Anush Sriram Ramesh
Machine Learning and Pattern Recognition EECE 5644 V30
HW 4

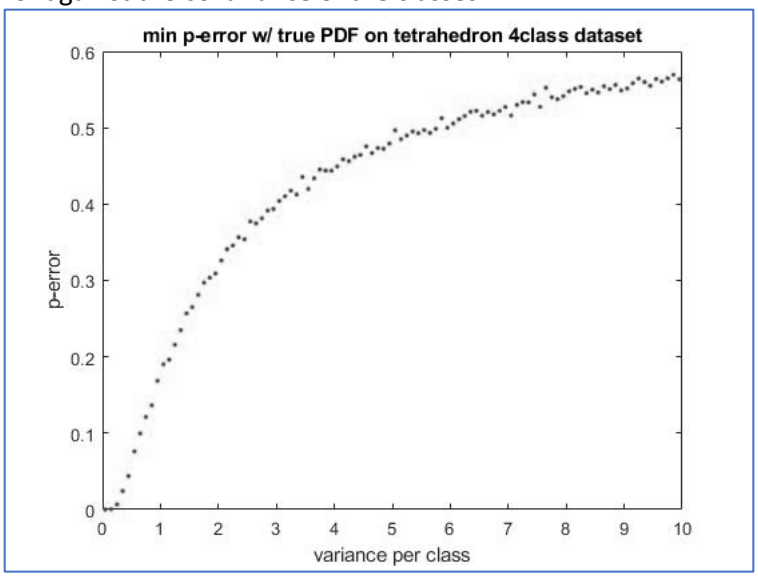
GitHub Link **– Go into folder HW4 for code and results**

## Question 1.

### 1.1 Gaussian Mixture

A gaussian mixture was selected, with uniform prior, such that given the true pdf the min p-error is between 0.1 and 0.2. The gaussian mm given in HW2 was with a minimum p-error of 0.17 approx., so that was chosen.

Means at the vertices of the unit vectors along the axis is chosen (1,1,1), (1,-1,-1), (-1,-1,1), (-1,1,-1). The covariance is a diag(Cov) [a diagonal matrix for some value of Cov]. Ideal covariance was chosen by plotting the min p-error against the covariance of the classes.




A cov value of approximately between 0.8 to 1.9 should have the min p-error between 0.1 and 0.2.

Thus, a cov value of 1 was chosen. $\begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix}$ was chosen as the covariance matrix.

Training sets are generated for sizes 100,200,500,1000,2000 and 5000. Test set of 100000 was created. These are stored as csv files in HW4.

### 1.2 Structure of Network and Software Packages

ELU activation is used as the activation function for the perceptron's instead of RELU due to its smooth nature. SoftMax activation function is also used in this context, the ideal value of k in the SoftMax function is chosen through k fold cross validation.

Categorical cross entropy loss function was chosen, which is an extension of binary cross entropy loss function described in class multiple class labels.

All code was implemented with TensorFlow which uses keras as a backend. Adam optimizer, which changes the step size dynamically, is used with **keras.losses.CategoricalCrossentropy()** loss function defined in keras.

Within keras, we create a network with dense layer connections, and activation function ELU and a dense 4-layer activation SoftMax is created.

### 1.3 Results and Observations

**Minimum P-Error over data samples**

The performance of the model is upper bounded by the theoretical optimal classifier. But with more datapoints the model can predict the pdf better.

