EECE5644 - Assignment 2

Dhruv Agarwal

October 2024

1 Question 1

1.1 Data Generation

x is a 3-dimenstional random vector which is generated using 4 Gaussian models with uniform priors, with each class having its own means and variances.

$$P_0 = 0.25, P_1 = 0.25, P_2 = 0.25, P_3 = 0.25$$

$$\mu_0 = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} \mu_1 = \begin{bmatrix} 3 & 3 & 3 \end{bmatrix} \mu_2 = \begin{bmatrix} 6 & 6 & 6 \end{bmatrix} \mu_3 = \begin{bmatrix} 9 & 9 & 9 \end{bmatrix}$$

$$\Sigma_0 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \Sigma_1 = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix} \Sigma_2 = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 3 \end{bmatrix} \Sigma_3 = \begin{bmatrix} 4 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 4 \end{bmatrix}$$

For each sample, the label is chosen as L_i with probability P_i
For label L_i:

$$x \sim \mathcal{N}(\mu_i, \Sigma_i^2)$$

1.2 MAP Classifier

$$P(L_0) = 0.25, \quad P(L_1) = 0.25, \quad P(L_2) = 0.25, \quad P(L_3) = 0.25$$

$$P(x|L_i) = \frac{1}{(2\pi)^{3/2}|\Sigma_i|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_i)^T \Sigma_i^{-1}(x-\mu_i)\right)$$

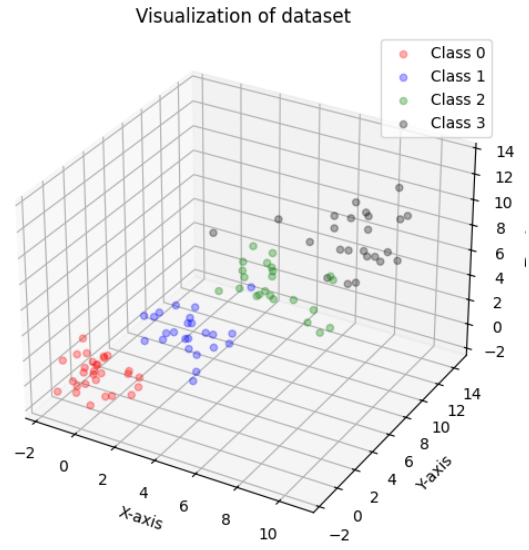
$$\text{post}(L_i|x) = P(L_i) \cdot P(x|L_i)$$

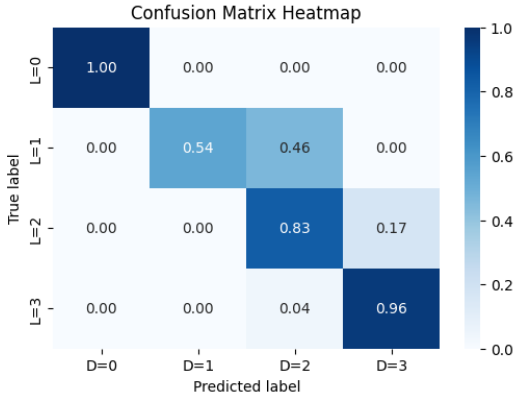
$$\hat{y} = \arg\max_i \text{post}(L_i|x)$$

