Anthony Yalong

NUID: 002156860

EECE5644

10 November 2025

Assignment #3

1. Question #1

1.1. Background

This problem explores how well multi-layer perceptrons (MLPs) can approximate optimal classification performance when trained on limited data. The goal is to train MLPs to classify 3-dimensional data into 4 classes and compare their performance against the theoretical optimum.

1.2. Methodology

To generate the data, `numpy` was used to sample from 4 Gaussian distributions with uniform class priors of 25% each. The distributions were positioned in 3D space to create a moderately challenging classification problem. The class means were positioned at the corners of a cube: Class 1 at [0, 0, 0], Class 2 at [2.5, 2.5, 0], Class 3 at [0, 2.5, 2.5], and Class 4 at [2.5, 0, 2.5]. This positioning creates natural separation between classes while maintaining some overlap. For the covariance structures, a mix of diagonal and non-diagonal matrices was employed. Classes 1 and 2 have diagonal covariances scaled by 1.5, while Class 3 has small positive correlations (0.3 off-diagonal) and Class 4 has small negative correlations (-0.3 off-diagonal). This setup produced a theoretical optimal error rate of approximately 18.75%, falling within the target 10-20% range. Training sets of varying sizes (100, 500, 1000, 5000, and 10000 samples) were generated along with a large test set of 100,000 samples for reliable performance evaluation.

Using the true distribution parameters, the MAP (Maximum A Posteriori) classifier was implemented with `scipy` and `numpy` to establish the theoretical baseline. This represents the best possible performance achievable for this classification problem and serves as the benchmark against which the trained MLPs are compared.

A simple two-layer neural network was built using Keras with one hidden layer containing a variable number of perceptrons (P), followed by a 4-unit softmax output layer. For the hidden layer activation function, ELU was chosen, which is a smooth activation that helps with gradient flow during training. The softmax output layer ensures that the network outputs valid probability distributions over the four classes. For optimization, the Adam optimizer was used with a weight decay of 1e-4, which provides adaptive learning rates for each parameter and helps prevent overfitting by penalizing large weights.

The training strategy was designed to balance model capacity with generalization ability. Each model was trained for a maximum of 100 epochs with a batch size of 64 for efficient gradient updates. To prevent overfitting, early stopping was implemented with a patience of 10 epochs, which monitors the training loss and stops training if it does not improve for 10 consecutive epochs. When early stopping triggers, the model weights are restored to the best checkpoint encountered during training. This approach allows the model to train long enough to learn the patterns in the data while avoiding the degradation that comes from training too long.

For each training set size, 10-fold cross-validation was performed to select the optimal number of perceptrons from the candidate values: 1, 2, 3, 5, 7, 10, 15, and 20. The cross-validation process works by splitting the training data into 10 folds, then for each perceptron count, training on 9 folds and validating on the remaining fold. This is repeated for all 10 folds, and the validation errors are averaged to get a robust estimate of model performance. The perceptron count with the lowest average validation error is selected as optimal for that training set size.

Once the best number of perceptrons for each training set size was identified, the final model was trained using that architecture on the full training set. For each configuration, 5 seperate models were trained with different random initializations (which Keras handles automatically when creating a new model) and the one with the best training performance was kept. This multiple-initialization strategy helps ensure that the final model represents a good solution to the optimization problem rather than a local minimum.

1.3. Results

1.3.1. Theoretical Optimum Mathematical Derivation

- Number of classes: $C = 4$
- Class priors: $P(L = j) = 0.25$ for $j \in \{1, 2, 3, 4\}$ (uniform)
- Class-conditional densities: $p(\mathbf{x}|L = j) = \mathcal{N}(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$

The mean vectors are:

$$\boldsymbol{\mu}_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad \boldsymbol{\mu}_2 = \begin{bmatrix} 2.5 \\ 2.5 \\ 0 \end{bmatrix}, \quad \boldsymbol{\mu}_3 = \begin{bmatrix} 0 \\ 2.5 \\ 2.5 \end{bmatrix}, \quad \boldsymbol{\mu}_4 = \begin{bmatrix} 2.5 \\ 0 \\ 2.5 \end{bmatrix}$$

The covariance matrices are:

$$\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2 = 1.5\mathbf{I}_3 = \begin{bmatrix} 1.5 & 0 & 0 \\ 0 & 1.5 & 0 \\ 0 & 0 & 1.5 \end{bmatrix}$$
$$\boldsymbol{\Sigma}_3 = \begin{bmatrix} 1.8 & 0.3 & 0.3 \\ 0.3 & 1.8 & 0.3 \\ 0.3 & 0.3 & 1.8 \end{bmatrix}$$
$$\boldsymbol{\Sigma}_4 = \begin{bmatrix} 1.8 & -0.3 & 0 \\ -0.3 & 1.8 & 0 \\ 0 & 0 & 1.8 \end{bmatrix}$$