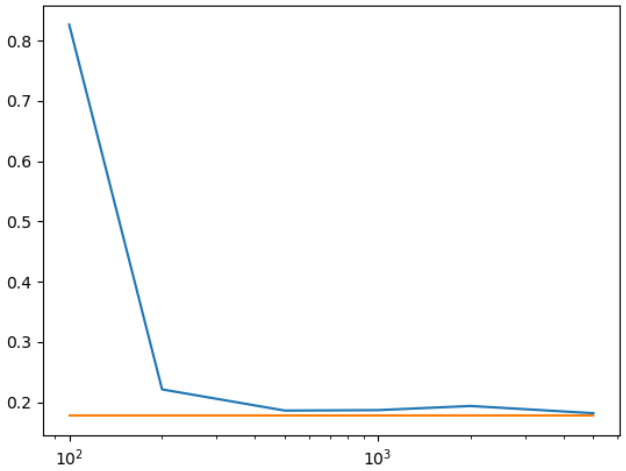



**Fig: Blue line is the performance of the model plotted against the number of data points.**

It Is clearly seen that 100 samples are not anywhere near sufficient but adding more samples helps the model greatly, where the model almost accurately predicts the pdf at around 1000 samples.

**Rate Convergence wrt different datasets**

Another key observation is that the number of perceptron varies inversely with the number of data points. The theoretical optimum is met by 18 perceptron when the size of the dataset is 500. But as the size increases the theoretical optimum is met by 8 perceptron's itself.

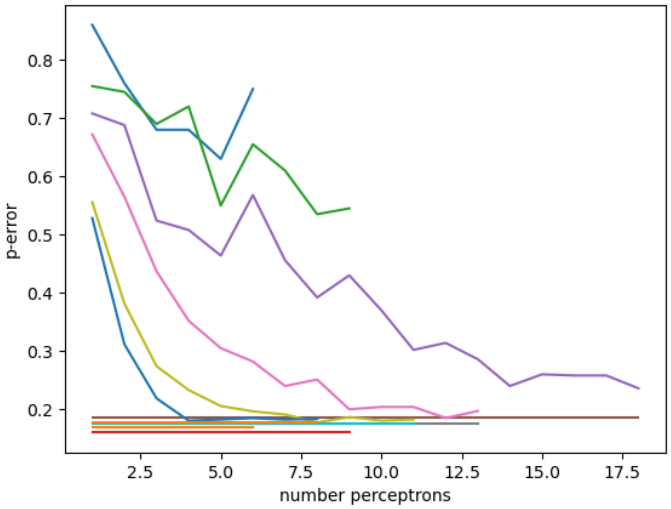



**Fig: Blue 100 samples, Green 200, Purple 500, Pink 1000, Yellow 2000 and so on.**

**Question 2.**

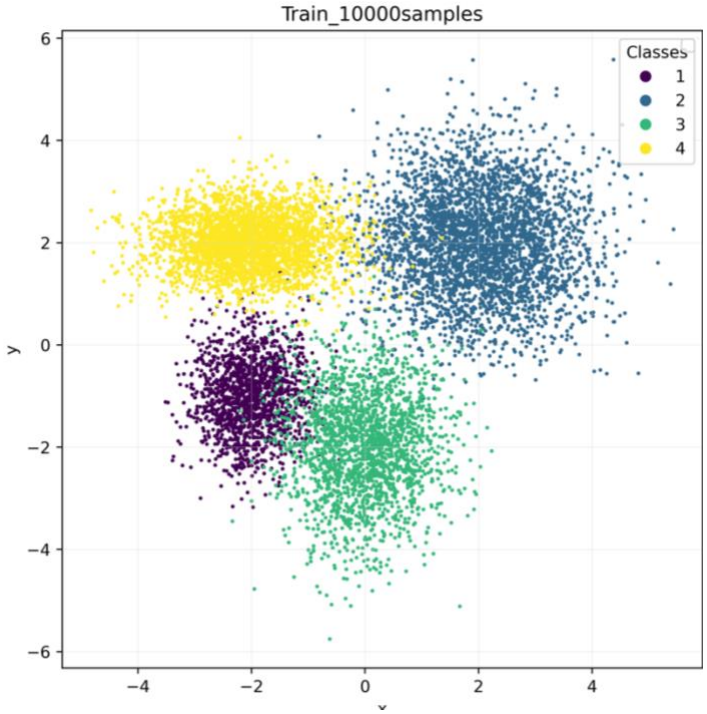
**2.1. Software Packages**

We use the Scikit Learns' Gaussian Mixture class to make use of the EM algorithm to fit gaussians to the data. The fit and score sample's function was used to train the model to fit the data and to calculate the log-likelihood.