$$\hat{y} = \arg\max_i P(L_i) \cdot P(x|L_i)$$

1.3 Result for data generation and restricting MAP classification results to 10-20% error

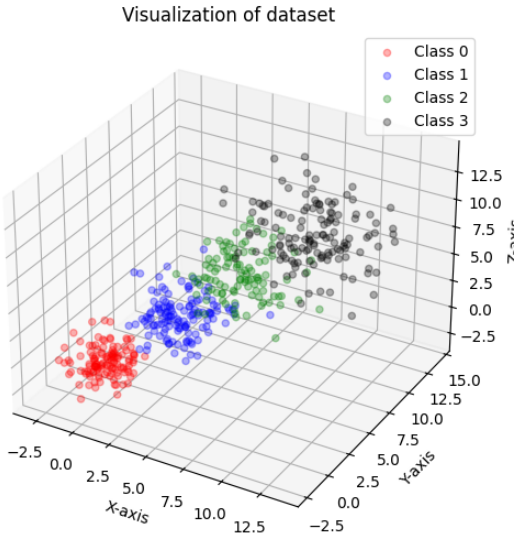
- 100 samples training data

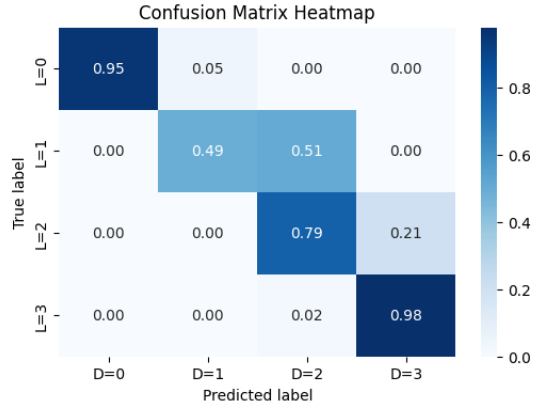





```
Accuracy: 0.84
Minimum probability error: 0.16
```

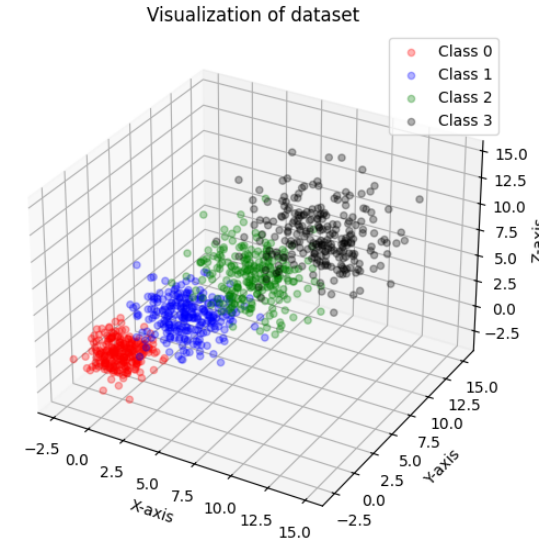
- 500 samples training data

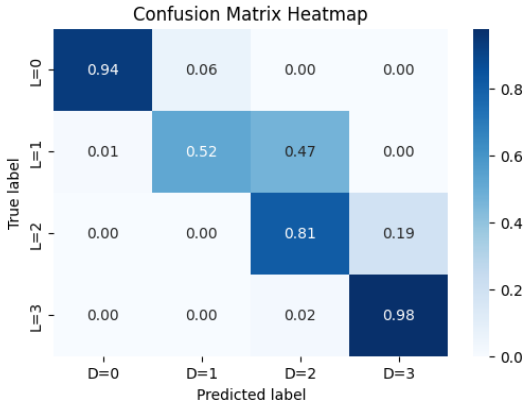





```
Accuracy: 0.80
Minimum probability error: 0.20
```

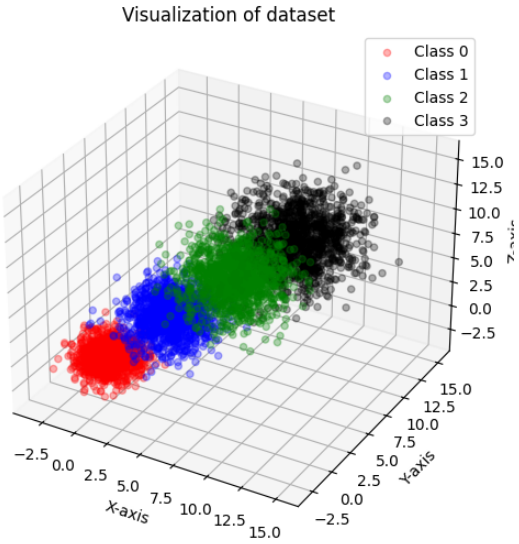
- 1000 samples training data

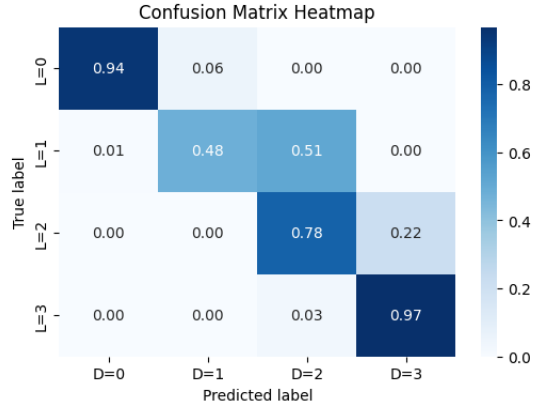





```
Accuracy: 0.81
Minimum probability error: 0.19
```