Figure 1.3.1.1 – Problem Setup

The MAP classifier minimizes the probability of error by selecting the class with the highest posterior probability:

$$\hat{L}(\mathbf{x}) = \arg \max_{j \in \{1,2,3,4\}} P(L = j|\mathbf{x})$$

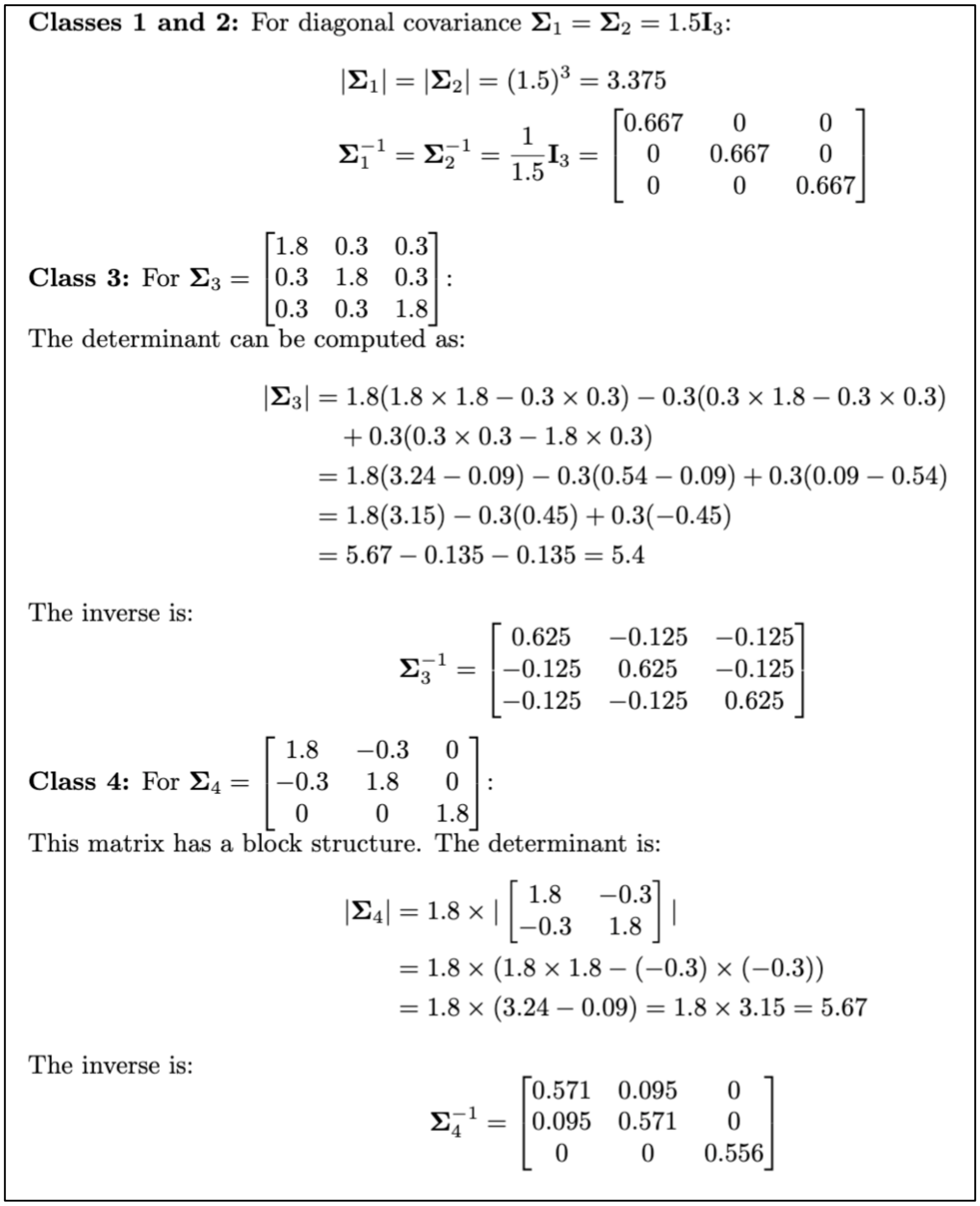
Using Bayes' theorem:

$$P(L = j|\mathbf{x}) = \frac{p(\mathbf{x}|L = j)P(L = j)}{p(\mathbf{x})} = \frac{p(\mathbf{x}|L = j)P(L = j)}{\sum_{i=1}^{4} p(\mathbf{x}|L = i)P(L = i)}$$

Since all class priors are equal ($P(L = j) = 0.25$), they cancel out in the comparison, and the MAP rule simplifies to:

$$\hat{L}(\mathbf{x}) = \arg \max_{j \in \{1,2,3,4\}} p(\mathbf{x}|L = j)$$

Figure 1.3.1.2 – MAP Decision Rule



Classes 1 and 2: For diagonal covariance $\mathbf{\Sigma}_1 = \mathbf{\Sigma}_2 = 1.5\mathbf{I}_3$:

$$|\mathbf{\Sigma}_1| = |\mathbf{\Sigma}_2| = (1.5)^3 = 3.375$$

$$\mathbf{\Sigma}_1^{-1} = \mathbf{\Sigma}_2^{-1} = \frac{1}{1.5}\mathbf{I}_3 = \begin{bmatrix} 0.667 & 0 & 0 \\ 0 & 0.667 & 0 \\ 0 & 0 & 0.667 \end{bmatrix}$$

Class 3: For $\mathbf{\Sigma}_3 = \begin{bmatrix} 1.8 & 0.3 & 0.3 \\ 0.3 & 1.8 & 0.3 \\ 0.3 & 0.3 & 1.8 \end{bmatrix}$:
The determinant can be computed as:

$$\begin{aligned} |\mathbf{\Sigma}_3| &= 1.8(1.8 \times 1.8 - 0.3 \times 0.3) - 0.3(0.3 \times 1.8 - 0.3 \times 0.3) \\ &\quad + 0.3(0.3 \times 0.3 - 1.8 \times 0.3) \\ &= 1.8(3.24 - 0.09) - 0.3(0.54 - 0.09) + 0.3(0.09 - 0.54) \\ &= 1.8(3.15) - 0.3(0.45) + 0.3(-0.45) \\ &= 5.67 - 0.135 - 0.135 = 5.4 \end{aligned}$$

The inverse is:

$$\mathbf{\Sigma}_3^{-1} = \begin{bmatrix} 0.625 & -0.125 & -0.125 \\ -0.125 & 0.625 & -0.125 \\ -0.125 & -0.125 & 0.625 \end{bmatrix}$$

