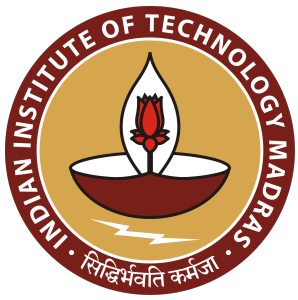
ASSIGNMENT 3

CS5691 Pattern Recognition and Machine Learning

CS5691 Assignment 3

Team Members:	
BE17B007	N Sowmya Manojna
PH17B010	Thakkar Riya Anandbhai
PH17B011	Chaithanya Krishna Moorthy

Indian Institute of Technology, Madras



Contents

1 Dataset 1A

This dataset contains data for four classes - 0, 1, 2 and 3. The classes are linearly separable and the dimension of the feature space is 2.

1.1 Perceptron

Varying the hyperparameter : Learning Rate (η) for the Perceptron model, the accuracies on the training and validation (30% of the file `dev.csv`) data for all possible pairings of the classes (leading to a total of 6 pairs) were obtained and the best η value based on CV accuracies was chosen as follows:

1.1.1 Classes 0 and 1

Hyperparameter	Training Accuracy	CV Accuracy
0.001	100.0	100.0
0.005	100.0	100.0
0.01	100.0	100.0
0.05	100.0	100.0
0.1	100.0	100.0
1.0	100.0	100.0
5.0	100.0	100.0
10.0	100.0	100.0
100.0	100.0	100.0

Table 1: Training and CV accuracies for classes 0 and 1 of dataset 1A.

The accuracy is 100% for all values of the hyperparameters. Taking the default value of 0.01, the accuracy on the test data is **100%**. The confusion matrices for the training and test data are as in Figure 1.

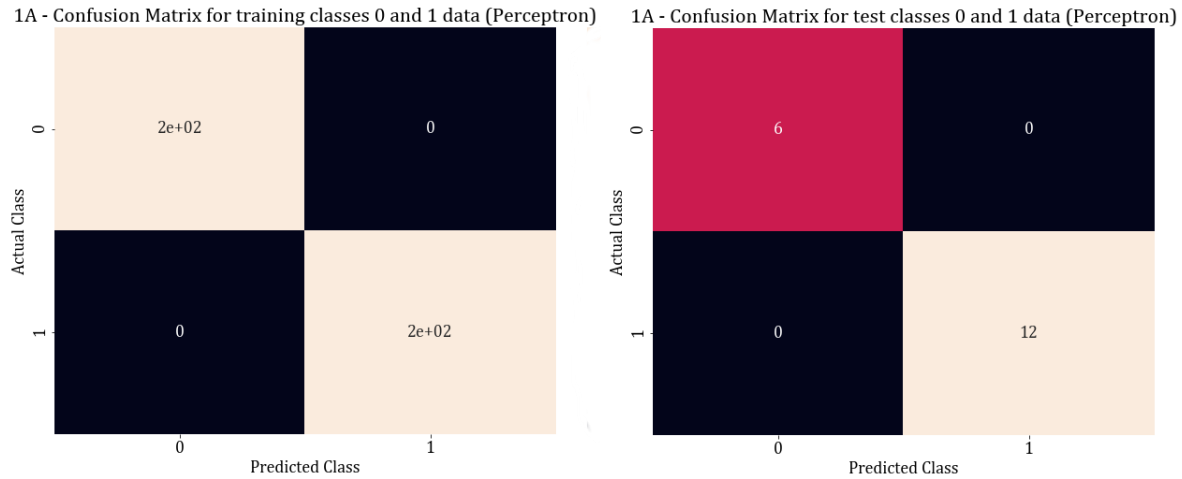


Figure 1: Confusion matrices for training and test data belonging to classes 0 and 1 of data 1A using perceptron classifier

The decision region plot for the perceptron classifier is in Figure 2.

Figure 2: Decision region plot for classes 0 and 1 of data 1A using perceptron classifier

1.1.2 Classes 0 and 2

Hyperparameter	Training Accuracy	CV Accuracy
0.001	100.0	100.0
0.005	100.0	100.0
0.01	100.0	100.0
0.05	100.0	100.0
0.1	100.0	100.0
1.0	100.0	100.0
5.0	100.0	100.0
10.0	100.0	100.0
100.0	100.0	100.0

Table 2: Training and CV accuracies for classes 0 and 2 of dataset 1A.

The accuracy is 100% for all values of the hyperparameters. Taking the default value of 0.01, the accuracy on the test data is **100%**. The confusion matrices for the training and test data are as in Figure 3.

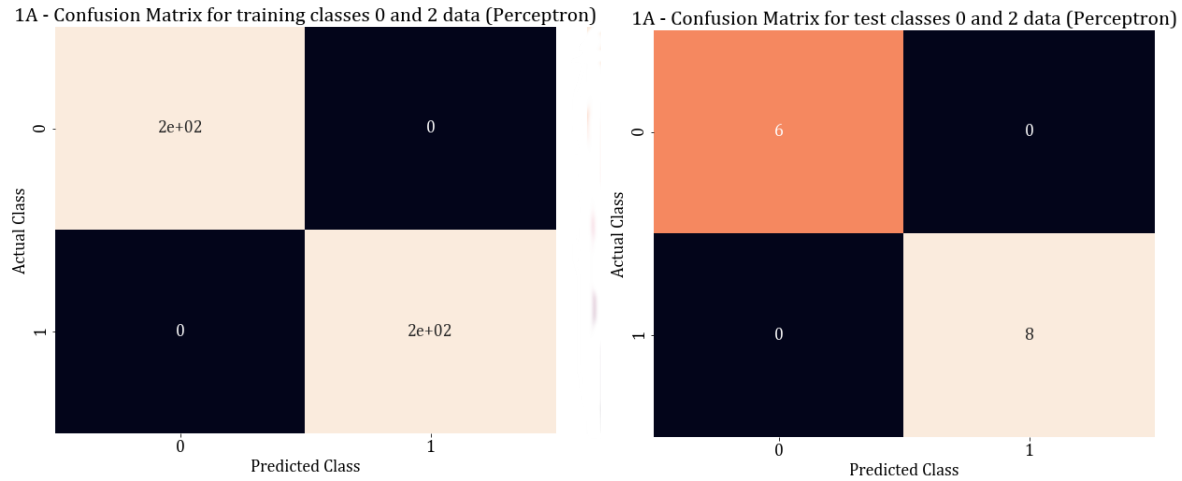


Figure 3: Confusion matrices for training and test data belonging to classes 0 and 2 of data 1A using perceptron classifier

The decision region plot for the perceptron classifier is in Figure 4.

Figure 4: Decision region plot for classes 0 and 2 of data 1A using perceptron classifier

1.1.3 Classes 0 and 3

Hyperparameter	Training Accuracy	CV Accuracy
0.001	100.0	100.0
0.005	100.0	100.0
0.01	100.0	100.0
0.05	100.0	100.0
0.1	100.0	100.0
1.0	100.0	100.0
5.0	100.0	100.0
10.0	100.0	100.0
100.0	100.0	100.0

Table 3: Training and CV accuracies for classes 0 and 3 of dataset 1A.

The accuracy is 100% for all values of the hyperparameters. Taking the default value of 0.01, the accuracy on the test data is **100%**. The confusion matrices for the training and test data are as in Figure 5.

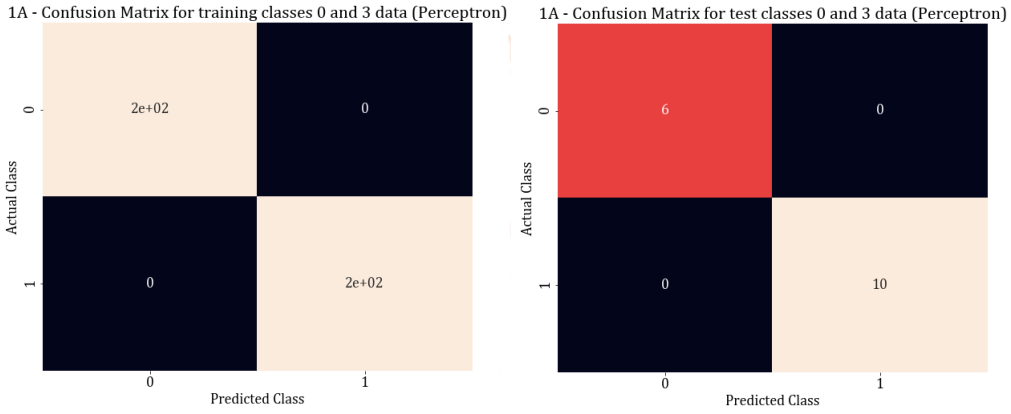


Figure 5: Confusion matrices for training and test data belonging to classes 0 and 3 of data 1A using perceptron classifier

The decision region plot for the perceptron classifier is in Figure 6.

Figure 6: Decision region plot for classes 0 and 3 of data 1A using perceptron classifier

1.1.4 Classes 1 and 2

Hyperparameter	Training Accuracy	CV Accuracy
0.001	100.0	97.5
0.005	100.0	97.5
0.01	100.0	97.5
0.05	100.0	100.0
0.1	100.0	100.0
1.0	100.0	100.0
5.0	100.0	100.0
10.0	100.0	100.0
100.0	100.0	100.0

Table 4: Training and CV accuracies for classes 1 and 2 of dataset 1A.

The accuracy on the CV data is best for $\eta = 0.05$. Using this value for the perceptron model, the accuracy on the test data is **100%**. The confusion matrices for the training and test data are as in Figure 7.

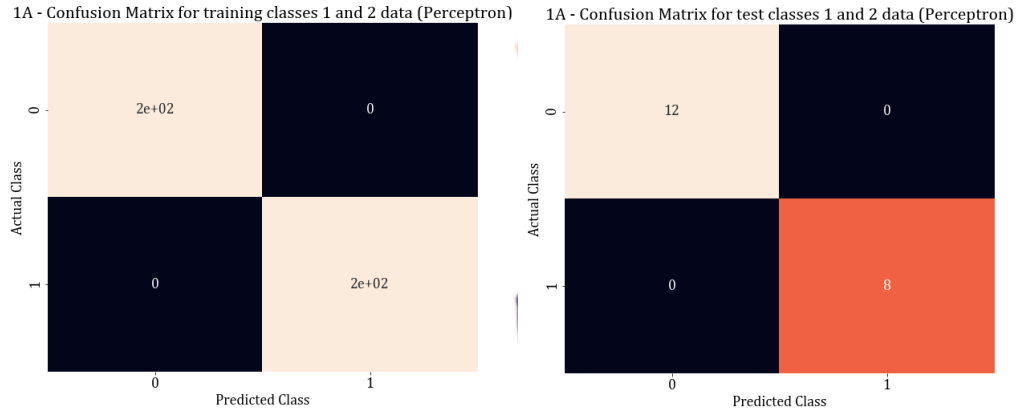


Figure 7: Confusion matrices for training and test data belonging to classes 1 and 2 of data 1A using perceptron classifier

The decision region plot for the perceptron classifier is in Figure 8.

Figure 8: Decision region plot for classes 1 and 2 of data 1A using perceptron classifier

1.1.5 Classes 1 and 3

Hyperparameter	Training Accuracy	CV Accuracy	Hyperparameter	Training Accuracy	CV Accuracy
0.001	100.0	100.0	1.0	100.0	100.0
0.005	100.0	100.0	5.0	100.0	100.0
0.01	100.0	100.0	10.0	100.0	100.0
0.05	100.0	100.0	100.0	100.0	100.0
0.1	100.0	100.0			

Table 5: Training and CV accuracies for classes 1 and 3 of dataset 1A.

The accuracy is 100% for all values of the hyperparameters. Taking the default value of 0.01, the accuracy on the test data is **100%**. The confusion matrices for the training and test data are as in Figure 9.

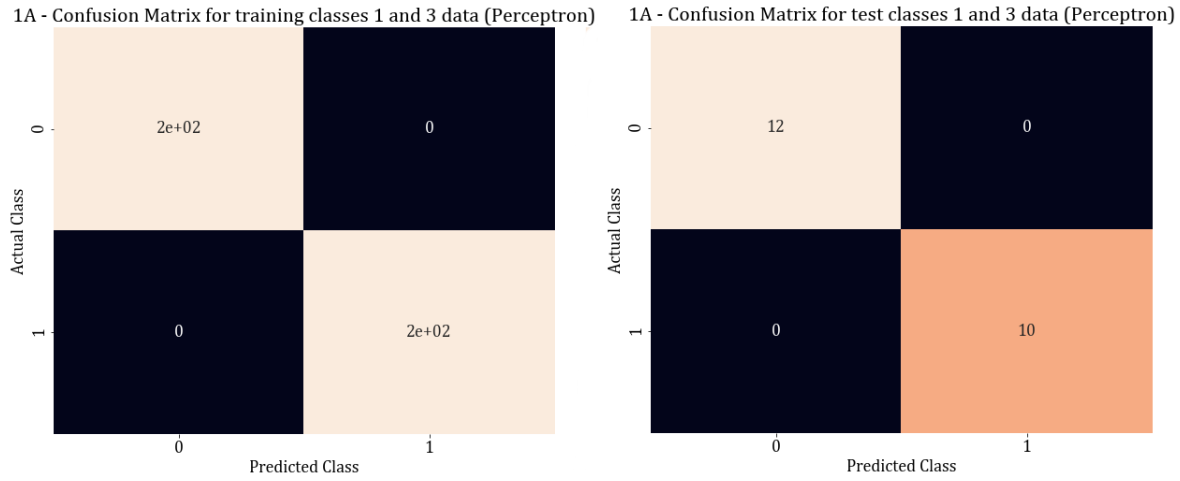


Figure 9: Confusion matrices for training and test data belonging to classes 1 and 3 of data 1A using perceptron classifier

The decision region plot for the perceptron classifier is in Figure 10.