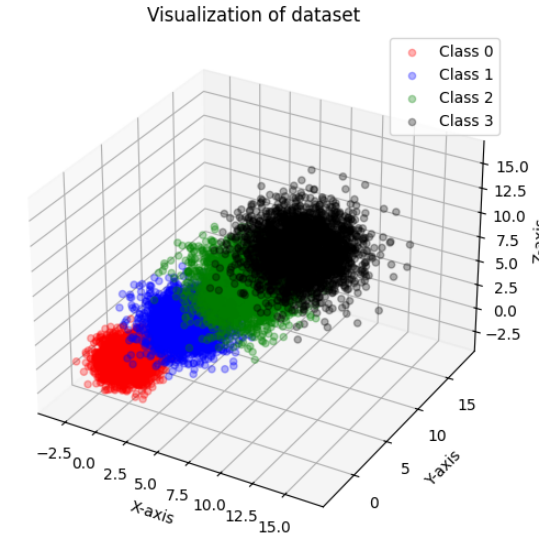
- 5000 samples training data

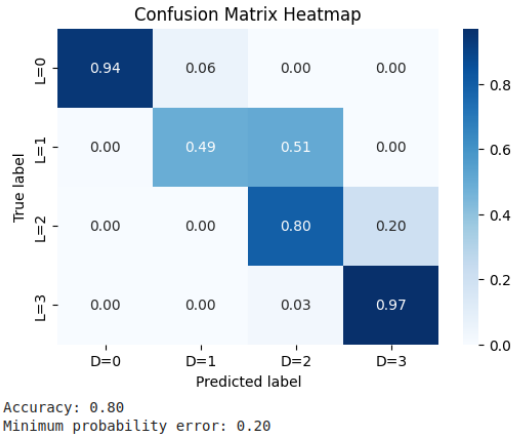





```
Accuracy: 0.79
Minimum probability error: 0.21
```

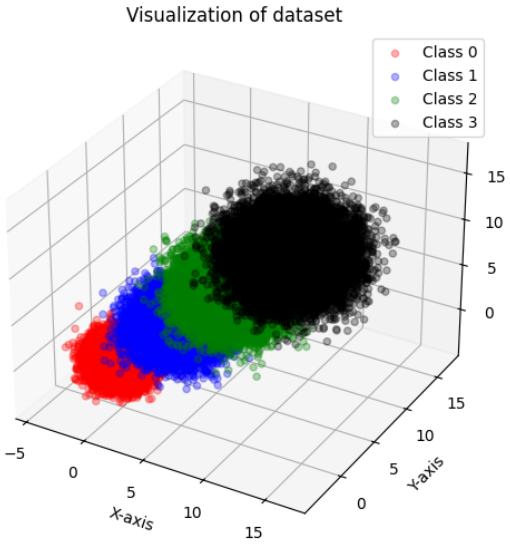
- 10000 samples training data

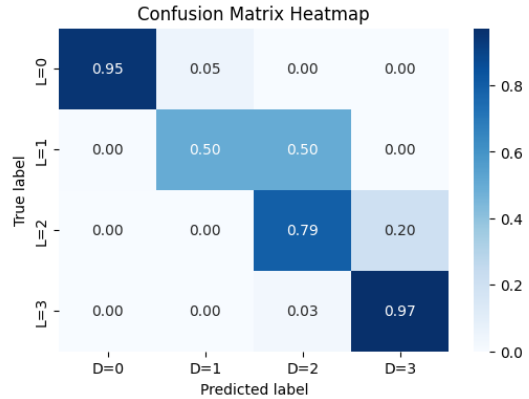





```
Accuracy: 0.80
Minimum probability error: 0.20
```

- 100000 samples validation data






```
Accuracy: 0.80
Minimum probability error: 0.20
```

1.4 MLP Model

$$\mathbf{y} = \mathbf{W}_2\,\sigma\left(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1\right) + \mathbf{b}_2$$