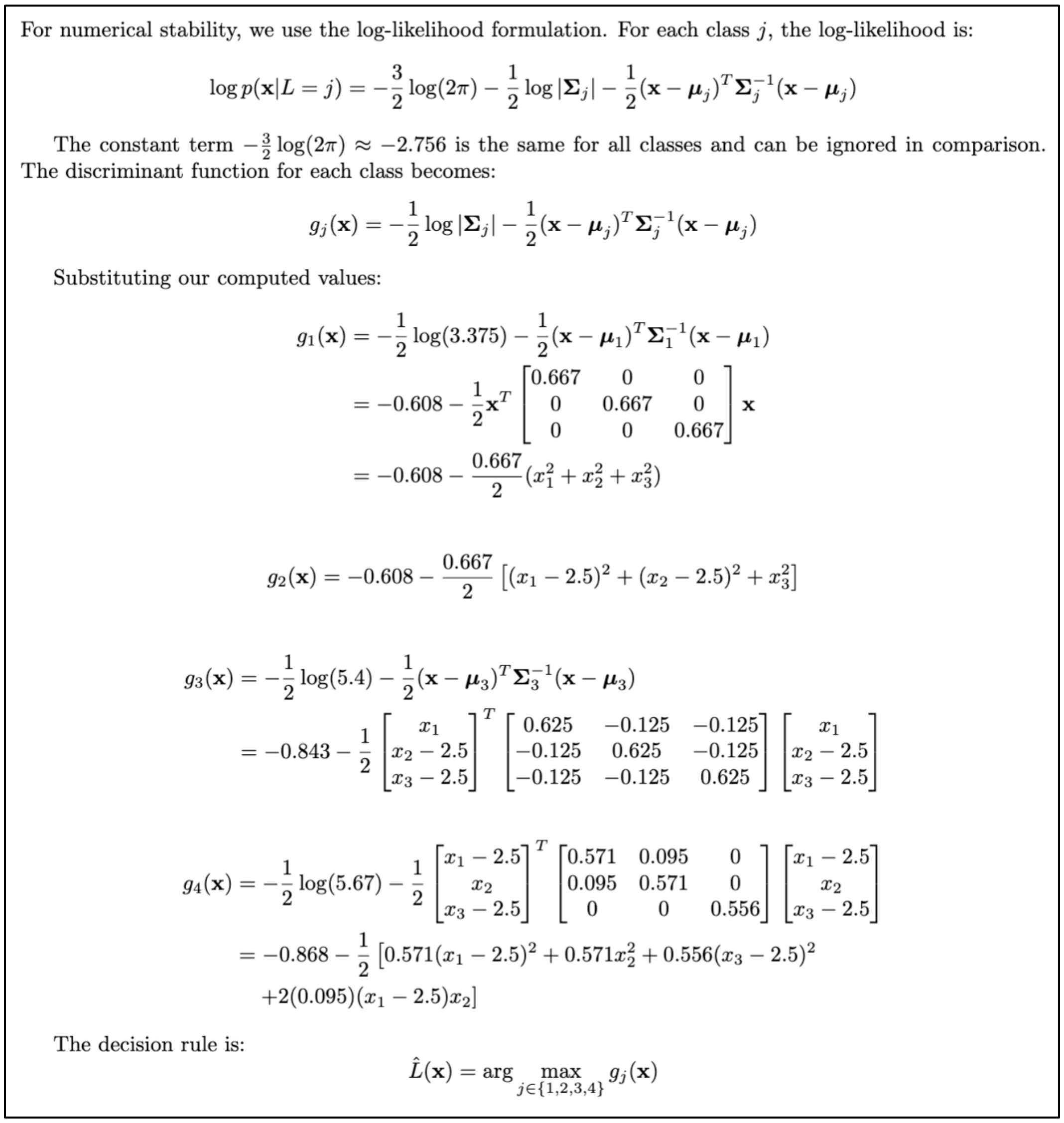
Class 4: For $\mathbf{\Sigma}_4 = \begin{bmatrix} 1.8 & -0.3 & 0 \\ -0.3 & 1.8 & 0 \\ 0 & 0 & 1.8 \end{bmatrix}$:
This matrix has a block structure. The determinant is:

$$\begin{aligned} |\mathbf{\Sigma}_4| &= 1.8 \times | \begin{bmatrix} 1.8 & -0.3 \\ -0.3 & 1.8 \end{bmatrix} | \\ &= 1.8 \times (1.8 \times 1.8 - (-0.3) \times (-0.3)) \\ &= 1.8 \times (3.24 - 0.09) = 1.8 \times 3.15 = 5.67 \end{aligned}$$

The inverse is:

$$\mathbf{\Sigma}_4^{-1} = \begin{bmatrix} 0.571 & 0.095 & 0 \\ 0.095 & 0.571 & 0 \\ 0 & 0 & 0.556 \end{bmatrix}$$

Figure 1.3.1.3 – Computing Determinants & Inverses



For numerical stability, we use the log-likelihood formulation. For each class j, the log-likelihood is:

$$\log p(\mathbf{x}|L=j) = -\frac{3}{2}\log(2\pi) - \frac{1}{2}\log|\mathbf{\Sigma}_j| - \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_j)^T\mathbf{\Sigma}_j^{-1}(\mathbf{x}-\boldsymbol{\mu}_j)$$

The constant term $-\frac{3}{2}\log(2\pi) \approx -2.756$ is the same for all classes and can be ignored in comparison. The discriminant function for each class becomes:

$$g_j(\mathbf{x}) = -\frac{1}{2}\log|\mathbf{\Sigma}_j| - \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_j)^T\mathbf{\Sigma}_j^{-1}(\mathbf{x}-\boldsymbol{\mu}_j)$$