Figure 10: Decision region plot for classes 1 and 3 of data 1A using perceptron classifier

1.1.6 Classes 2 and 3

Hyperparameter	Training Accuracy	CV Accuracy	Hyperparameter	Training Accuracy	CV Accuracy
0.001	100.0	92.86	1.0	100.0	100.0
0.005	100.0	100.0	5.0	100.0	97.62
0.01	100.0	100.0	10.0	100.0	100.0
0.05	100.0	100.0	100.0	100.0	100.0
0.1	100.0	100.0			

Table 6: Training and CV accuracies for classes 2 and 3 of dataset 1A.

The accuracy on the CV data is best for η greater than or equal to 0.005. Using the default value of $\eta = 0.01$ for the perceptron model, the accuracy on the test data is **100%**. The confusion matrices for the training and test data are as in Figure 11.

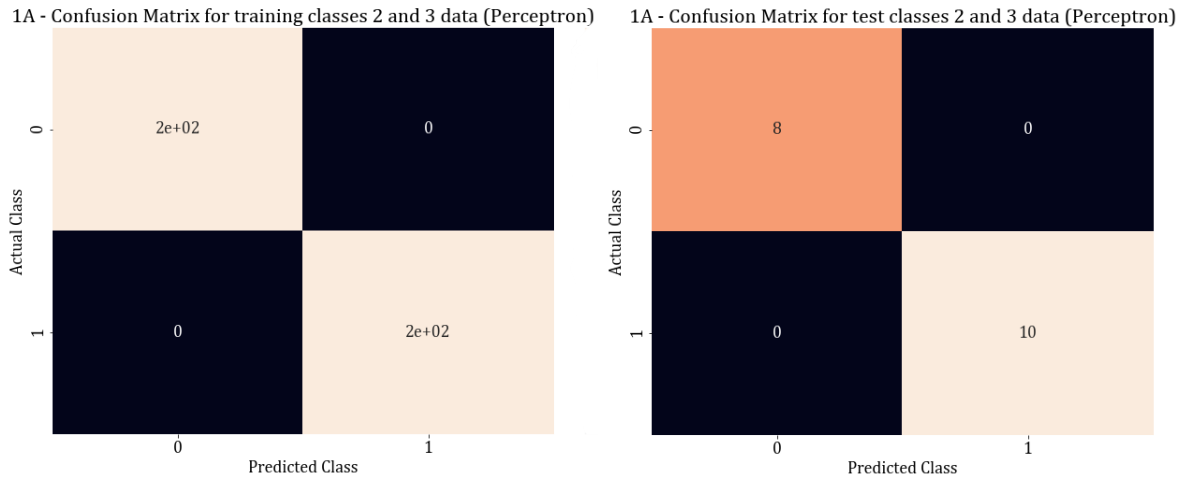


Figure 11: Confusion matrices for training and test data belonging to classes 2 and 3 of data 1A using perceptron classifier

The decision region plot for the perceptron classifier is in Figure 12.

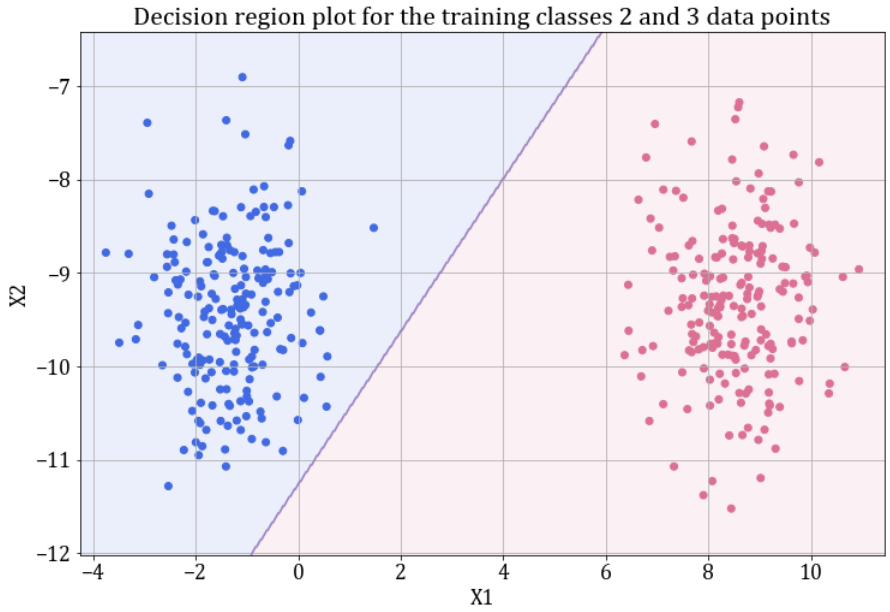


Figure 12: Decision region plot for classes 2 and 3 of data 1A using perceptron classifier

1.1.7 Observations

From the above obtained decision boundary plots we can observe that:

- The decision boundaries are linear
- The decision boundaries obtained are not optimal in nature and tend to be extremely close to either of the claases depending on the initial conditions used.

1.2 MLFFNN

The hyperparameters varied and sweeped for are - `hidden layer size`, `optimizer`, `batch size`, `learning rate`, `L2 regularization` α. They were varied as follows:

- `hidden_layer_sizes`: 5,8,10,15
- `activation`: `logistic`, `tanh`, `relu`
- `solver`: `lbfgs`, `sgd`, `adam`
- `batch_size`: 100, 200
- `alpha`: 0, 0.0001
- `learning_rate`: `constant`, `adaptive`, `invscaling`

Hence, a total of 432 parameter combinations were sweeped for and analyzed.

1.2.1 Classification Accuracies

The classification accuracies on the training and validation datasets (30% of the `dev.csv`) are as follows:

# Neurons	Activation	Solver	Batch Size	α	Learning Rate	Accuracy	Validation Accuracy
5	tanh	lbfgs	200	0.0001	adaptive	100.0	100.0
5	tanh	lbfgs	200	0.0001	constant	100.0	100.0
5	tanh	lbfgs	200	0.0	invscaling	100.0	100.0
5	tanh	lbfgs	200	0.0	adaptive	100.0	100.0
5	tanh	lbfgs	200	0.0	constant	100.0	100.0
5	tanh	lbfgs	100	0.0	adaptive	100.0	100.0
5	tanh	lbfgs	100	0.0001	invscaling	100.0	100.0
5	relu	lbfgs	200	0.0	constant	100.0	100.0
5	relu	lbfgs	100	0.0001	invscaling	100.0	100.0
5	relu	lbfgs	200	0.0	adaptive	100.0	100.0

Table 7: Best 10 Train and Validation Accuracies obtained after performing a `GridSearch` on 432 parameter combinations.

1.2.2 Best Model

The parameter combination were additionally sorted based on minimum fitting time (least fitting time - first) and the model that gave the best accuracy the fastest (and potentially the most minimal model that best fits the data), was chosen. Hence the best parameter combination chosen is:

- `hidden_layer_sizes`: 5
- `activation`: tanh
- `solver`: lbfgs
- `batch_size`: 200
- `alpha`: 0.0001
- `learning_rate`: adaptive

The classification accuracy of the best model on the testing data is: **100%**. The confusion matrices obtained are as follows:

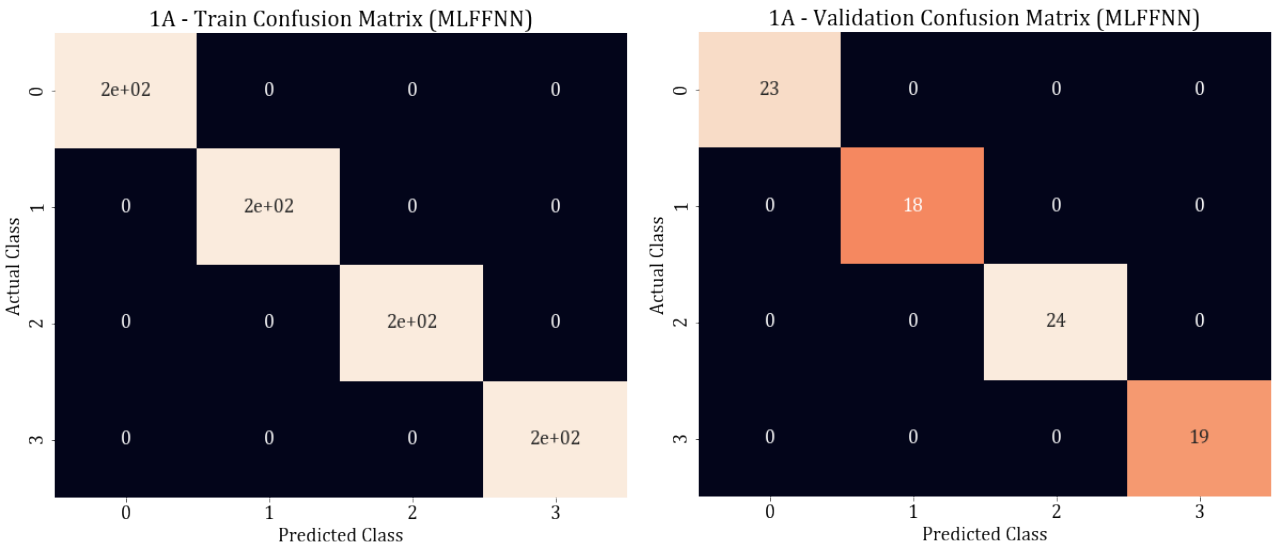


Figure 13: Training and Validation confusion matrices obtained for the best parameter combination, on the left and right respectively.

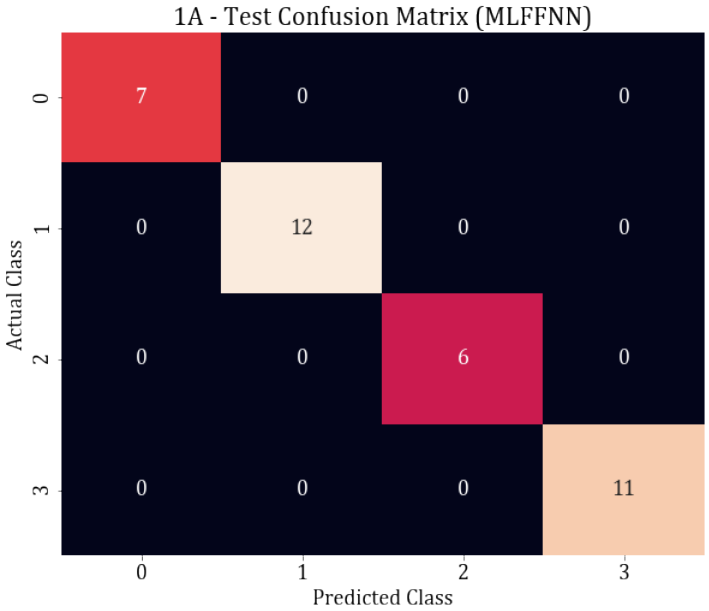


Figure 14: Testing confusion matrices obtained for the best parameter combination.

1.2.3 Decision Region

The decision region plots obtained. From the Figure 15, we can see that the decision boundaries aren't linear and that they are slightly curved (nonlinear) at the intersection. This is potentially because of the smooth non-linearity of the `tanh` activation function.