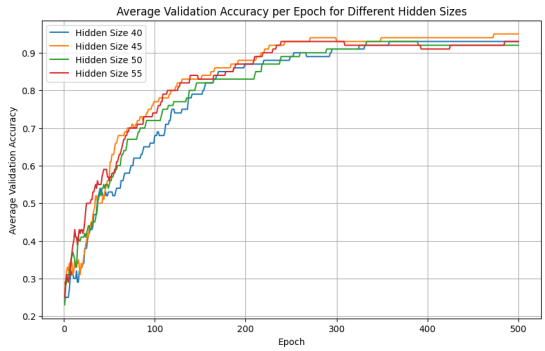
$$\sigma(\mathbf{h}) = \begin{cases} \mathrm{ISRU}(\mathbf{h}) = \frac{\mathbf{h}}{\sqrt{1+\alpha\mathbf{h}^2}} & \text{ISRU} \\ \mathrm{ELU}(\mathbf{h}) = \begin{cases} \mathbf{h}, & \mathbf{h} \geq 0 \\ \alpha(e^{\mathbf{h}} - 1), & \mathbf{h} < 0 \end{cases} & \text{ELU} \\ \text{Smooth ReLU}(\mathbf{h}) = \log(1 + e^{\mathbf{h}}) & \text{Smooth ReLU} \end{cases}$$

$$\text{Loss} = -\frac{1}{N}\sum_{i=1}^{N}\log\left(\frac{e^{\hat{y}_i}}{\sum_j e^{\hat{y}_j}}\right)$$

1.5 Result for accuracy with different number of neurons in the MLP using different activation functions

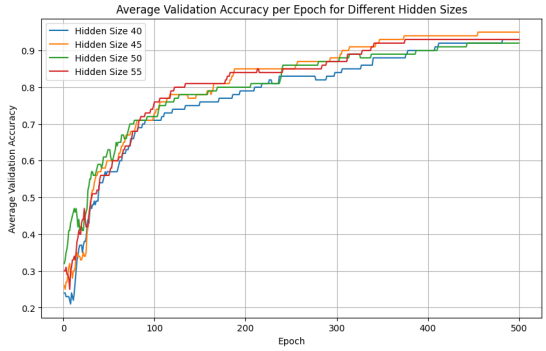
- 100 training sample

Activation Function: ISRU



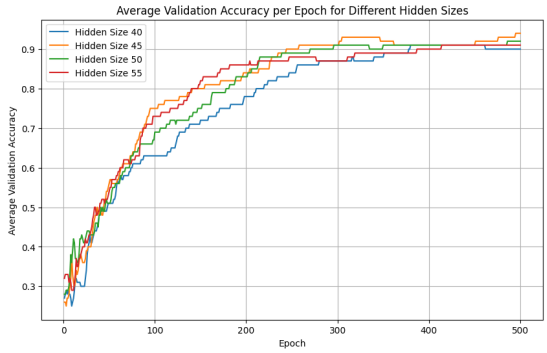
Best hidden size: 45, Best epoch: 473, Best accuracy: 0.9500