Substituting our computed values:

$$\begin{aligned} g_1(\mathbf{x}) &= -\frac{1}{2}\log(3.375) - \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_1)^T\mathbf{\Sigma}_1^{-1}(\mathbf{x}-\boldsymbol{\mu}_1) \\ &= -0.608 - \frac{1}{2}\mathbf{x}^T\begin{bmatrix}0.667 & 0 & 0\\ 0 & 0.667 & 0\\ 0 & 0 & 0.667\end{bmatrix}\mathbf{x} \\ &= -0.608 - \frac{0.667}{2}(x_1^2+x_2^2+x_3^2) \end{aligned}$$

$$g_2(\mathbf{x}) = -0.608 - \frac{0.667}{2}\left[(x_1-2.5)^2+(x_2-2.5)^2+x_3^2\right]$$

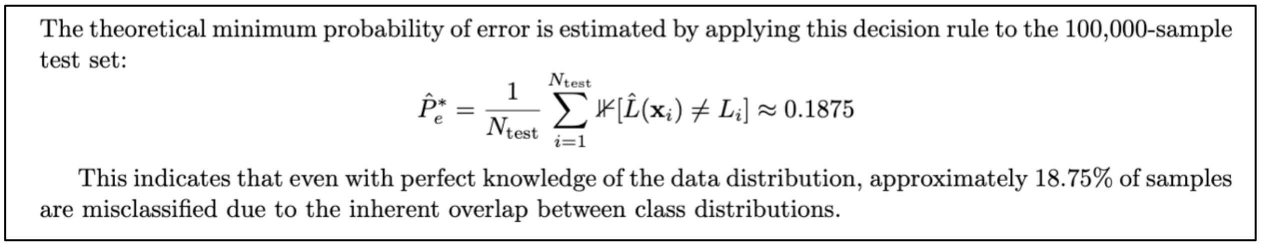
$$\begin{aligned} g_3(\mathbf{x}) &= -\frac{1}{2}\log(5.4) - \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_3)^T\mathbf{\Sigma}_3^{-1}(\mathbf{x}-\boldsymbol{\mu}_3) \\ &= -0.843 - \frac{1}{2}\begin{bmatrix}x_1\\ x_2-2.5\\ x_3-2.5\end{bmatrix}^T\begin{bmatrix}0.625 & -0.125 & -0.125\\ -0.125 & 0.625 & -0.125\\ -0.125 & -0.125 & 0.625\end{bmatrix}\begin{bmatrix}x_1\\ x_2-2.5\\ x_3-2.5\end{bmatrix} \end{aligned}$$

$$\begin{aligned} g_4(\mathbf{x}) &= -\frac{1}{2}\log(5.67) - \frac{1}{2}\begin{bmatrix}x_1-2.5\\ x_2\\ x_3-2.5\end{bmatrix}^T\begin{bmatrix}0.571 & 0.095 & 0\\ 0.095 & 0.571 & 0\\ 0 & 0 & 0.556\end{bmatrix}\begin{bmatrix}x_1-2.5\\ x_2\\ x_3-2.5\end{bmatrix} \\ &= -0.868 - \frac{1}{2}\left[0.571(x_1-2.5)^2 + 0.571x_2^2 + 0.556(x_3-2.5)^2 \right. \\ &\quad \left. +2(0.095)(x_1-2.5)x_2\right] \end{aligned}$$

The decision rule is:

$$\hat{L}(\mathbf{x}) = \arg\max_{j\in\{1,2,3,4\}} g_j(\mathbf{x})$$

Figure 1.3.1.4 – Log-Likelihood Decision Rule



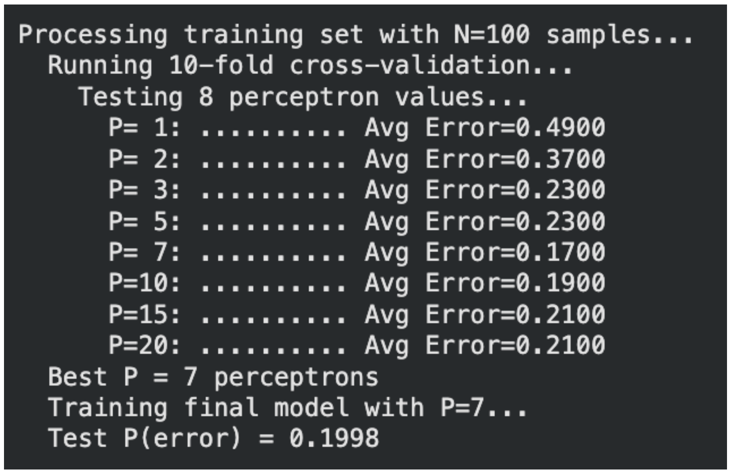
The theoretical minimum probability of error is estimated by applying this decision rule to the 100,000-sample test set:

$$\hat{P}_e^* = \frac{1}{N_{\text{test}}}\sum_{i=1}^{N_{\text{test}}} \nVdash[\hat{L}(\mathbf{x}_i) \neq L_i] \approx 0.1875$$

This indicates that even with perfect knowledge of the data distribution, approximately 18.75% of samples are misclassified due to the inherent overlap between class distributions.