The plot obtained is as follows:

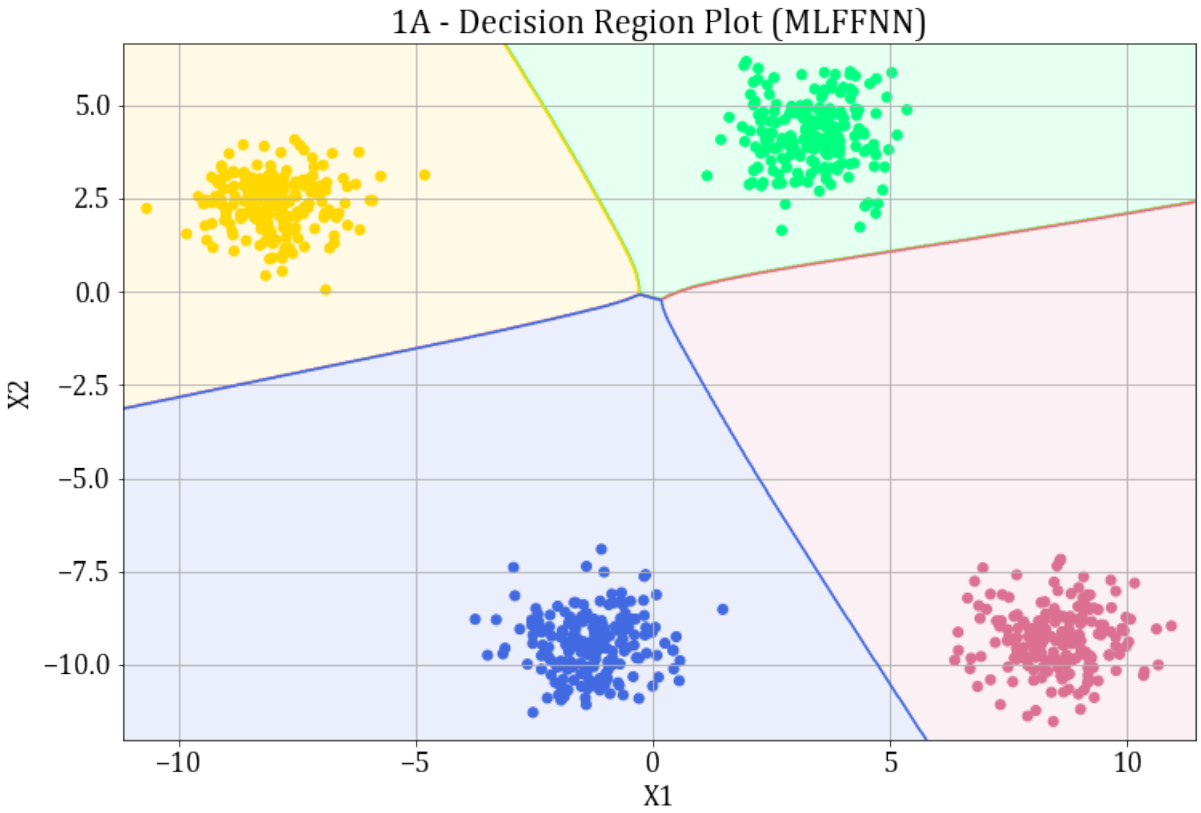


Figure 15: Decision Region Plot obtained for the best parameter combination.

1.3 Linear SVM

The Support Vector Machines (SVMs) are supervised learning models that can be used for regression, classification and outlier detection. In case of a linear classifier, the goal of the SVM is to construct a hyperplane(s) such that the distance from the nearest training data point is maximized. This distance is called the functional margin.

1.3.1 Mathematical Formulation

For a hyperplane of the form:

$$g(\vec{x}) = \vec{w}^T \vec{x} - w_0 = 0 \tag{1}$$

The distance of any point $\vec{x}$ from the hyperplane is then equivalent to:

$$r = \frac{g(\vec{x})}{||\vec{w}||} \tag{2}$$

For a canonically seperating hyperplane,

$$|g_c(\vec{x^*})| = 1 \tag{3}$$

Where $\vec{x^*}$ is the training example nearest to the hyperplane. The distance of $\vec{x^*}$ from the hyperplane is then given by:

$$r = \frac{1}{||\vec{w}||} \tag{4}$$

For all points $(\vec{x_n}, t_n)$,

$$(t_n)(\vec{w}^T X_n + w_0) \geq 1 \tag{5}$$

Where t_n is the target variable $\epsilon\ (-1, 1)$.
The primal form of the Lagrangian objective function is:

$$L_p(\vec{w}, w_0, \vec{\alpha}) = \frac{||\vec{w}||}{2} - \sum_{n=1}^{N} \alpha_n [t_n(\vec{w}^T \vec{x_n} + w_0) - 1] \tag{6}$$

Optimality condition:

$$\frac{\partial L_p}{\partial \vec{w}} = \vec{0} \tag{7}$$

$$\frac{\partial L_p}{\partial w_0} = 0 \tag{8}$$

Substituting the form of L_p and differentiating:

$$\vec{w} - \sum_{n=1}^{N} \alpha_n t_n \vec{x_n} = \vec{0} \tag{9}$$

$$\sum_{n=1}^{N} \alpha_n t_n = 0 \tag{10}$$

The dual form of Lagrangian can be obtained by substituting the form of $\vec{w}$ in the primal form.
The task is now to find $\vec{\alpha}$ for which $L_D(\vec{\alpha})$ is optimal subject to $\alpha_n \geq 0$. This is a quadratic problem and can be solved by a QP solver.

$$L_D(\vec{\alpha}) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{m=1}^{N} \sum_{n=1}^{N} \alpha_m \alpha_n t_m t_n \vec{x_m}^T \vec{x_n} \tag{11}$$

1.3.2 `sklearn svm.SVC()`

The `svm.SVC()` is a pre-defined module for Support Vector Classification task. The important optional arguments of this classifier are:

- `kernel` : Specifies the kernel type to be used in the algorithm, available options are:
  - Linear kernel: $< x, x' >$
  - Polynomial kernel: $(\gamma < x, x' > + r)^d$
  - RBF kernel: $-\gamma ||x - x'||^2$
  - sigmoid kernel: $\tanh(\gamma < x, x' > + r)$

  The default kernel is the rbf(/gaussian). For dataset 1A, linear kernel is used.
- `C`: strictly positive float. By default it is 1.0. The strength of regularization is inversely proportional to C. For dataset 1A, C is varied from 0.1 to 1000.
- `degree`: This is relevant only when the kernel is set to polynomial and is ignored by all other kernel values.
- `gamma`: float or [`scale`, `auto`]. Set to `scale` by default, it is the kernel coefficient for `rbf`, `poly` and `sigmoid`.
- `decision_function_shape`: [`ovr`, `ovo`]. It specifies whether one-versus-rest or one-versus-one methodology should be used for multi-class classification. By default, it is `ovr`.

1.3.3 Pairwise SVM

The dataset 1A has 4 class labels: [0.0,1.0,2.0,3.0], and 2 dimensional feature vectors. Those are linearly separable as can be seen in the plot above.

- Data pre-processing:
  - There are no missing values in the data.
  - The ranges of feature vectors are nearly same, hence scaling is not essential.
- The data is then split according to the class labels to further build one-vs-one svm models.
- `sklearn.svm.SVC()`,with a linear kernel is then trained using a pair of classes at once, hence, number of models trained is $\frac{3(3+1)}{2} = 6$.

- The decision boundary and support vectors are then obtained for each of those models. However, the decision region for $y[i]$ vs $y[j]$ model is defined only for x_1 and x_2 range where $y[i]$ and $y[j]$ exist in the training data
- To predict the class label for a test point: we obtain 6 labels from the 6 different models trained earlier. From those 6 labels, the label occurring with the most frequency is chosen as the label for the test point, a collective decision region and confusion matrix are thus obtained.

1.3.4 Classification Accuracies

The hyper-parameter C is varied in the range [0.1,1,10,100,1000] and corresponding accuracy over train data and cross validation data is obtained as follows:

C	Train Accuracy	Validation Accuracy
0.1	1.00	1.00
1	1.00	1.00
10	1.00	1.00
100	1.00	1.00
1000	1.00	1.00

Table 8: Accuracy table for dataset 1A, Pairwise Linear SVM

Since the accuracy is independent of the value of hyper-parameter C, the default value $C = 1$ is used for the production of plots and confusion matrices.The accuracy obtained on test data set is also 100 %.

1.3.5 Confusion Matrices

Since the accuracy obtained is 100%, the confusion matrices are diagonal.