Activation Function: Smooth ReLU



Best hidden size: 45, Best epoch: 455, Best accuracy: 0.9500

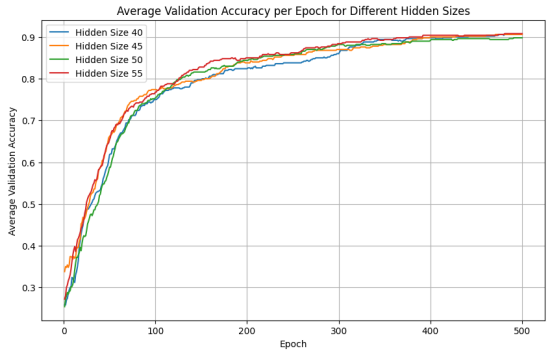
Activation Function: ELU



Best hidden size: 45, Best epoch: 495, Best accuracy: 0.9400

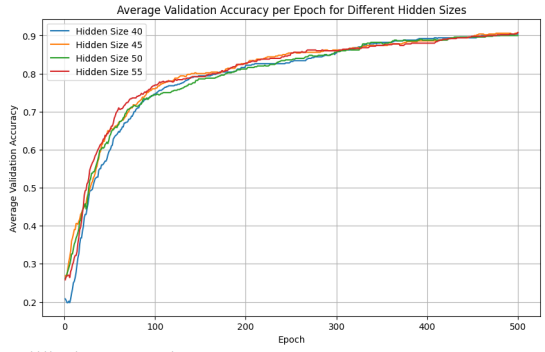
- 500 training sample

Activation Function: ISRU



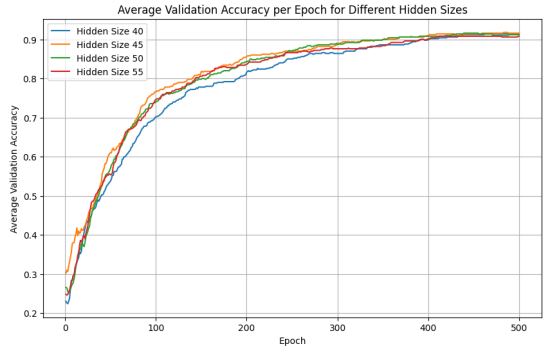
Best hidden size: 55, Best epoch: 481, Best accuracy: 0.9080

Activation Function: Smooth ReLU



Best hidden size: 55, Best epoch: 500, Best accuracy: 0.9080

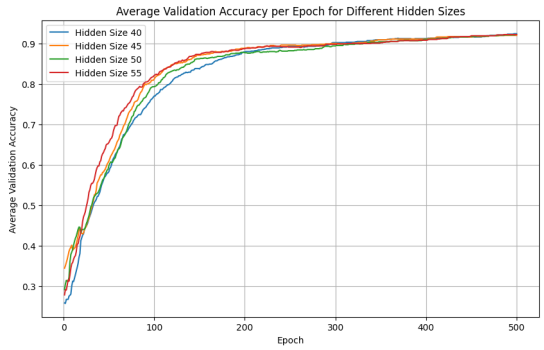
```
Activation Function: ELU
```




```
Best hidden size: 45, Best epoch: 483, Best accuracy: 0.9180
```

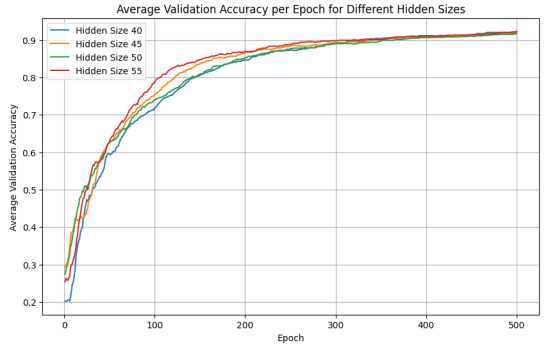
- 1000 training sample

```
Activation Function: ISRU
```



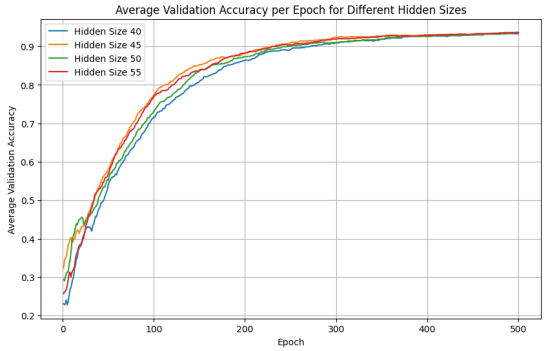

```
Best hidden size: 40, Best epoch: 497, Best accuracy: 0.9240
```

```
Activation Function: Smooth ReLU
```




```
Best hidden size: 55, Best epoch: 499, Best accuracy: 0.9220
```

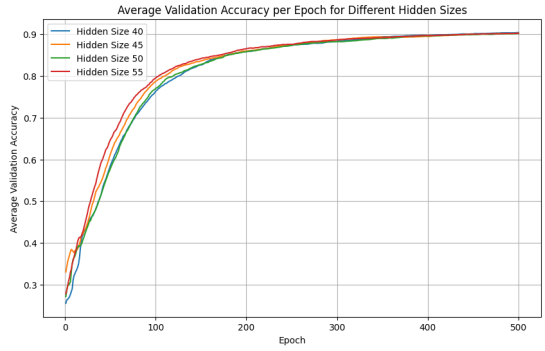
```
Activation Function: ELU
```




```
Best hidden size: 45, Best epoch: 497, Best accuracy: 0.9360
```

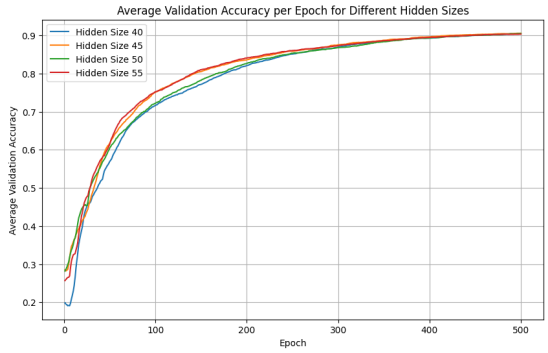
- 5000 training sample

```
Activation Function: ISRU
```