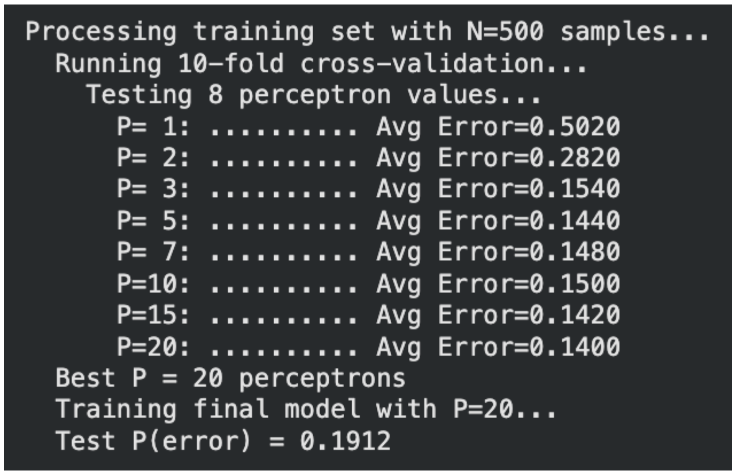
Figure 1.3.1.5 – Empirical Error Rate

1.3.2. Screenshots



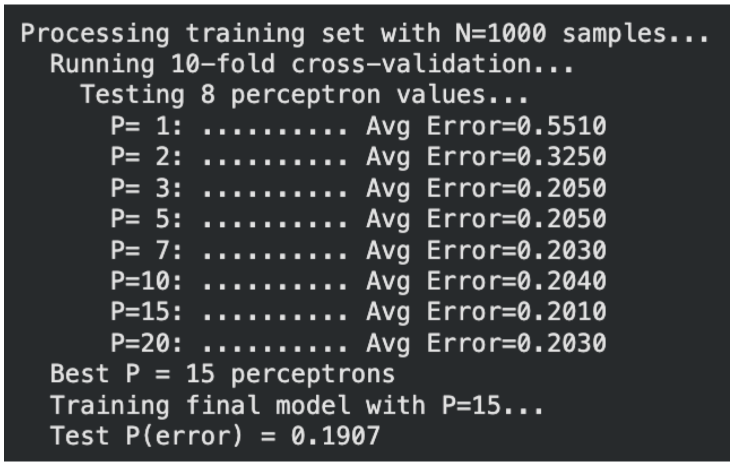
```
Processing training set with N=100 samples...
  Running 10-fold cross-validation...
    Testing 8 perceptron values...
      P= 1: .......... Avg Error=0.4900
      P= 2: .......... Avg Error=0.3700
      P= 3: .......... Avg Error=0.2300
      P= 5: .......... Avg Error=0.2300
      P= 7: .......... Avg Error=0.1700
      P=10: .......... Avg Error=0.1900
      P=15: .......... Avg Error=0.2100
      P=20: .......... Avg Error=0.2100
  Best P = 7 perceptrons
  Training final model with P=7...
  Test P(error) = 0.1998
```

Figure 1.3.2.1 – Cross-validation results for N=100 training samples.



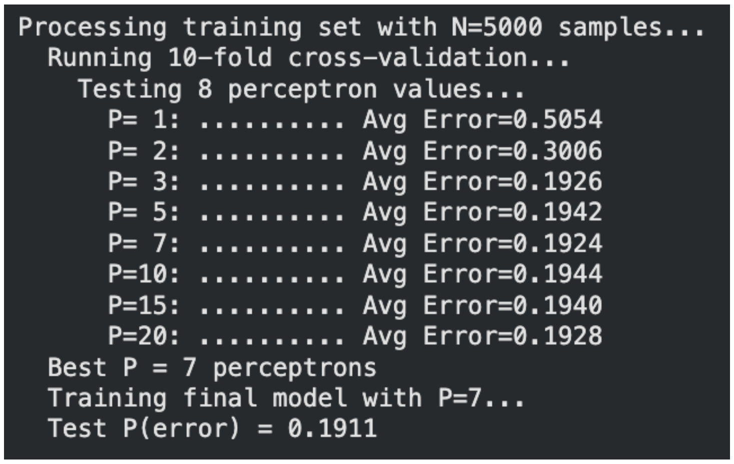
```
Processing training set with N=500 samples...
  Running 10-fold cross-validation...
    Testing 8 perceptron values...
      P= 1: .......... Avg Error=0.5020
      P= 2: .......... Avg Error=0.2820
      P= 3: .......... Avg Error=0.1540
      P= 5: .......... Avg Error=0.1440
      P= 7: .......... Avg Error=0.1480
      P=10: .......... Avg Error=0.1500
      P=15: .......... Avg Error=0.1420
      P=20: .......... Avg Error=0.1400
  Best P = 20 perceptrons
  Training final model with P=20...
  Test P(error) = 0.1912
```

Figure 1.3.2.2 – Cross-validation results for N=500 training samples.



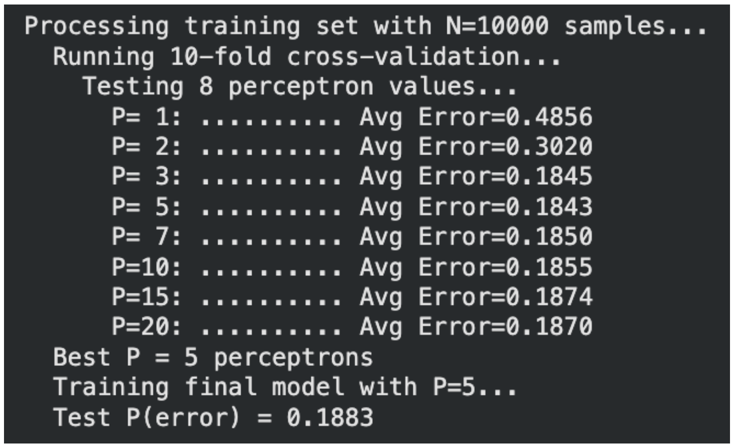
```
Processing training set with N=1000 samples...
  Running 10-fold cross-validation...
    Testing 8 perceptron values...
      P= 1: .......... Avg Error=0.5510
      P= 2: .......... Avg Error=0.3250
      P= 3: .......... Avg Error=0.2050
      P= 5: .......... Avg Error=0.2050
      P= 7: .......... Avg Error=0.2030
      P=10: .......... Avg Error=0.2040
      P=15: .......... Avg Error=0.2010
      P=20: .......... Avg Error=0.2030
  Best P = 15 perceptrons
  Training final model with P=15...
  Test P(error) = 0.1907
```

Figure 1.3.2.3 – Cross-validation results for N=1,000 training samples.



