = 2) = 0.25$, $P(L = 3) = 0.25$, and $P(L = 4) = 0.3$.We select our class-conditional pdfs to have mean vectors located on a square:

$$m_1 = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} m_2 = \begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix} m_3 = \begin{bmatrix} -1 \\ -1 \\ 0 \end{bmatrix} m_4 = \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix}$$

We also select our class-conditional matrices to have covariances such that our pdfs are spherically symmetric (i.e. $C_i = \lambda_i I$ where I is the three-dimensional identity matrix). In the interest of having a reasonably instructive amount of overlap between the different Gaussians/labels, we select said $\lambda = 0.35 \ \forall i$, which is approximately 10-20% of the distance between any two mean vectors.

We generate 10000 data points using the same method as before: we first randomly choose a label according to the class priors, and then randomly draw a data point from the corresponding Gaussian. The same data set is used in both part A and part B.

Part A: ERM with 0-1 Loss (MAP)

As provided in $L02a\_ExpectedRiskMinimizationBasedClassifierDesign.pdf$:

For ERM generalized to n classes, the decision rule is

$$D = \underset{i}{argmin} R = \underset{i}{argmin} \Lambda P$$

Where Decision D is equal to index i of the minimum risk in risk vector R, which is equal to loss matrix Λ times posterior vector P. In this case:

$$R = \begin{bmatrix} R(D = 1|x) \\ R(D = 2|x) \\ R(D = 3|x) \\ R(D = 4|x) \end{bmatrix} P = \begin{bmatrix} P(L = 1|x) \\ P(L = 2|x) \\ P(L = 3|x) \\ P(L = 4|x) \end{bmatrix}$$

After Bayes Rule, P becomes:

$$P = \frac{1}{p(x)} \begin{bmatrix} P(x|L=1)P(L=1) \\ P(x|L=2)P(L=2) \\ P(x|L=3)P(L=3) \\ P(x|L=4)P(L=4) \end{bmatrix}$$

For the specified minimum probability of error (0-1 loss), Λ becomes 0 for correct decisions (along the diagonal) and 1 for incorrect decisions (elsewhere):

After substitution and simplification, the decision rule becomes:

$$D = \underset{i}{argmin} \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} |C_1|^{-\frac{1}{2}} e^{-\frac{1}{2}(x-m_1)^T C_1^{-1}(x-m_1)} \\ |C_2|^{-\frac{1}{2}} e^{-\frac{1}{2}(x-m_2)^T C_2^{-1}(x-m_2)} \\ |C_3|^{-\frac{1}{2}} e^{-\frac{1}{2}(x-m_3)^T C_3^{-1}(x-m_3)} \\ |C_4|^{-\frac{1}{2}} e^{-\frac{1}{2}(x-m_4)^T C_4^{-1}(x-m_4)} \end{bmatrix}$$

After classifying each data point according to this decision rule we check what the true label was, and count that towards a total for that decision-label pair. After classifying the entire data set, we compute the confusion matrix where each entry is the number of times the classifier arrived at that decision-label pair divided by the total number of instances of that true label. Note, the diagonal entries of the confusion matrix represent and estimation of the likelihood that a given label is correctly classified. See $Figure2$ for an typical confusion matrix for this MAP case. Approximately 90-92% success rates for all four classes seems typical, though label 1 is slightly lower.

Finally, we plot the data set compressed onto the $x1, x2$ plane. Since we chose the mean vectors to be points on a square in that plane, this should be sufficient for visualizing the classifier. See $Figure3$ for reference. It seems visually obvious that the classifier is performing reasonably well.

Part B: ERM with a Loss Matrix

Here, we repeated the analysis from the previous part with one small change:

Instead of 0-1 loss we use a full loss matrix:

$$\Lambda = \begin{bmatrix} 0 & 1 & 2 & 3 \\ 10 & 0 & 5 & 10 \\ 20 & 10 & 0 & 1 \\ 30 & 210 & 1 & 0 \end{bmatrix}$$

The methods and calculations used are otherwise practically identical. However, due to the vastly changed Λ, we have vastly different results. As one can see in a typical confusion for this ERM case, we now have an extremely high success rate around 97-99% for label 1, at the cost of of lower success rates for other labels. In particular, label 2 tends to be quite low at around 75%. See $Figure4$ for a typical confusion matrix for this ERM case.