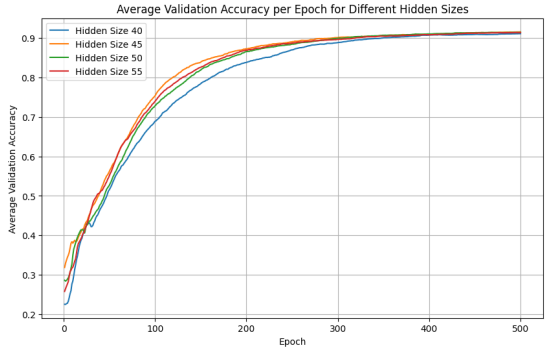

```
Best hidden size: 40, Best epoch: 498, Best accuracy: 0.9038
```

```
Activation Function: Smooth ReLU
```




```
Best hidden size: 45, Best epoch: 500, Best accuracy: 0.9058
```

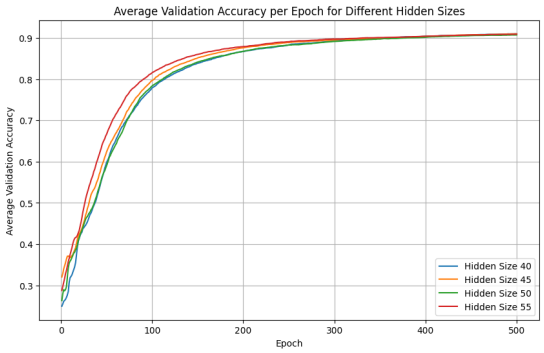
```
Activation Function: ELU
```




```
Best hidden size: 45, Best epoch: 499, Best accuracy: 0.9156
```

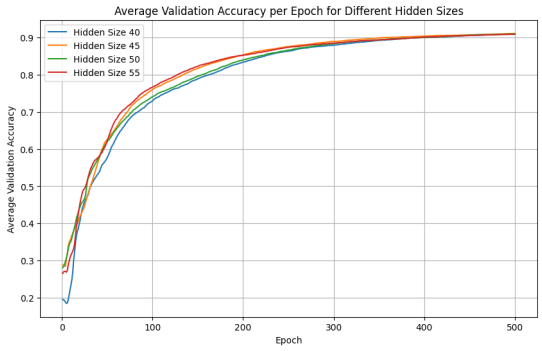
- 10000 training sample

```
Activation Function: ISRU
```

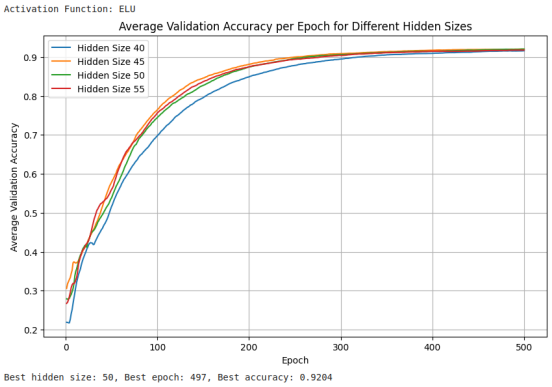



```
Best hidden size: 40, Best epoch: 498, Best accuracy: 0.9101
```

```
Activation Function: Smooth ReLU
```




```
Best hidden size: 45, Best epoch: 499, Best accuracy: 0.9114
```



1.6 Final Results

We use the MLP using the best results of above tests and use them on the validation data, from the result we generate the priors using the frequency analysis and then do the MAP classification using those priors but the original means and variances.

- 100 samples training data: 0.9278900027275085
- 500 samples training data: 0.9246799945831299
- 1000 samples training data: 0.9271900057792664
- 5000 samples training data: 0.9274200201034546
- 10000 samples training data: 0.929069995880127

1.7 Conclusion

- Regarding the training graphs to get the best number of neurons and the best activation function, we can see that almost all the graphs perform similarly regardless of number of samples, but the training was done till 500 epochs. Around 100 epochs, we can see more pronounced differences. Also, even though the shape of the curve for the same activation function are similar, we can see that those which used higher number of training samples have a "smoother" curve.
- This may be explained as: When a larger number of samples are used, the model gets more diverse data to learn from, which typically leads to more stable and consistent updates to the model's weights during training.
- Surprisingly, the MLP priors perform better than the actual priors used to generate the data
- This can be explained as: The randomness in the data generation might cause slight deviations from the ideal Gaussian distribution, which the MLP can learn to handle better than a fixed Gaussian prior.