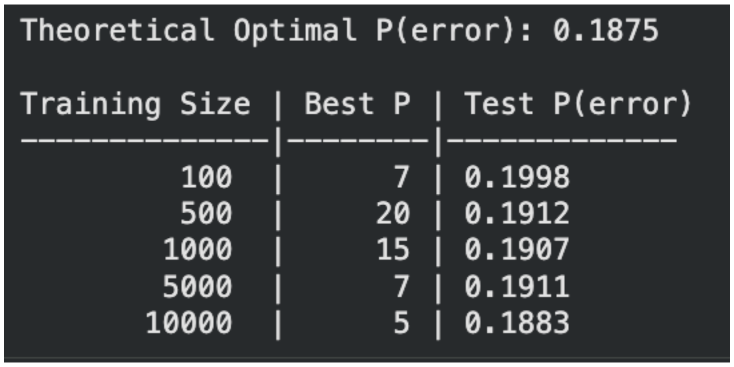
```
Processing training set with N=5000 samples...
  Running 10-fold cross-validation...
    Testing 8 perceptron values...
      P= 1: .......... Avg Error=0.5054
      P= 2: .......... Avg Error=0.3006
      P= 3: .......... Avg Error=0.1926
      P= 5: .......... Avg Error=0.1942
      P= 7: .......... Avg Error=0.1924
      P=10: .......... Avg Error=0.1944
      P=15: .......... Avg Error=0.1940
      P=20: .......... Avg Error=0.1928
  Best P = 7 perceptrons
  Training final model with P=7...
  Test P(error) = 0.1911
```

Figure 1.3.2.4 – Cross-validation results for N=5,000 training samples.



```
Processing training set with N=10000 samples...
  Running 10-fold cross-validation...
    Testing 8 perceptron values...
      P= 1: .......... Avg Error=0.4856
      P= 2: .......... Avg Error=0.3020
      P= 3: .......... Avg Error=0.1845
      P= 5: .......... Avg Error=0.1843
      P= 7: .......... Avg Error=0.1850
      P=10: .......... Avg Error=0.1855
      P=15: .......... Avg Error=0.1874
      P=20: .......... Avg Error=0.1870
  Best P = 5 perceptrons
  Training final model with P=5...
  Test P(error) = 0.1883
```

Figure 1.3.2.5 – Cross-validation results for N=10,000 training samples.



```
Theoretical Optimal P(error): 0.1875

Training Size | Best P | Test P(error)
--------------|--------|-------------
      100     |      7 | 0.1998
      500     |     20 | 0.1912
     1000     |     15 | 0.1907
     5000     |      7 | 0.1911
    10000     |      5 | 0.1883
```

Figure 1.3.2.6 – Summary of cross-validation results and test performance.

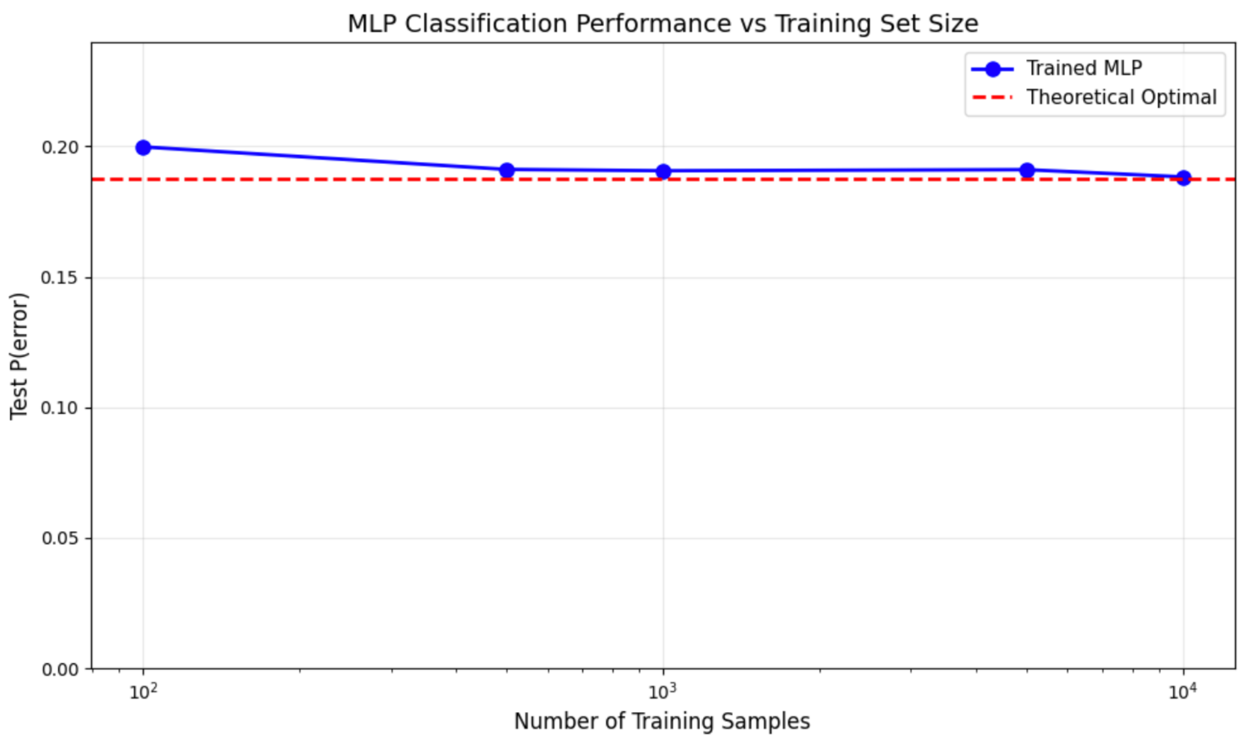


Figure 1.3.2.7 – MLP test performance versus training set size.