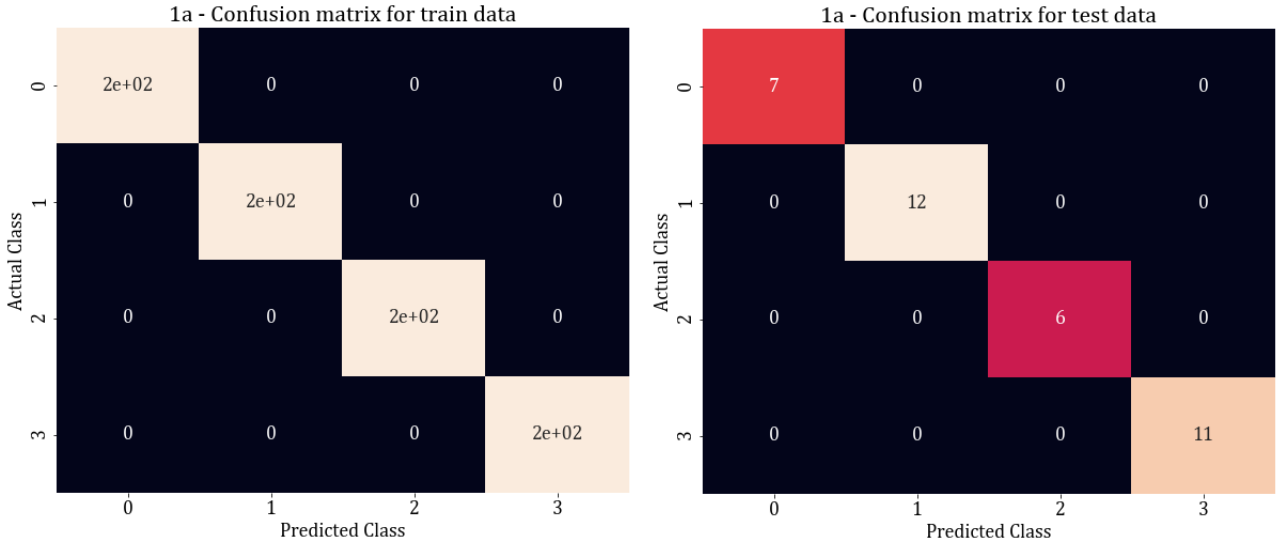


Figure 16: Confusion matrix for train and test data respectively

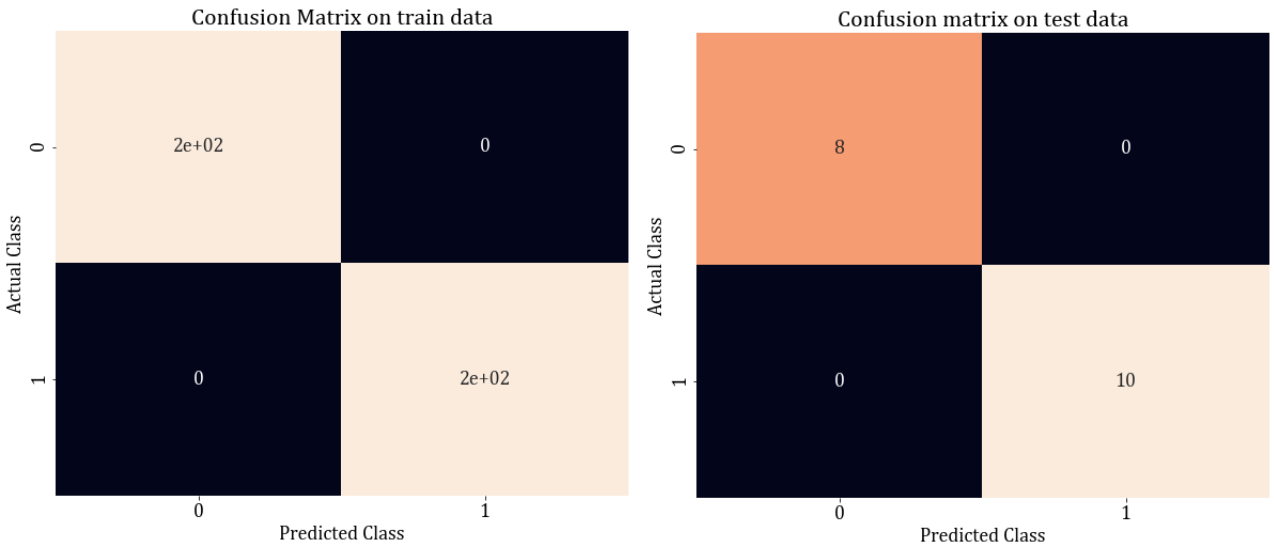


Figure 17: Confusion matrix for class 0 and 1, train and test data respectively

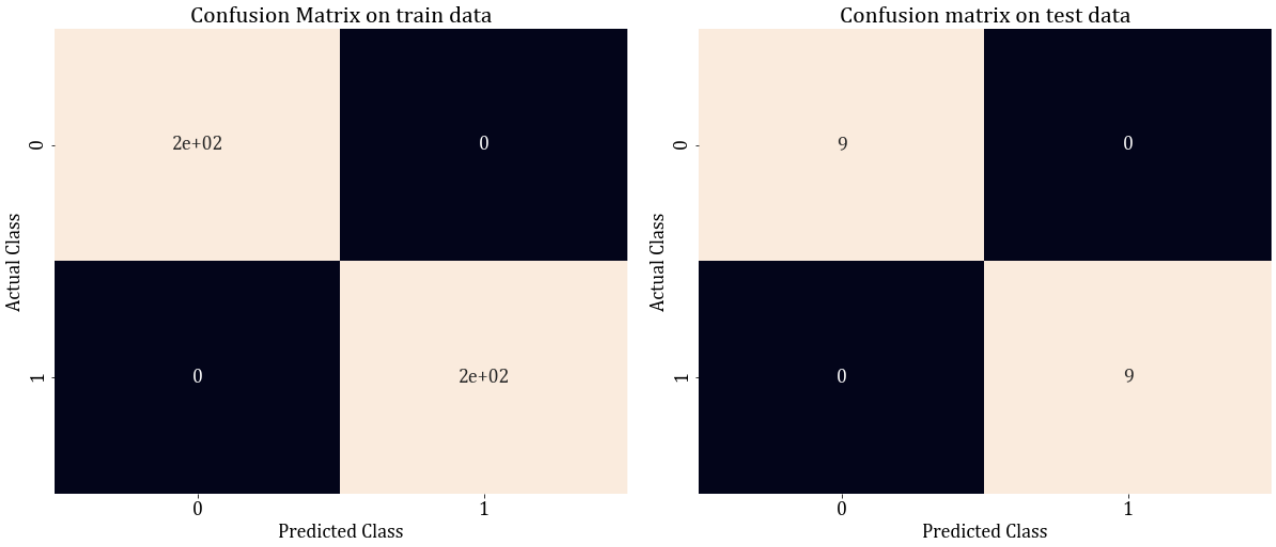


Figure 18: Confusion matrix for class 0 and 2, train and test data respectively

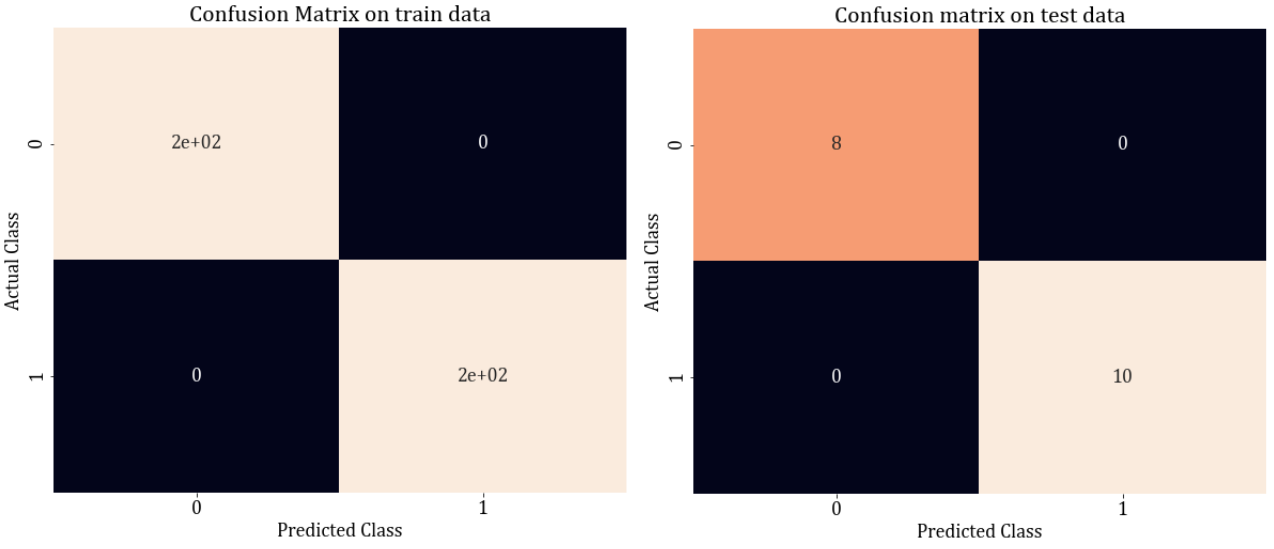


Figure 19: Confusion matrix for class 0 and 3, train and test data respectively

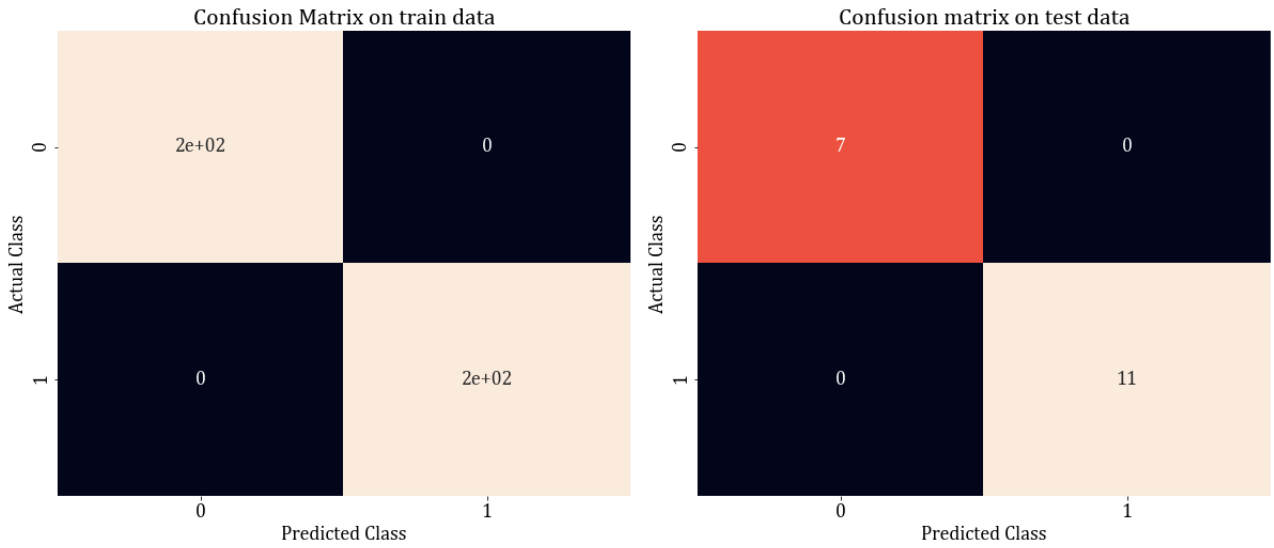


Figure 20: Confusion matrix for class 1 and 2, train and test data respectively

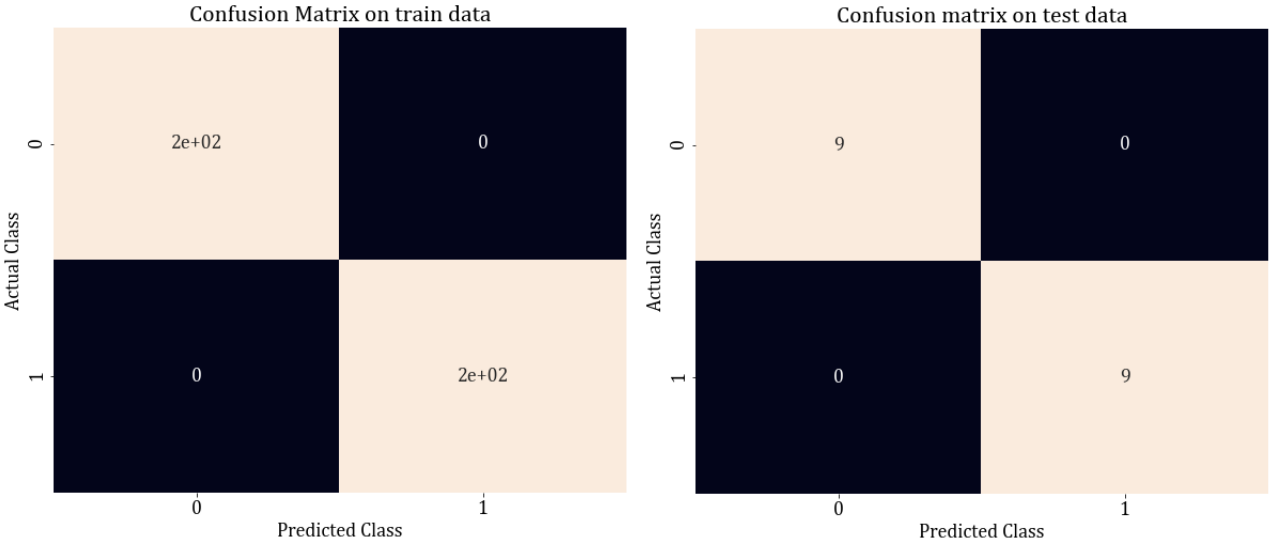


Figure 21: Confusion matrix for class 1 and 3, train and test data respectively

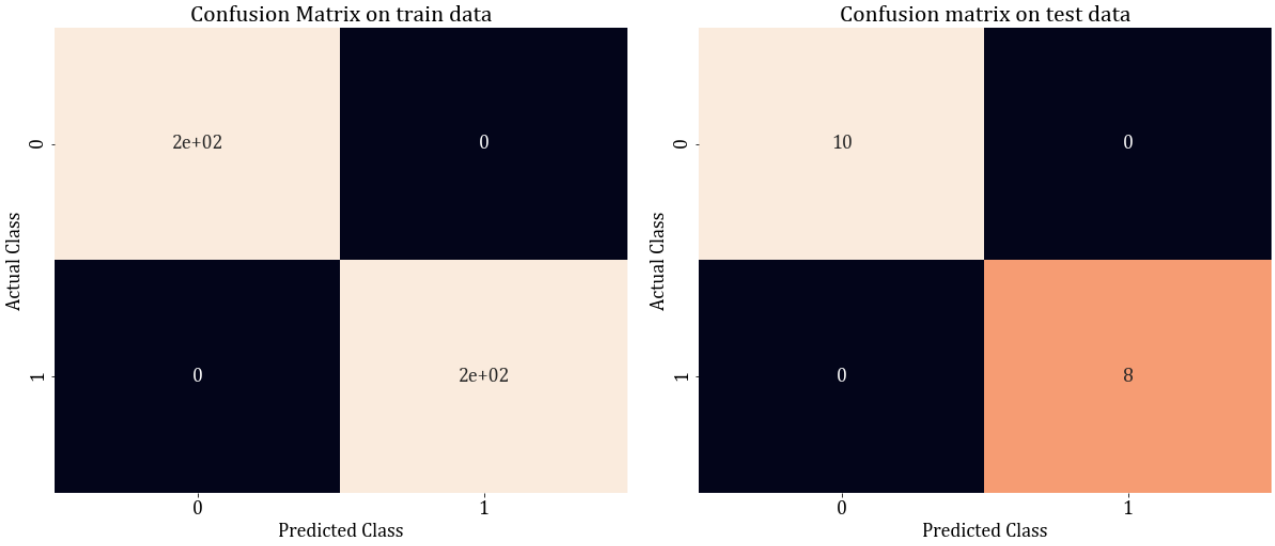


Figure 22: Confusion matrix for class 2 and 3, train and test data respectively