2 Question 2

2.1 Data Generation

x is a 2-dimensional vector which is obtained from one of 4 Gaussian distributions, which have their own means and variances. 2 of the 4 Gaussian components have significant overlap.

$$P_0 = 0.3, P_1 = 0.2, P_2 = 0.25, P_3 = 0.25$$

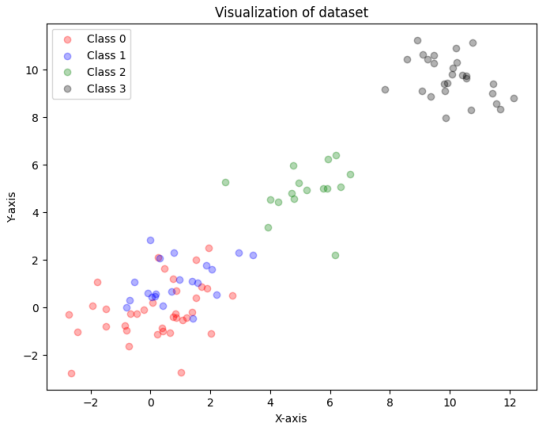
$$\mu_0 = \begin{bmatrix} 0 & 0 \end{bmatrix} \mu_1 = \begin{bmatrix} 1 & 1 \end{bmatrix} \mu_2 = \begin{bmatrix} 5 & 5 \end{bmatrix} \mu_3 = \begin{bmatrix} 10 & 10 \end{bmatrix}$$

$$\Sigma_0 = \begin{bmatrix} 1.5 & 0.5 \\ 0.5 & 1.5 \end{bmatrix} \Sigma_1 = \begin{bmatrix} 1.0 & 0.3 \\ 0.3 & 1.0 \end{bmatrix} \Sigma_2 = \begin{bmatrix} 1.2 & 0.0 \\ 0.0 & 1.2 \end{bmatrix} \Sigma_3 = \begin{bmatrix} 0.8 & -0.2 \\ -0.2 & 0.8 \end{bmatrix}$$

For each sample, the label is chosen as L_i with probability P_i
For label L_i:

$$x \sim \mathcal{N}(\mu_i, \Sigma_i^2)$$

Example of data:



2.2 Maximum Likelihood Parameter Estimation Theory

- Likelihood function:

$$L(\theta) = \prod_{i=1}^{N} p(x_i \mid \theta)$$

- Log likelihood function:

$$\log L(\theta) = \sum_{i=1}^{N} \log \left(\sum_{k=1}^{K} \pi_k \mathcal{N}(x_i \mid \mu_k, \Sigma_k) \right)$$

- E-step posterior probability:

$$\gamma(z_{ik}) = \frac{\pi_k \mathcal{N}(x_i \mid \mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x_i \mid \mu_j, \Sigma_j)}$$

- M-step mixing coefficients update:

$$\pi_k^{(t+1)} = \frac{1}{N} \sum_{i=1}^{N} \gamma(z_{ik})$$

- M-step means update:

$$\mu_k^{(t+1)} = \frac{\sum_{i=1}^{N} \gamma(z_{ik}) x_i}{\sum_{i=1}^{N} \gamma(z_{ik})}$$

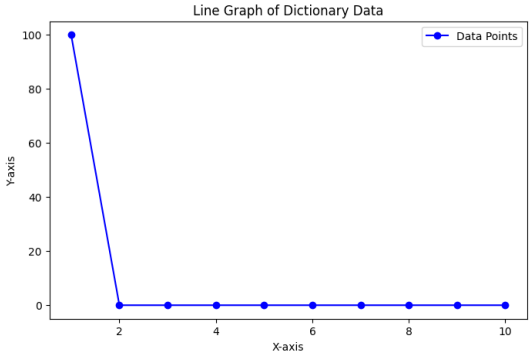
- M-step covariances update:

$$\Sigma_k^{(t+1)} = \frac{\sum_{i=1}^{N} \gamma(z_{ik})(x_i - \mu_k^{(t+1)})(x_i - \mu_k^{(t+1)})^T}{\sum_{i=1}^{N} \gamma(z_{ik})}$$