1.4. Analysis

Several interesting patterns emerge from the experimental results shown in Figure 1.3.2.7 and summarized in Figure 1.3.2.6. The trained MLPs consistently approached but did not quite reach the theoretical optimum performance. With just 100 training samples, the test error was approximately 1.23 percentage points above the theoretical optimal of 18.75%. As the training data increased to 10,000 samples, this gap narrowed dramatically to only 0.08 percentage points, with the MLP achieving 18.83% error compared to the theoretical 18.75%. This outcome is expected, as the theoretical optimum represents the performance of a classifier with perfect knowledge of the true data distribution, while the MLP must estimate this distribution from finite training data. The fact that the MLP approaches within 0.08 percentage points demonstrates that with sufficient data, neural networks can effectively approximate the optimal classifier despite this fundamental limitation.

The cross-validation process selected varying numbers of perceptrons across different training set sizes. Interestingly, larger datasets did not necessarily require more complex models. The smallest architecture (5 perceptrons) was selected for the largest dataset (N=10,000), while an intermediate dataset (N=500) selected the most complex architecture (20 perceptrons). This pattern reflects the bias-variance tradeoff: with limited data, cross-validation may favor slightly more complex models that happen to perform well on the validation folds, while abundant data allows simpler models to generalize effectively without overfitting.

The most significant performance improvement occurred between 100 and 1,000 training samples, where the test error dropped from 19.98% to 19.07%. Beyond 1,000 samples, the rate of improvement diminished. Moving from 1,000 to 10,000 samples only reduced the error by approximately 0.24 percentage points. This diminishing return suggests that for this particular classification problem, a moderately sized training set of around 1,000-2,000 samples would be sufficient to achieve near-optimal performance.

2. Question #2

2.1. Background

This problem investigates how cross-validation-based model order selection performs when applied to Gaussian Mixture Models (GMMs) with varying amounts of training data. The goal is to determine whether cross-validation can reliably identify the true number of components when the true data distribution is known, and how this capability changes with dataset size.

2.2. Methodology

To generate the data, a 4-component GMM was used as the true underlying distribution. The component means were positioned to create overlap between two of the components: Component 1 at [0, 0], Component 2 at [2, 2], Component 3 at [6, 0], and Component 4 at [0, 6]. Components 1 and 2 were given covariance matrices with small positive and negative correlations respectively, while Components 3 and 4 had simpler diagonal structures. All components had equal mixing weights of 25%, creating a balanced distribution where the overlap between Components 1 and 2 makes the model order selection more challenging.

Training datasets of three different sizes were generated: 10, 100, and 1000 samples. For each dataset size, the experiment was repeated 100 times to gather information about model selection behavior. This repeated trial approach allows for understanding not just which model is selected most often, but also how reliable the selection process is for different data sizes.

For each generated dataset, 10-fold cross-validation was performed to evaluate GMM candidates with 1 through 10 components. The cross-validation process used the Expectation-Maximization (EM) algorithm for parameter estimation. For each candidate number of components K, the training data was split into 10 folds, and a GMM with K components was fit to 9 folds and evaluated on the remaining fold using log-likelihood as the metric. This process was repeated for all 10 folds, and the log-likelihoods were averaged. The number of components achieving the highest average log-likelihood was selected as the best model for that dataset.

To ensure robust parameter estimation, each GMM was trained with multiple random initializations (n_init=10) and up to 200 EM iterations. For the smallest dataset size (N=10), the number of cross-validation folds was automatically reduced to 2 to ensure sufficient samples per fold for model fitting. The cross-validation procedure included a safety check that skipped any fold where the number of training samples was less than the number of components being tested. For N=10, this meant that models with K=6-10 components had no valid folds (since each training split contained only 5 samples), resulting in the negative infinity log-likelihoods shown in Table 2.3.1.

2.3. Results

K	Avg CV Log-Likelihood			Selection Rate (%)		
	N=10	N=100	N=1000	N=10	N=100	N=1000
1	-8.41	-4.75	-4.73	100.0	0.0	0.0
2	-10254.19	-4.48	-4.41	0.0	0.0	0.0
3	-983691.81	-4.35	-4.23	0.0	87.0	2.0
4*	**-1826724.20**	**-4.41**	**-4.22**	**0.0**	**13.0**	**87.0**
5	-2591981.19	-4.53	-4.22	0.0	0.0	10.0
6	$-\infty$	-4.63	-4.23	0.0	0.0	0.0
7	$-\infty$	-4.76	-4.23	0.0	0.0	0.0
8	$-\infty$	-4.89	-4.24	0.0	0.0	1.0
9	$-\infty$	-5.03	-4.24	0.0	0.0	0.0
10	$-\infty$	-5.22	-4.24	0.0	0.0	0.0

* True number of components (K=4). Based on 100 trials per dataset size.