1.3.6 Decision Boundaries and Support vectors

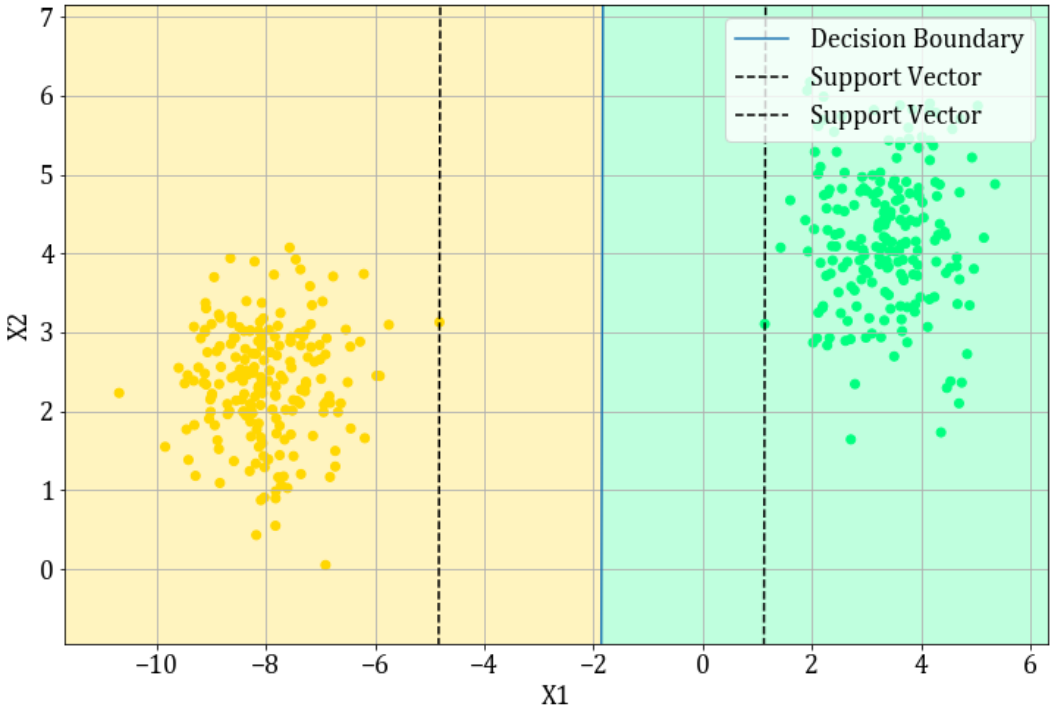


Figure 23: Decision region for class y=0 and y=2

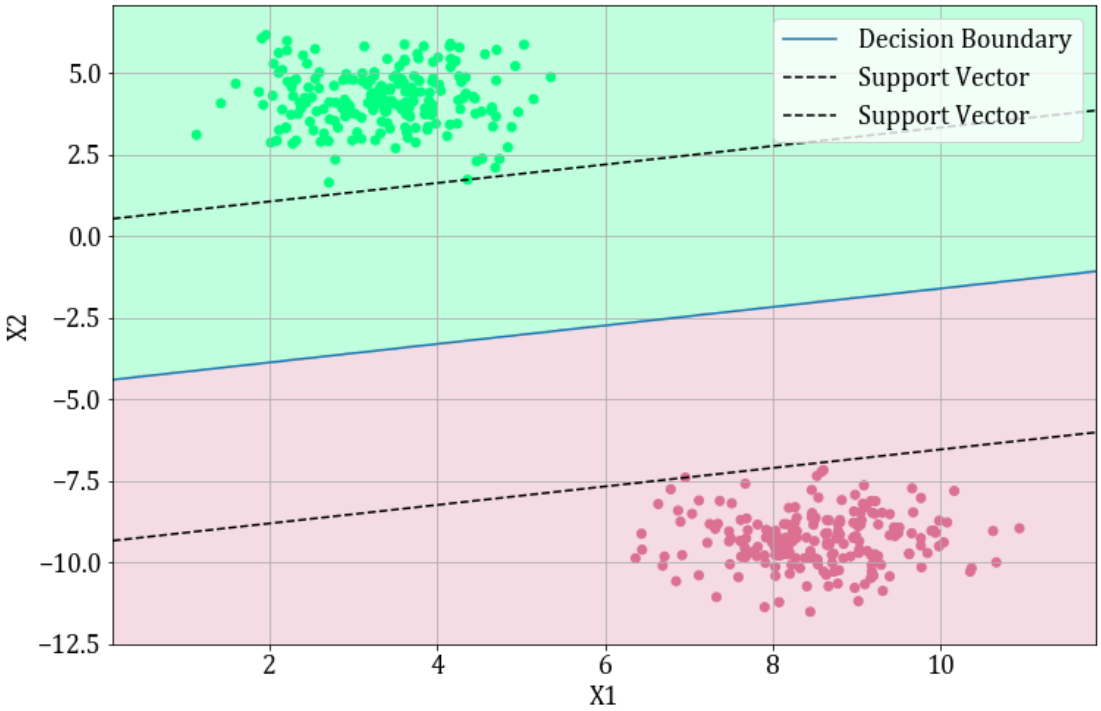


Figure 24: Decision region for class y=0 and y=2

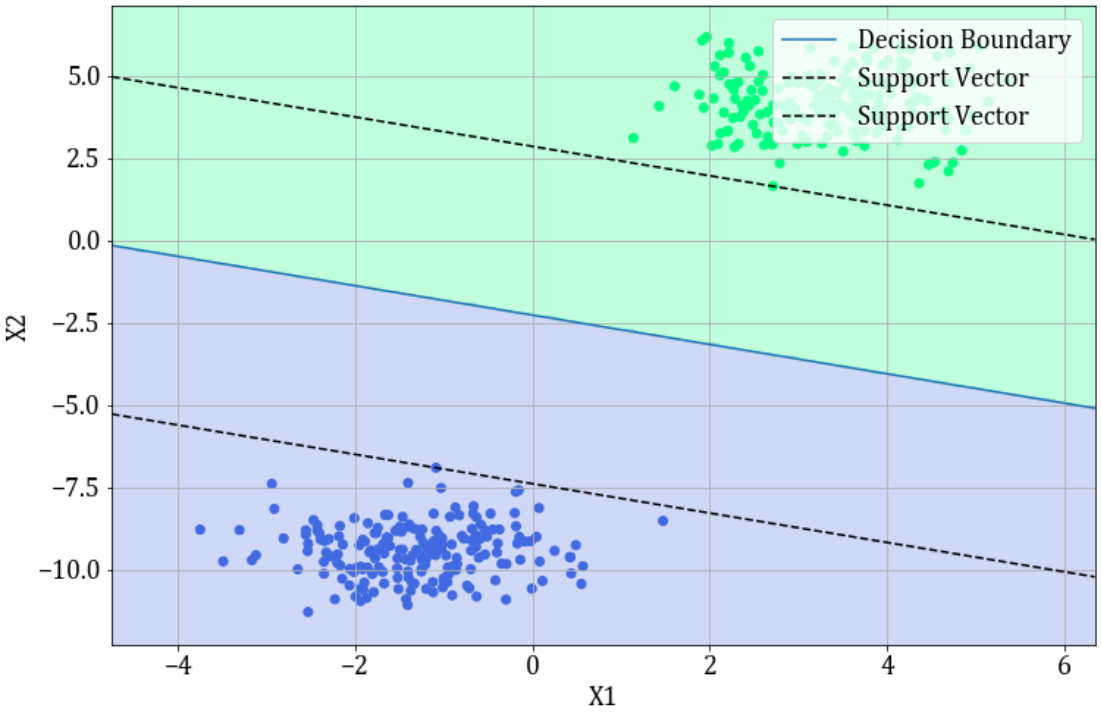


Figure 25: Decision region for class y=0 and y=3

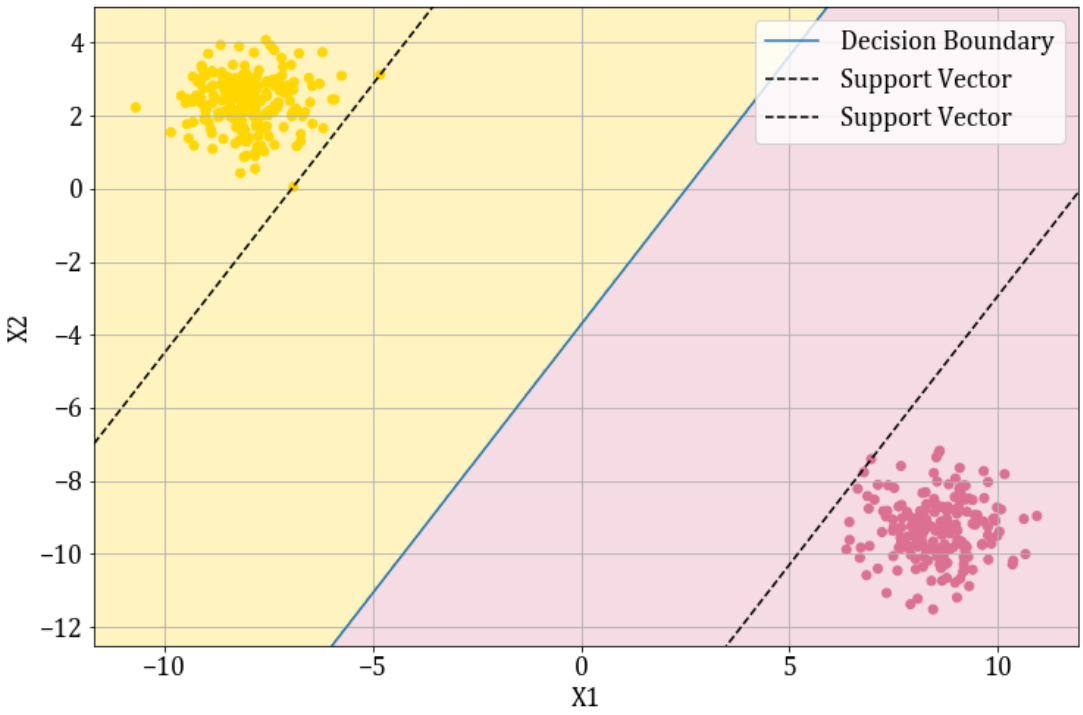


Figure 26: Decision region for class y=1 and y=2

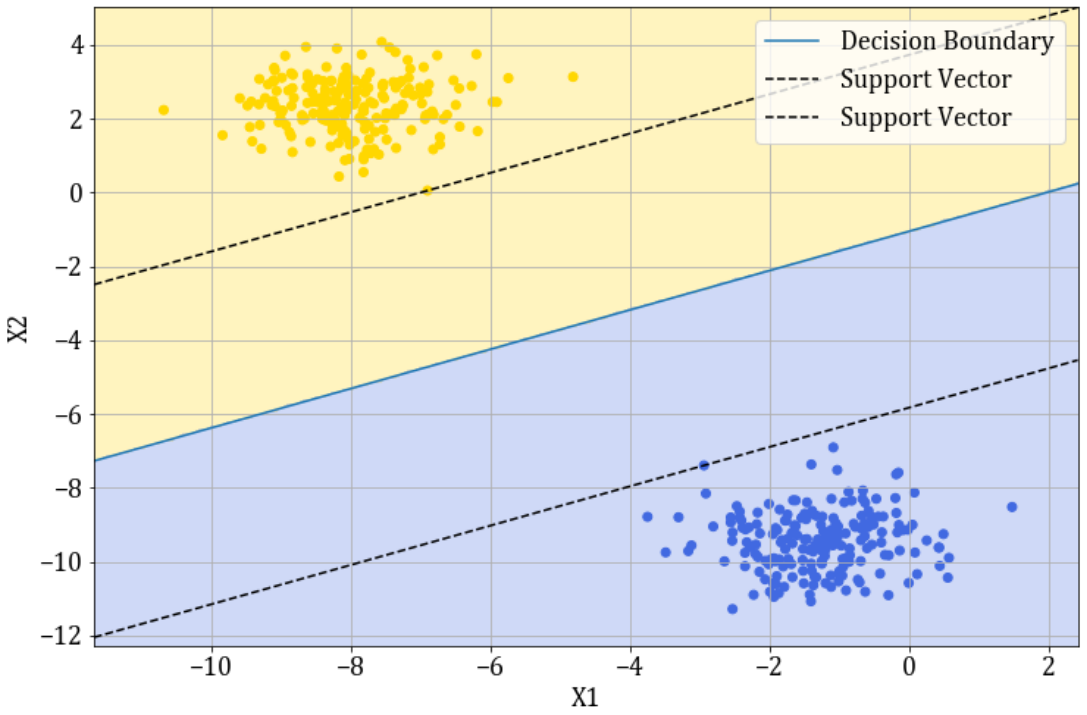


Figure 27: Decision region for class y=1 and y=3

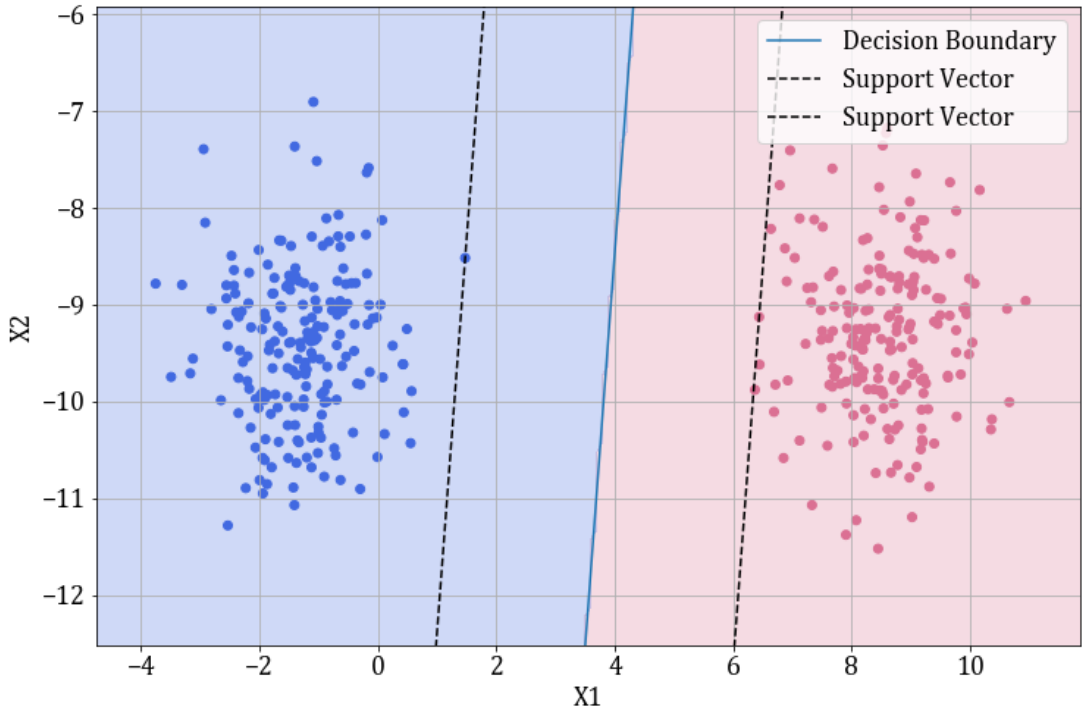


Figure 28: Decision region for class y=2 and y=3

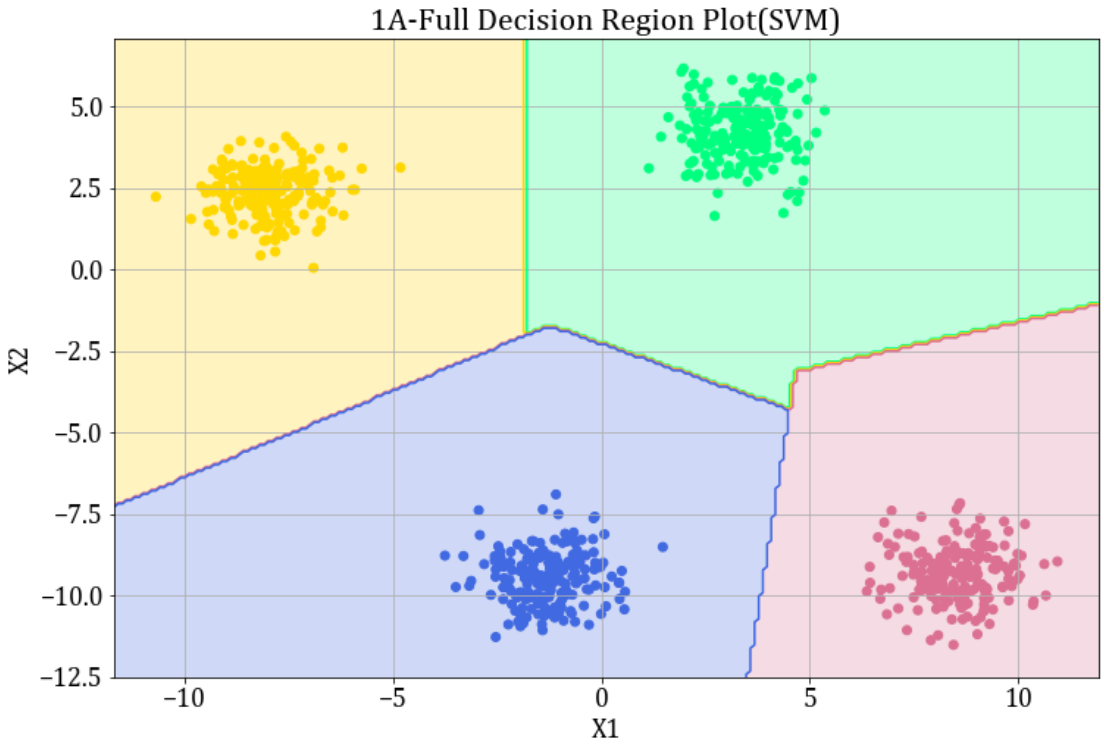


Figure 29: The effective decision region for dataset 1A

1.3.7 Conclusion

Observations from one-vs-one decision boundary plots:

- The separating hyper-planes are linear.
- The hyper-planes formed by the Support Vectors are parallel to the separating hyper-plane.
- Since no training data points lie between the supporting vector hyper-plane and the separating hyper-plane, it is a hard margin solution.

2 Dataset 1B

This dataset contains data for three classes - 0, 1 and 2. The classes are non-linearly separable and the dimension of the feature space is 2.

2.1 MLFFNN

The hyperparameters varied and sweeped for are - `hidden layer size`, `activation function`, `batch size`, `learning rate`, `L2 regularization` α. They were varied as follows:

- `hidden_layer_sizes`: (5,5), (6,6), (7,7), (8,8), (9,9), (10,10)
- `activation`: `logistic`, `relu`
- `batch_size`: 50, 100, 200,
- `early_stopping`: `True`, `False`
- `learning_rate`: `constant`, `adaptive`, `invscaling`
- `alpha`: 0.01, 0.001

Hence, a total of 432 parameter combinations were sweeped for and analyzed.

2.1.1 Classification Accuracies

The classification accuracies on the training and validation datasets (30% of the `dev.csv`) are as follows:

# Neurons	Activation	Batch Size	Early Stopping	Learning Rate	α	Accuracy	Validation Accuracy
(8, 8)	relu	50	False	adaptive	0.01	99.33	98.41
(8, 8)	relu	50	False	constant	0.001	99.33	98.41
(8, 8)	relu	50	False	invscaling	0.01	99.33	98.41
(8, 8)	relu	50	False	adaptive	0.001	99.33	98.41
(8, 8)	relu	50	False	invscaling	0.001	99.33	98.41
(8, 8)	relu	50	False	constant	0.01	99.33	98.41
(10, 10)	relu	50	False	adaptive	0.01	99.0	98.41
(10, 10)	relu	50	False	constant	0.01	99.0	98.41
(10, 10)	relu	50	False	invscaling	0.01	99.0	98.41
(10, 10)	relu	50	False	constant	0.001	99.0	96.82

Table 9: Best 10 Train and Validation Accuracies obtained after performing a `GridSearch` on 432 parameter combinations.

2.1.2 Best Model

The parameter combination were additionally sorted based on minimum fitting time (least fitting time - first) and the model that gave the best accuracy the fastest (and potentially the most minimal model that best fits the data), was chosen. Hence the best parameter combination chosen is:

- hidden_layer_sizes: (8, 8)
- activation: relu
- batch_size: 50
- early_stopping: False
- learning_rate: adaptive
- alpha (L2 regularization): 0.01

The classification accuracy of the best model on the testing data is: **96.296%**. The confusion matrices obtained are as follows:

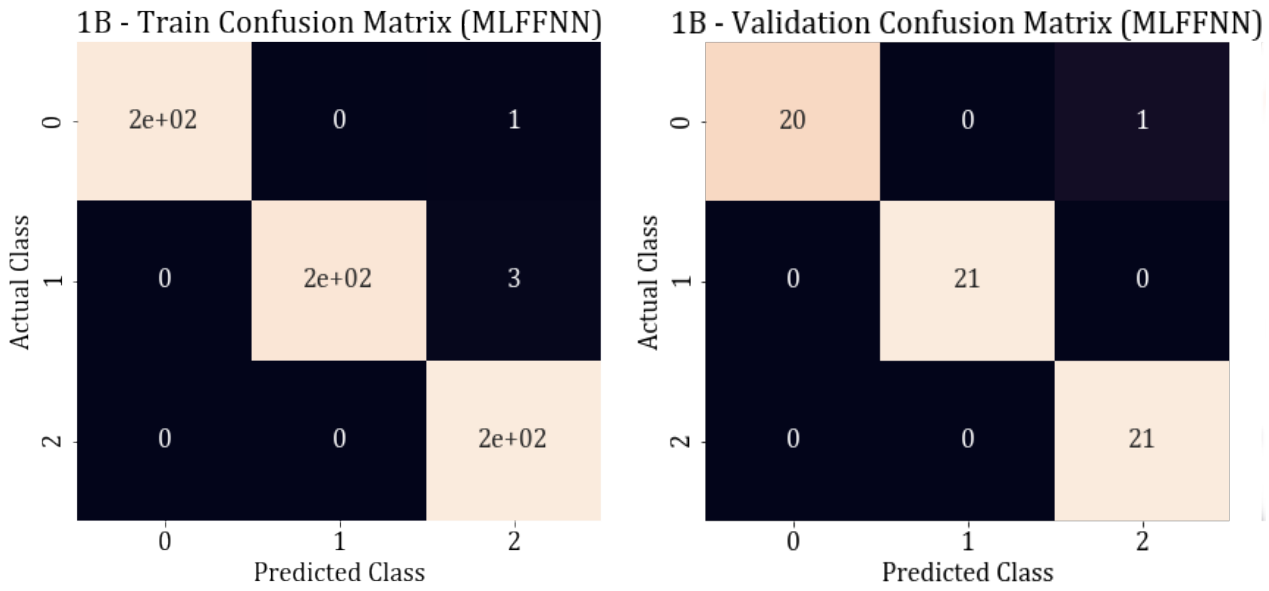


Figure 30: Training and Validation confusion matrices obtained for the best parameter combination, on the left and right respectively.

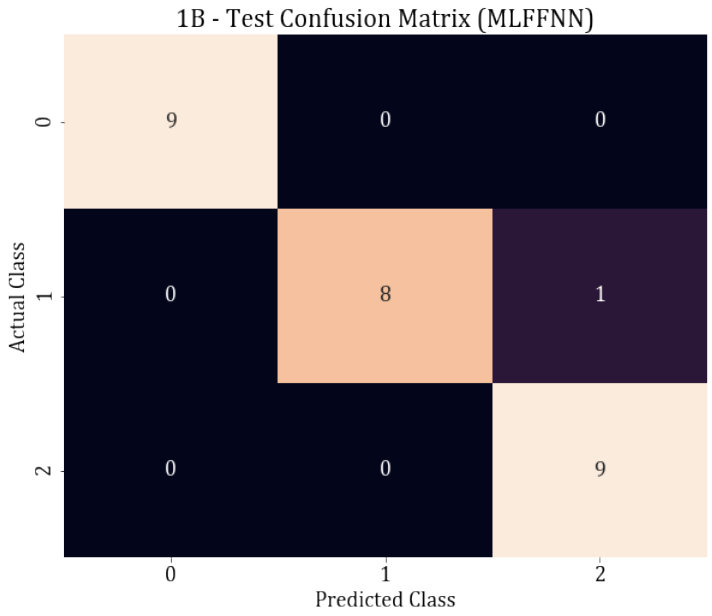


Figure 31: Testing confusion matrices obtained for the best parameter combination.

2.1.3 Decision Region

The decision region plots obtained. From the Figure 32, we can see that the decision boundaries aren't linear. However, they comprise of multiple small linear segments. This is potentially because of the linear discontinuous nature of the `ReLU` activation function.

The decision region plots obtained is as follows:

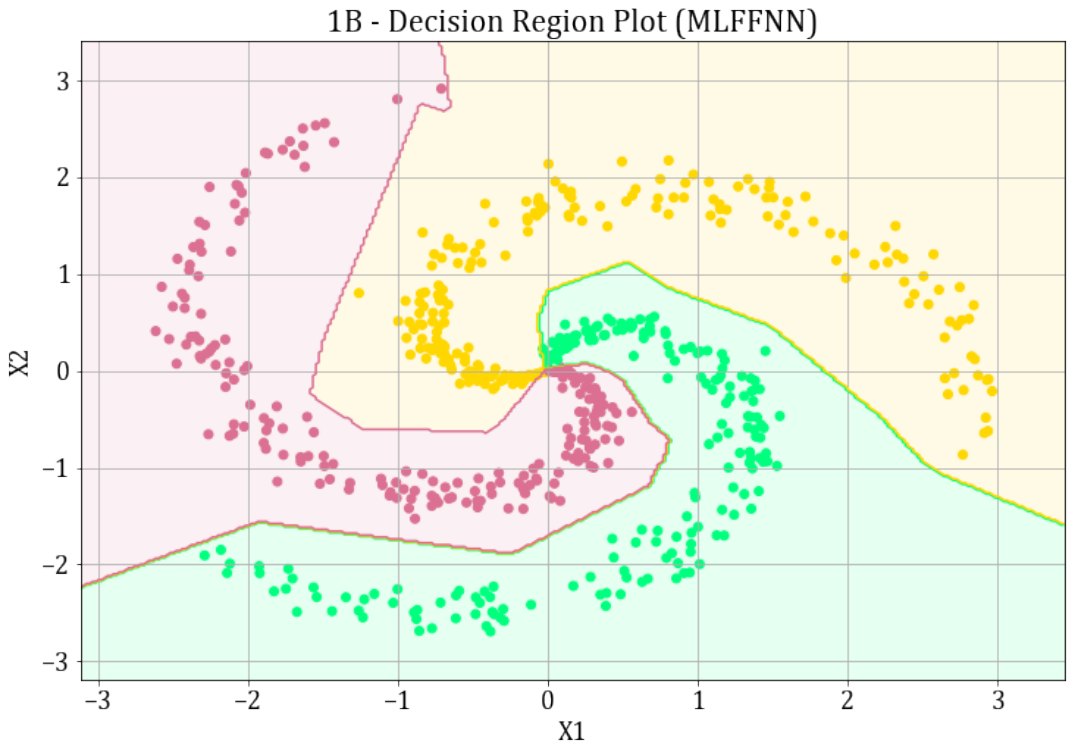


Figure 32: Decision Region Plot obtained for the best parameter combination.