Visually, the impact of the new loss matrix is obvious and profound. I imagine many people would balk at the visualization of this classifier, and deem it a failure. See $Figure5$ for reference.

However, one additional feature was added to this part. As we classified the entire data set according to this loss matrix, we kept a "running total loss" (i.e. after a decision $D = i|L = j$, Λ_{ij} was added to the running total loss). After the entire set is classified, dividing the total loss by the number of data points provides an estimate of expected risk.

Despite the large magnitude of some elements of the loss matrix, the ERM classifier was able to keep the expected loss value very low, below 0.5 typically.

This measurement shows the purpose of a full ERM loss matrix: while technically it may result in more errors of some types, it will very specifically minimize errors that have been deemed costly, even if it means a larger number of relatively less costly errors near boundaries. One thinks back to the example of classifying cancer cells...

Appendix A: Code

https://github.com/EAGrimaldi/EECE5644_Assigment_1

Appendix B: Figures

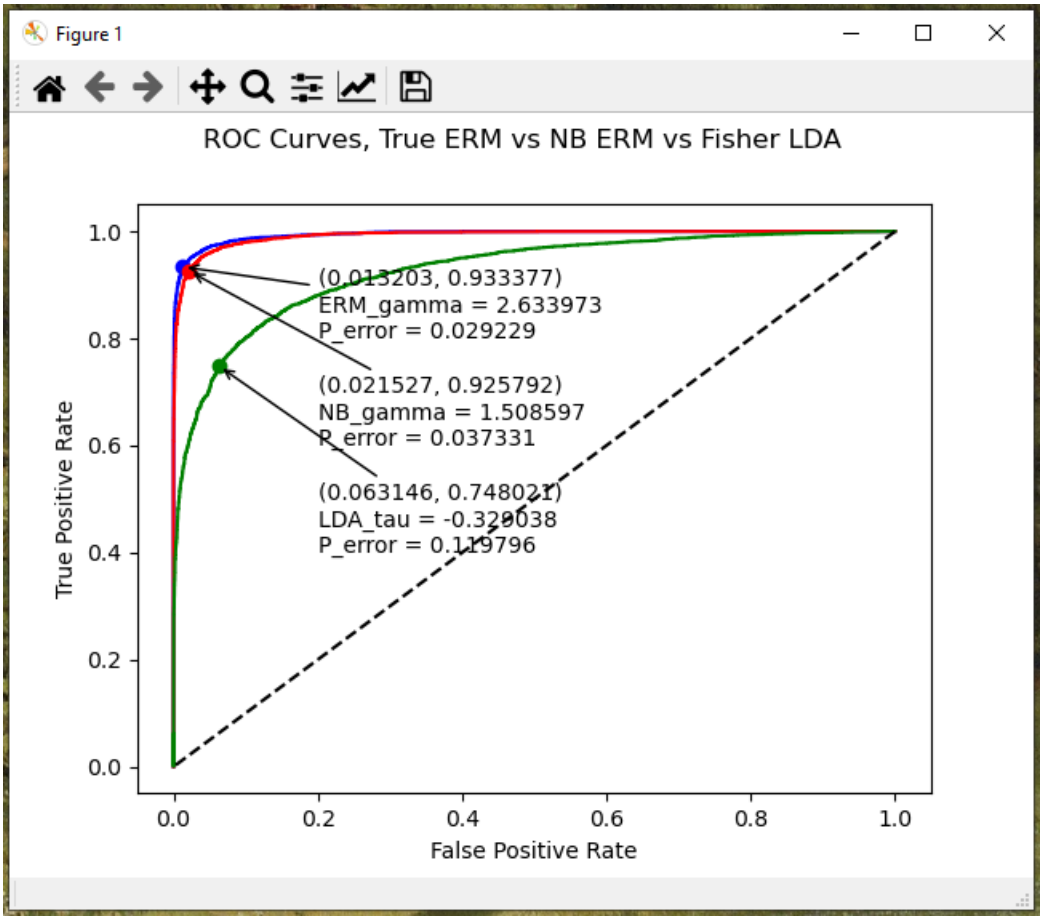

Figure 1: example ROC curves
True ERM in blue, NB ERM in red, Fisher LDA in green
The black dashed line represents random guesses