Table 2.3.1 – Model Order Selection Results

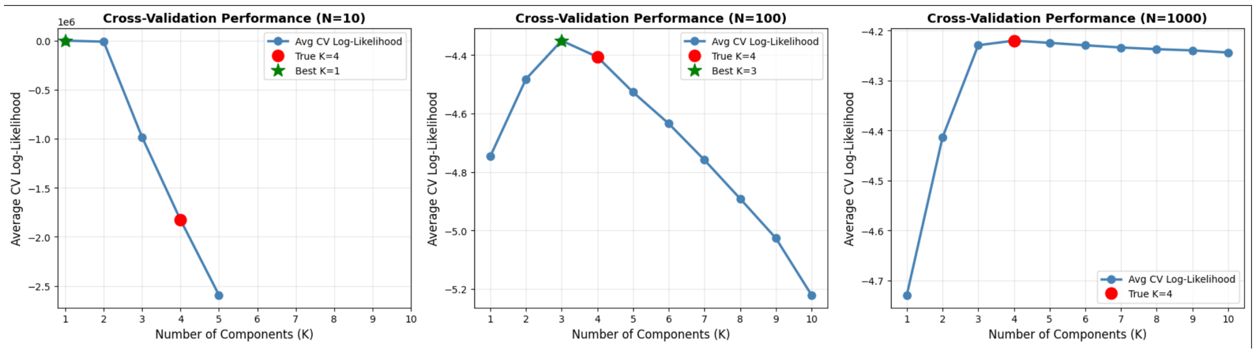


Figure 2.3.1 – Cross-validation performance

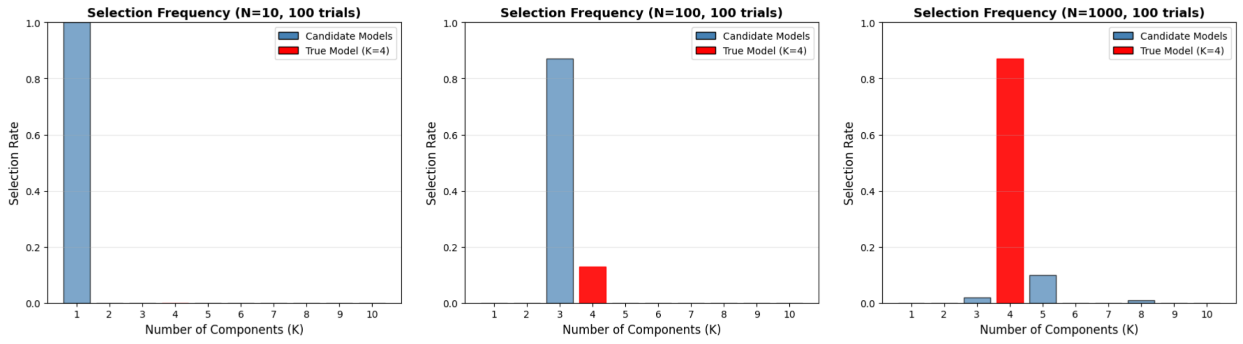


Figure 2.3.2 – Model Frequency Selection

2.4. Analysis

The results demonstrate a clear relationship between dataset size and the reliability of model order selection. With only 10 samples, cross-validation consistently underestimates the true complexity, selecting the simplest model (K=1) in every trial. This outcome makes sense because with such limited data, complex models cannot be reliably estimated. The extremely negative log-likelihoods for K=2 through K=5 suggest severe overfitting, where the models fit the small training folds too closely and perform poorly on validation.

With 100 samples, cross-validation shows substantial improvement but still exhibits a slight bias toward underfitting. The selection of K=3 in 87% of trials, despite K=4 being the true value, reflects the challenge of distinguishing between overlapping components with limited data. The very similar log-likelihoods for K=3 (-4.35) and K=4 (-4.41) indicate that both models explain the data almost equally well at this sample size, with cross-validation showing a slight preference for the simpler model to avoid overfitting.

The dramatic improvement with 1000 samples demonstrates that cross-validation becomes highly reliable when sufficient data is available. The correct selection of K=4 in 87% of trials shows that the true model structure can be identified accurately. The occasional selection of K=5 (10% of trials) with essentially identical performance (-4.22 for both K=4 and K=5) suggests that at this sample size, the model has enough flexibility that adding one extra component does not dramatically hurt generalization, though it also does not provide significant improvement.

The cross-validation performance curves reveal why certain models are selected. For N=10, the curve shows a sharp drop off after K=2, indicating that more complex models catastrophically overfit the tiny training set. For N=100 and N=1000, the curves become smoother, with K=3 and K=4 emerging as clear optima. The nearly identical log-likelihoods for K=3 and K=4 at N=100 explain the mixed selection results, while the clearer optimum at K=4 for N=1000 explains its more reliable selection.

This experiment successfully demonstrates the bias-variance tradeoff in model selection. With insufficient data, cross-validation errs on the side of simplicity to avoid overfitting, even when this means underestimating the true complexity. As data increases, cross-validation can distinguish finer differences between models and reliably identify the true structure. The 87% selection rate for K=4 with N=1000 shows that cross-validation is a powerful and reliable tool when adequate data is available, though some inherent uncertainty remains when multiple model orders perform similarly. The deliberate overlap between Components 1 and 2 in the true GMM successfully created a challenging scenario where model selection requires sufficient data to distinguish the underlying structure.

3. Code Repository Link

- https://github.com/yalongwastaken/yalong_eece5644_hw3

4. Citations

4.1. External Assistance:

- Claude AI (Anthropic) was consulted for debugging assistance, implementation guidance, verification of theoretical concepts, document organization, and writing clarity. All code, derivations, analyses, and results are my original work.