2.1.4 Surface Plots

The neuron-wise surface plots obtained for the hidden and output layers is as follows:

2.1.4.1 Hidden Layer 1, Node 1

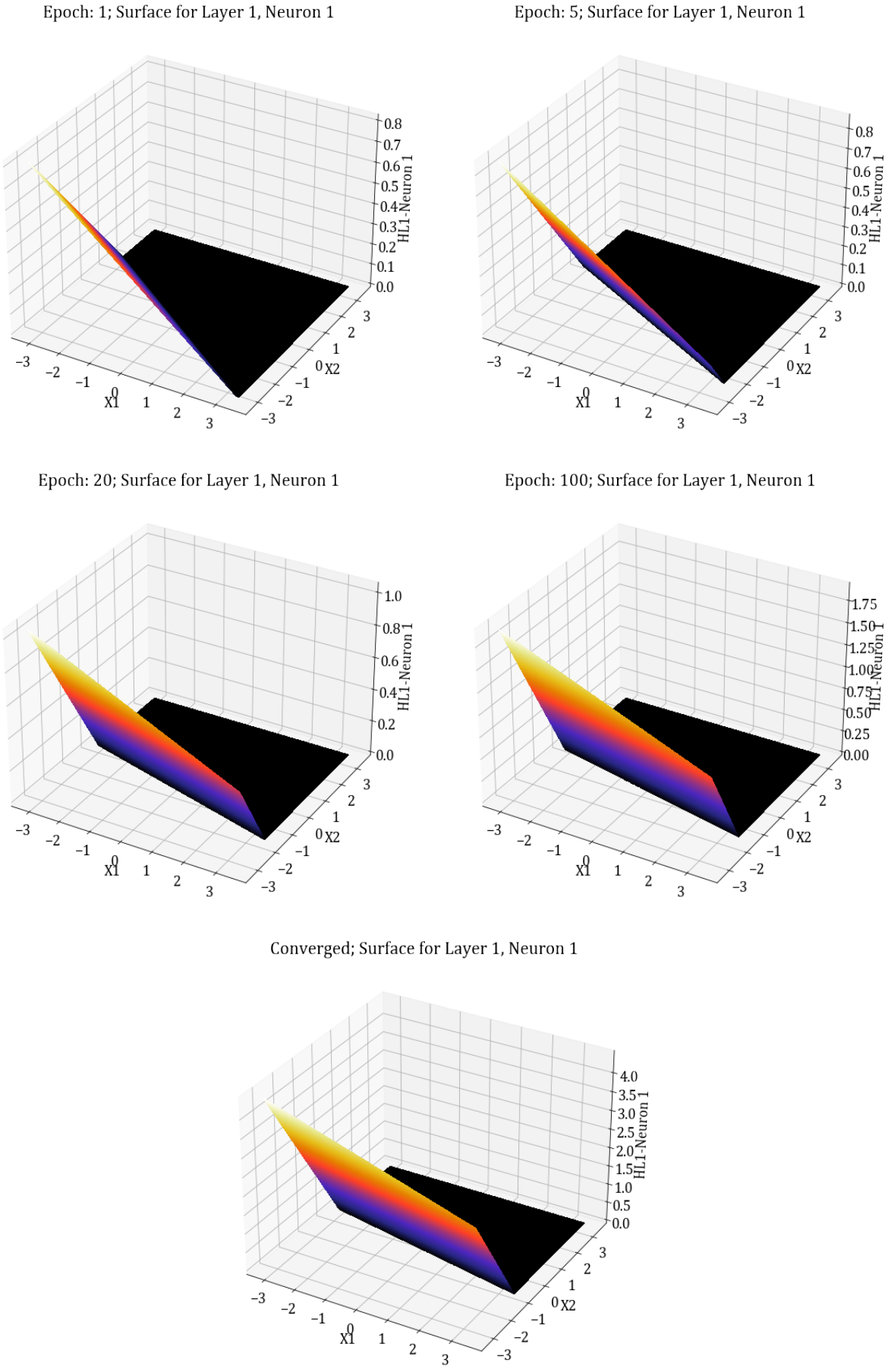


Figure 33: Surface Plots obtained for Hidden Layer 1, Neuron 1, across epochs.

2.1.4.2 Hidden Layer 1, Node 2

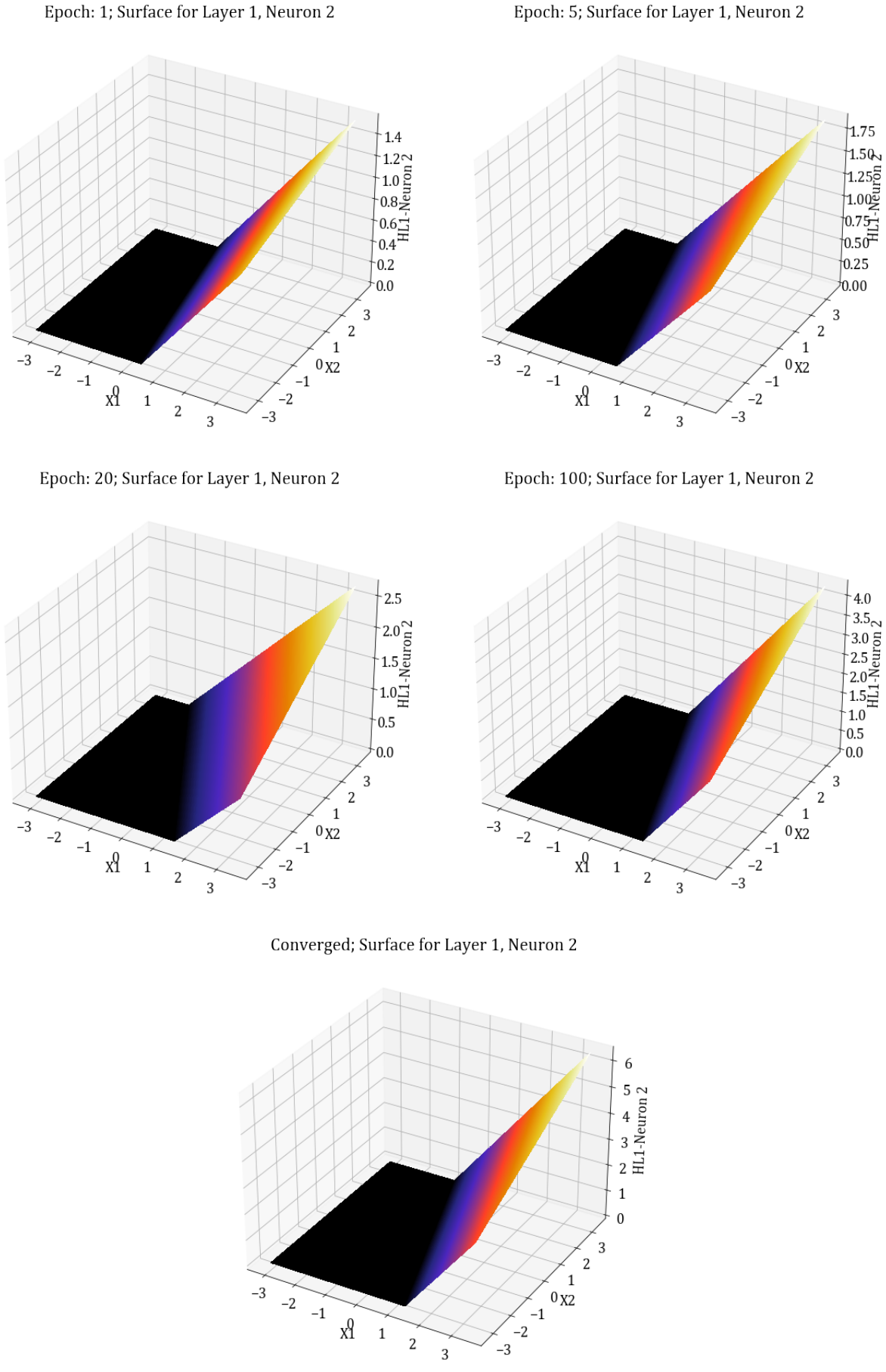


Figure 34: Surface Plots obtained for Hidden Layer 1, Neuron 2, across epochs.

2.1.4.3 Hidden Layer 1, Node 3

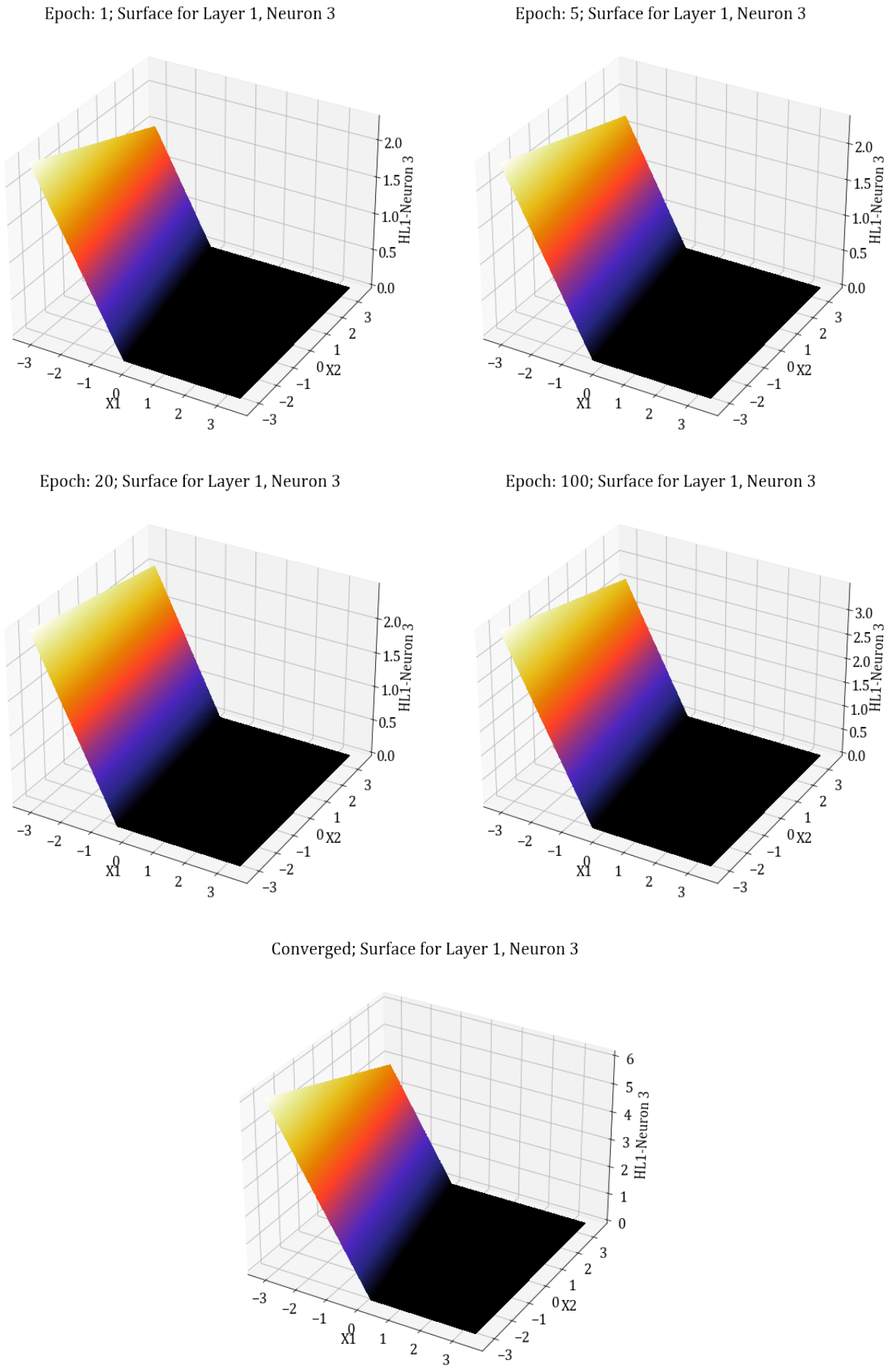


Figure 35: Surface Plots obtained for Hidden Layer 1, Neuron 3, across epochs.

2.1.4.4 Hidden Layer 1, Node 4

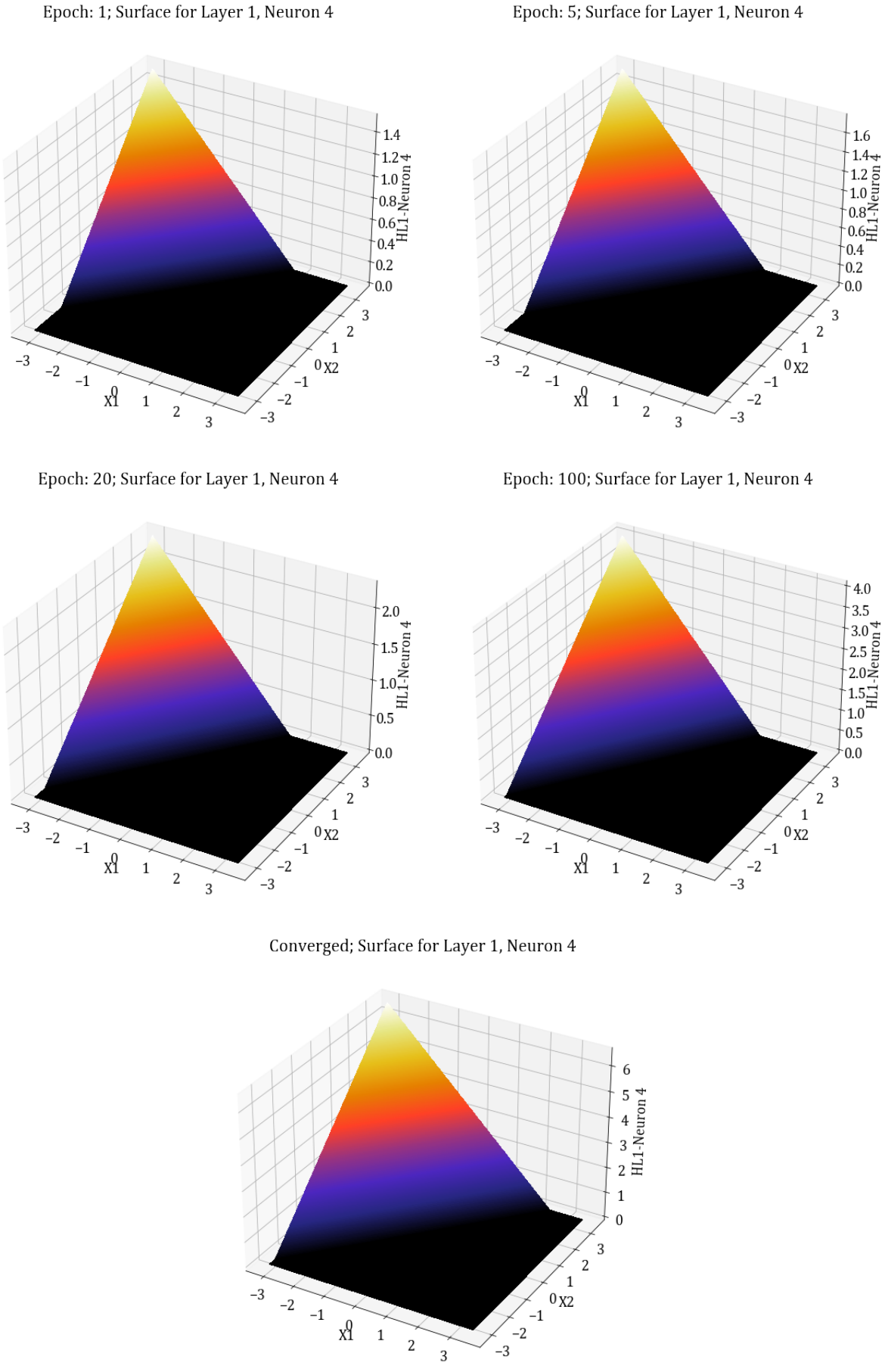


Figure 36: Surface Plots obtained for Hidden Layer 1, Neuron 4, across epochs.