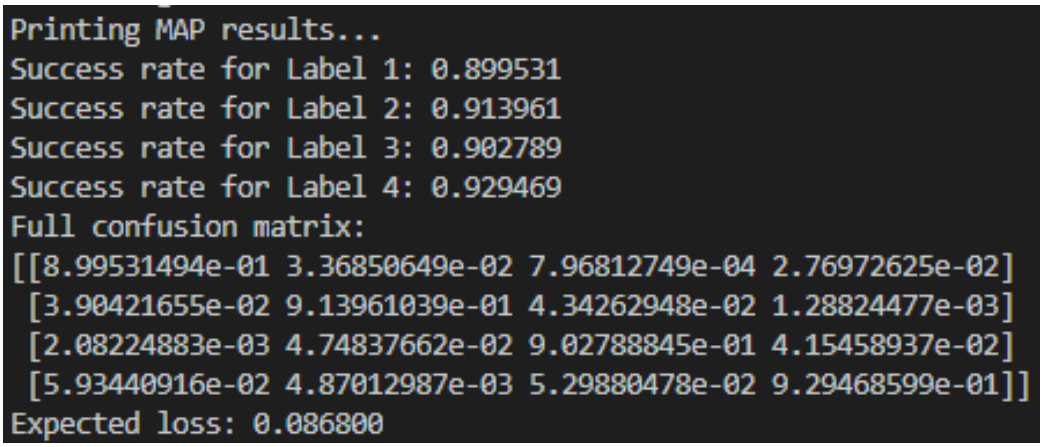

Figure 2: example confusion matrix for the MAP case

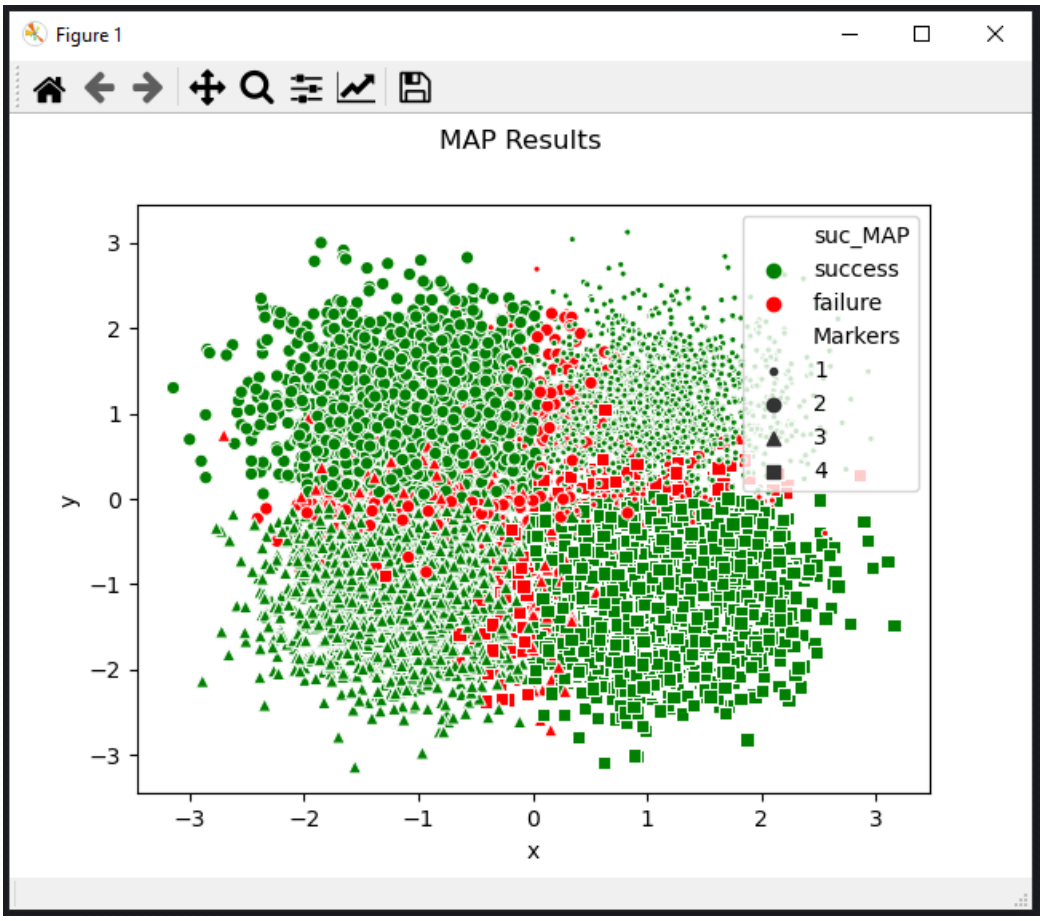

Figure 3: example plot for the MAP case

```
Printing ERM results...
Success rate for Label 1: 0.981780
Success rate for Label 2: 0.772321
Success rate for Label 3: 0.850598
Success rate for Label 4: 0.840258
Full confusion matrix:
[[9.81780323e-01 2.11444805e-01 8.72509960e-02 1.12721417e-01]
 [5.20562207e-03 7.72321429e-01 1.83266932e-02 3.22061192e-04]
 [2.08224883e-03 1.62337662e-02 8.50597610e-01 4.66988728e-02]
 [1.09318064e-02 0.00000000e+00 4.38247012e-02 8.40257649e-01]]
Expected loss: 0.371400
```

Figure 4: example confusion matrix for the ERM case

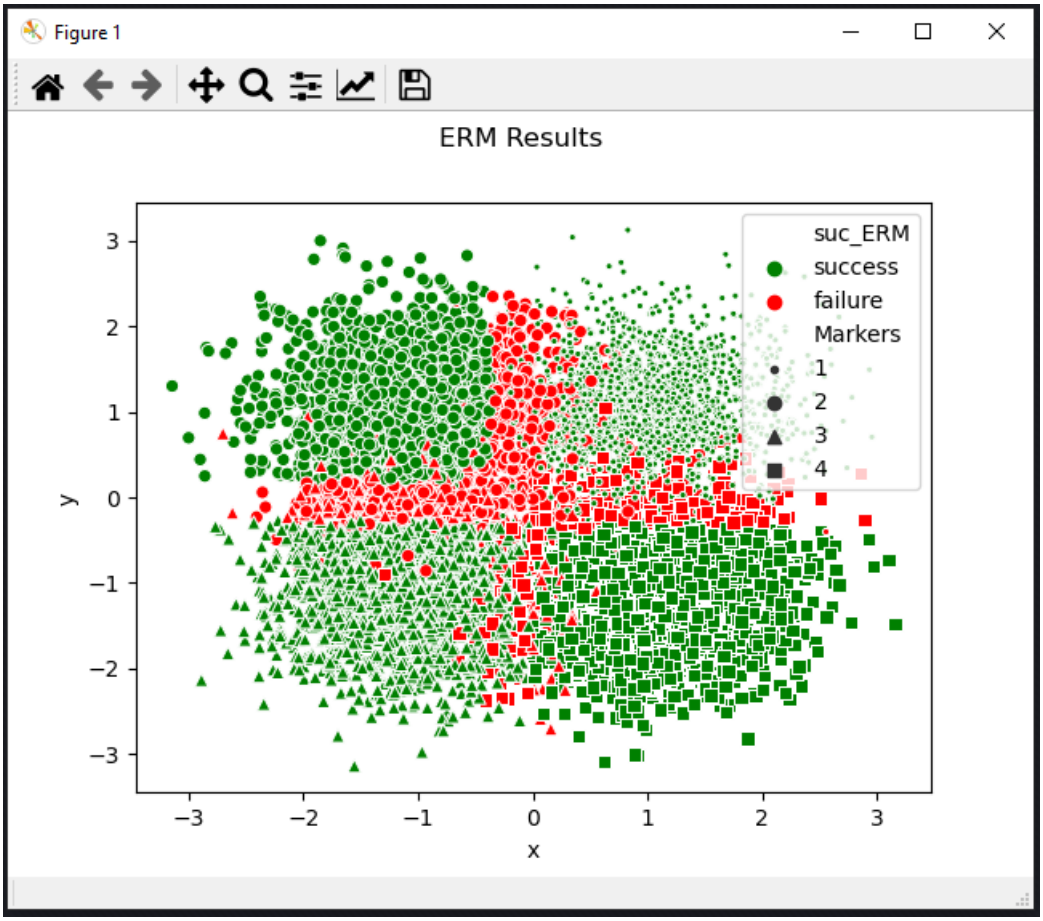

Figure 5: example plot for the ERM case