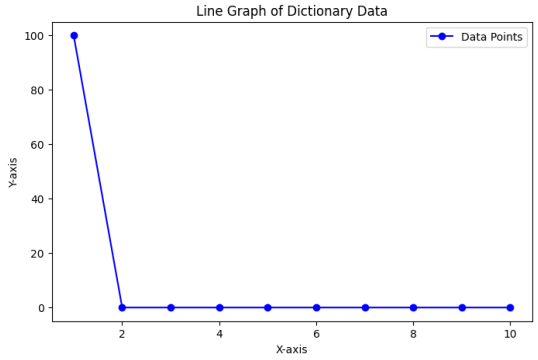
2.3 Results

We use above estimator 100 times for different training samples and record the model outputs.

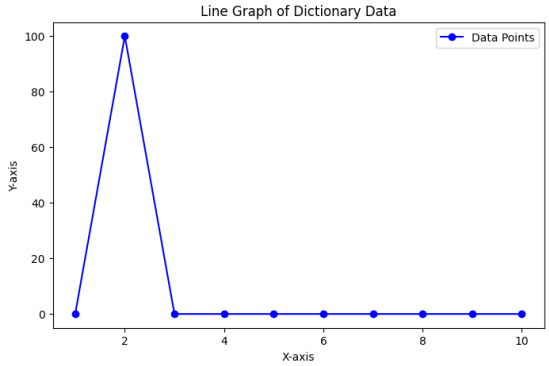
- Using 10 training samples shows error with function written from scratch because higher order models cannot be judged using only 10 samples, so result is calculated for lower orders only:



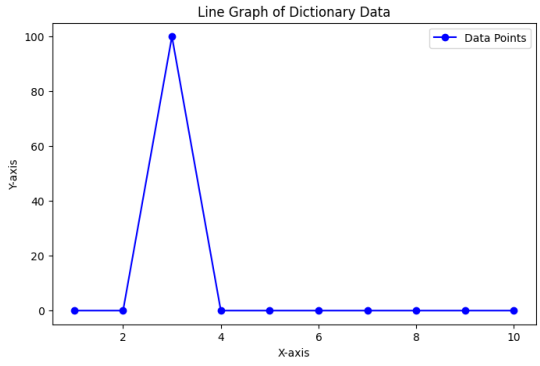
- Using 10 training samples and in-built function:



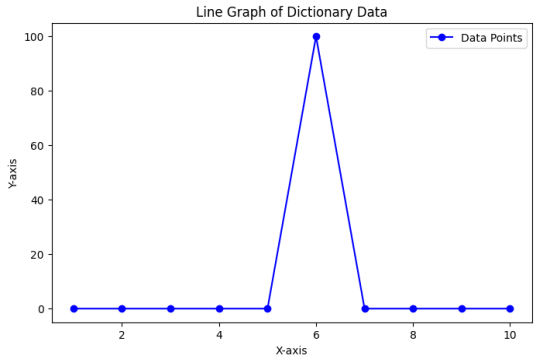
- Using 100 training samples and function written from scratch:



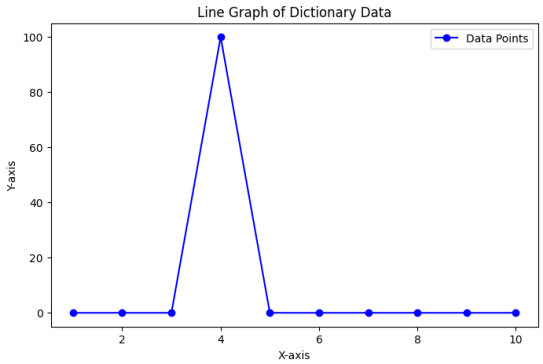
- Using 100 training samples and in-built function:



- Using 1000 training samples and function written from scratch:



- Using 1000 training samples and in-built function:



So, as we increase the training data, the accuracy of the model increases. Also, my function from scratch, admittedly, does not perform as good as the in-built function which achieves almost 100% accuracy with 1000 training samples.

Link to the GitHub Folder or copy this into your web browser:
https://github.com/Dhruv-2020EE30592/EECE-5644/tree/main/Assignment-3