**2.2 Part 1. Distribution Selection**

We are asked to select a GMM with 4 components.
The priors chosen for the components are 0.15,0.25,0.3,0.2.
The mean vectors are(−2,−1),(2,2),(0,−2),and(−2,2)
The covariances are cov(0.25,0.5), cov(1,1), cov(0.5,1), and cov(0.75,0.25), where cov(a,b) is

$$cov(a,b) = \begin{pmatrix} a & 0 \\ 0 & b \end{pmatrix}$$




Here the measure of how well the model performs is chosen as the log likelihood on the validation data, as its probability of error does not suffice. The model that best approximates the data will maximize this; we take the negative of that to make this a minimization problem.

**2.3 Part 3**

The matrix below captures the models selected during each of the 30 runs. Each row corresponds to dataset size [10, 100, 1000, 10000], with each column consisting of the number of components in the model [1,2,3,4,5,6].

**[[30. 0. 0. 0. 0. 0.]**
**[ 0. 0. 13. 13. 4. 0.]**
**[ 0. 0. 0. 13. 12. 5.]**

**[ 0.  0.  0. 20. 10.  0.]]**

Ideally the model with 4 components must be selected always, but due to the data sparsity we see a lot of confusion. Like in the case for 10 samples, 10-fold cross validation is pointless as there is only one point in each validation set. Thus, consistently model with 1 component being selected.

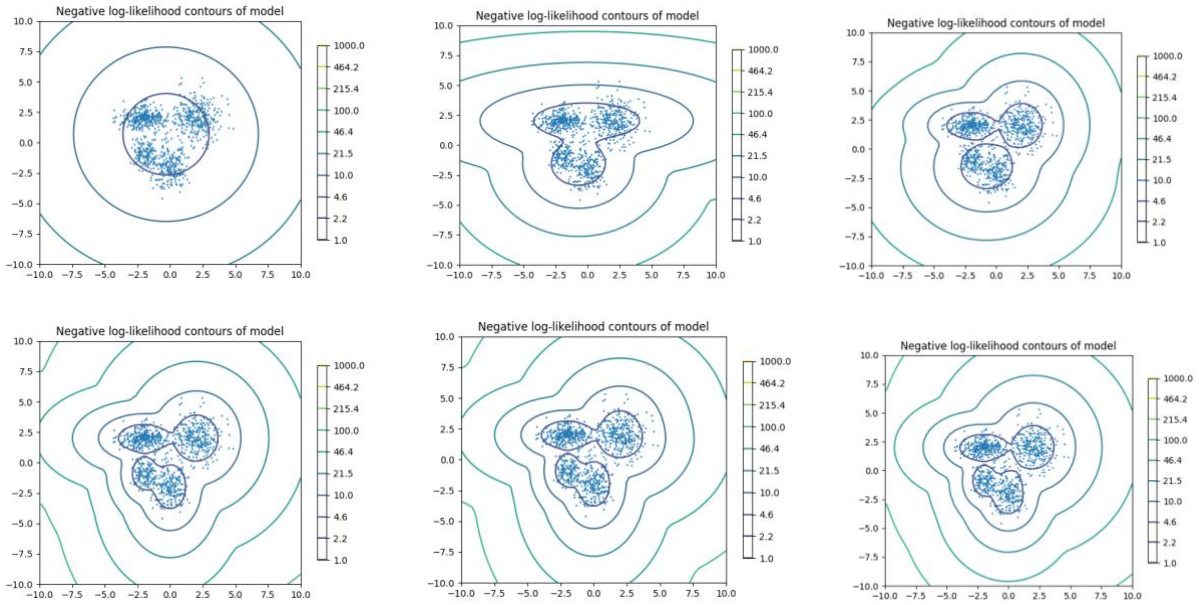



**Fig: GMM Model fitted to training data with 1000 samples**

Images of Models fitted to all other datasets of sizes 10, 100 and 10K are available in git repo. They're not demonstrated here, as the functionality is just repetitive and don't add any value.

### 2.4 Part 4 Probability of Correct Selection

In the 10 and 100 datasets, the tends to be pessimistically chosen, favoring errors of smaller model order overcompensating, like model 4 was decided most of the time in 100 sample datasets. This can be seen from the fact that a lower order model was selected 10/30 times whereas higher order model was selected only 2 times. But, as the size of dataset increases it never reports lower order models incorrectly.

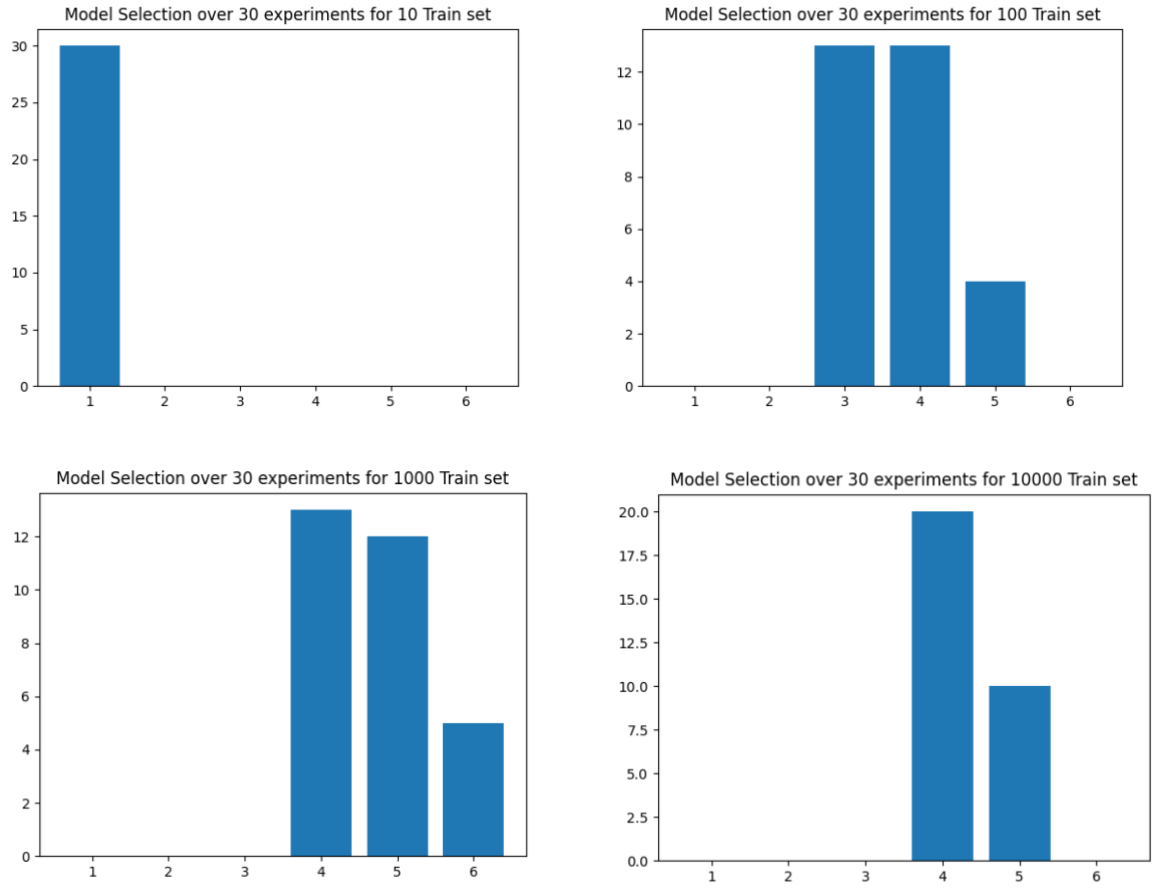



**Fig: Bar chart for model selected**