2.1.4.5 Hidden Layer 1, Node 5

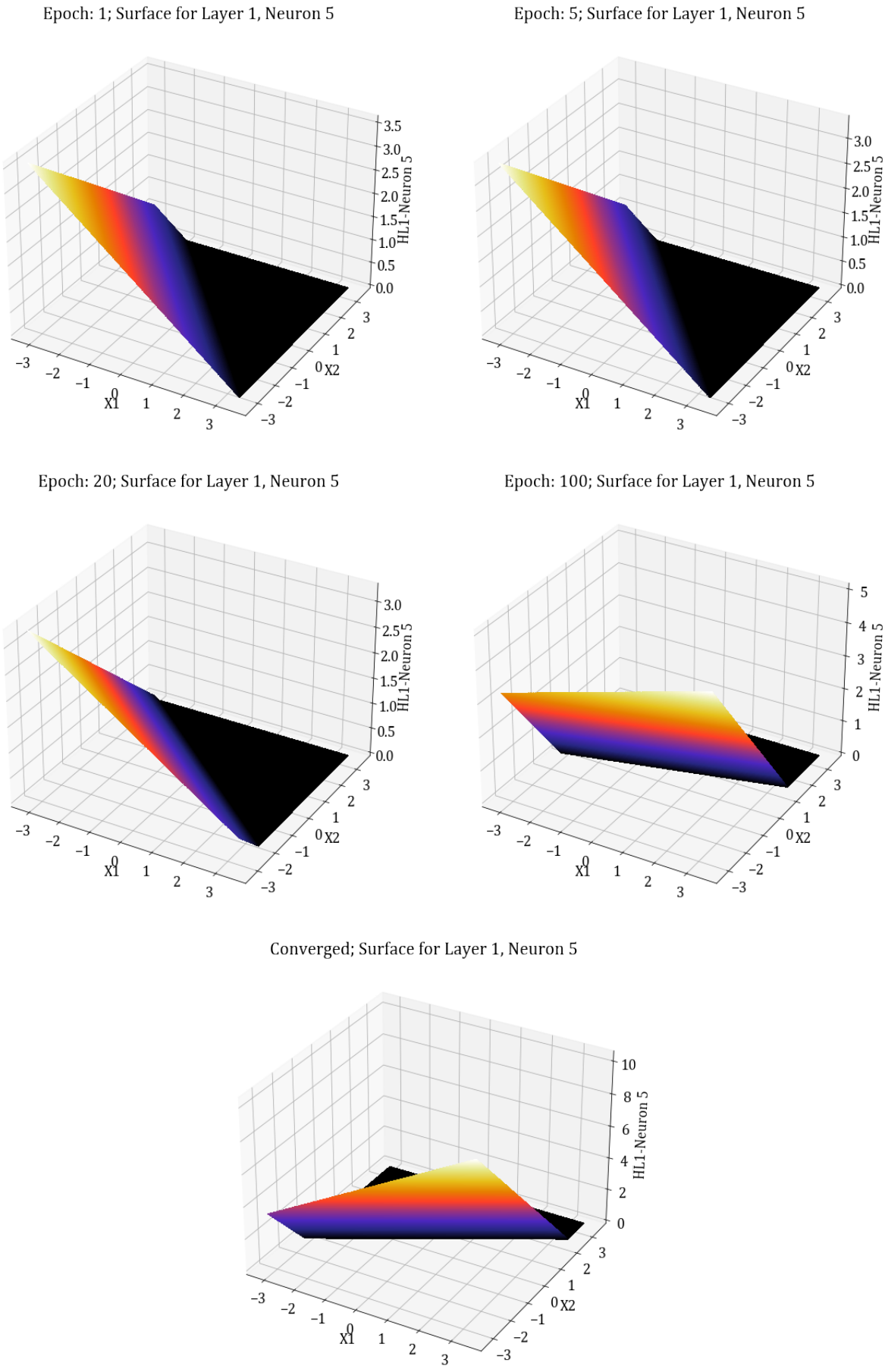


Figure 37: Surface Plots obtained for Hidden Layer 1, Neuron 5, across epochs.

2.1.4.6 Hidden Layer 1, Node 6

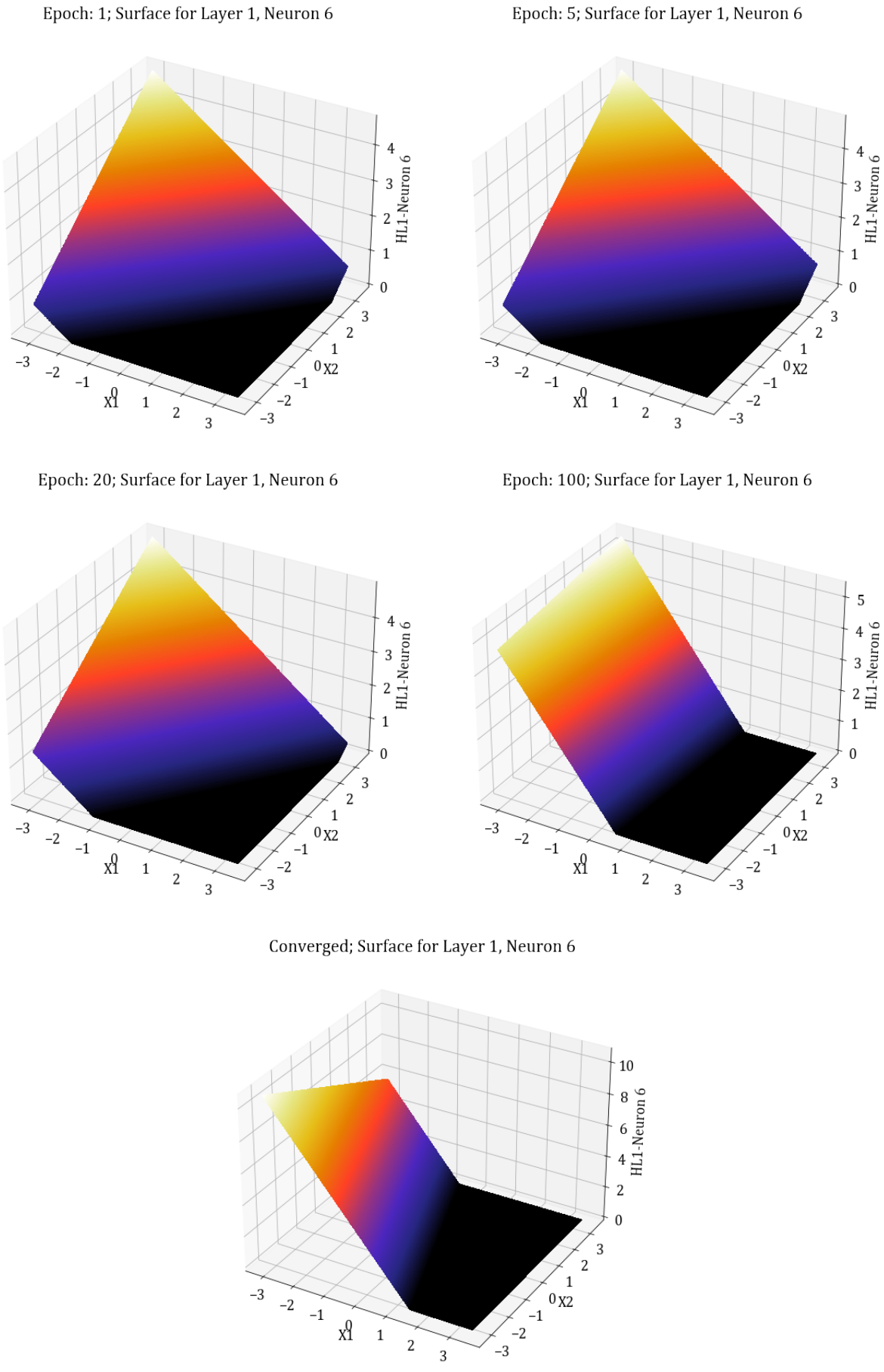


Figure 38: Surface Plots obtained for Hidden Layer 1, Neuron 6, across epochs.

2.1.4.7 Hidden Layer 1, Node 7

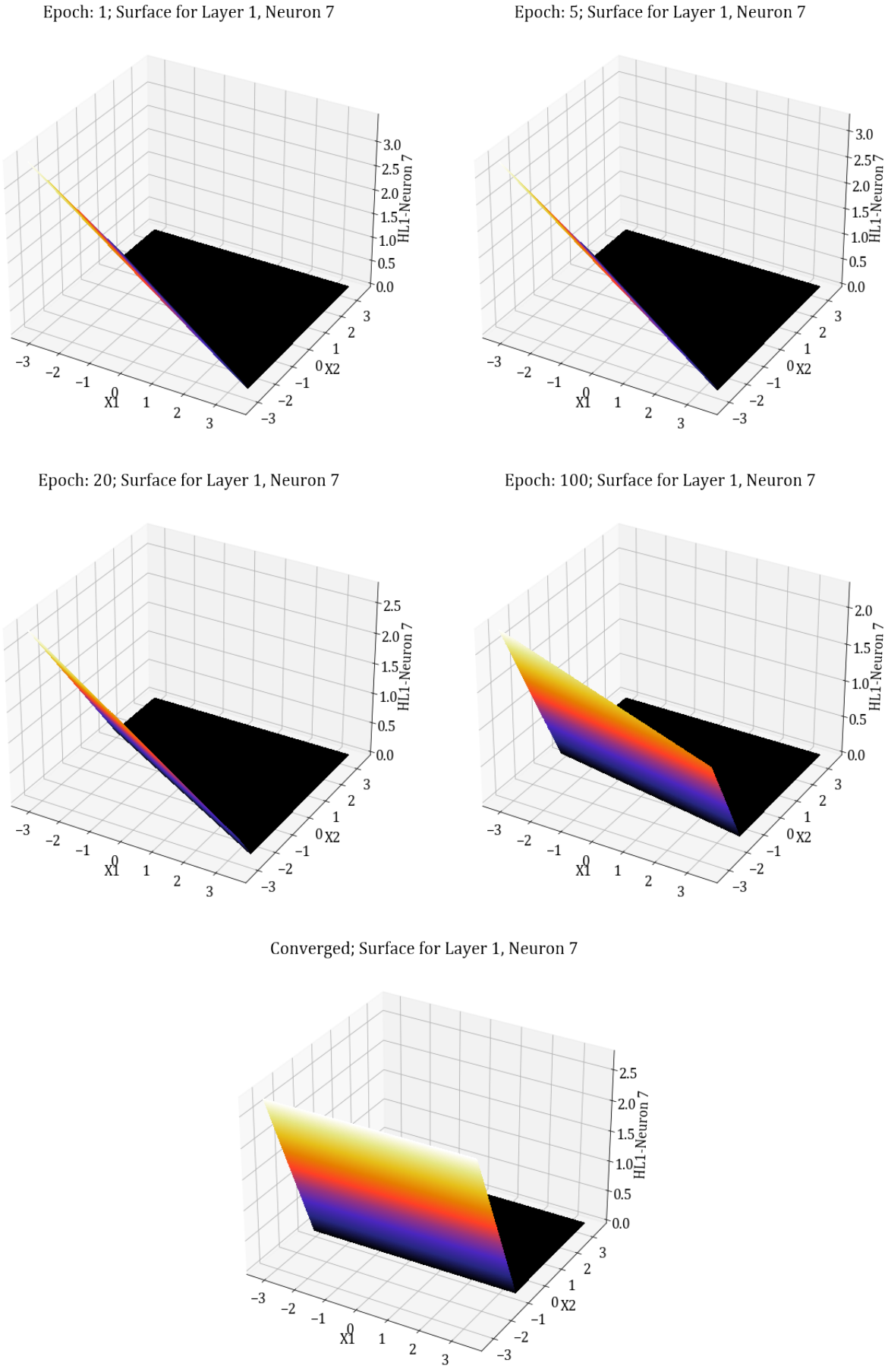


Figure 39: Surface Plots obtained for Hidden Layer 1, Neuron 7, across epochs.

2.1.4.8 Hidden Layer 1, Node 8

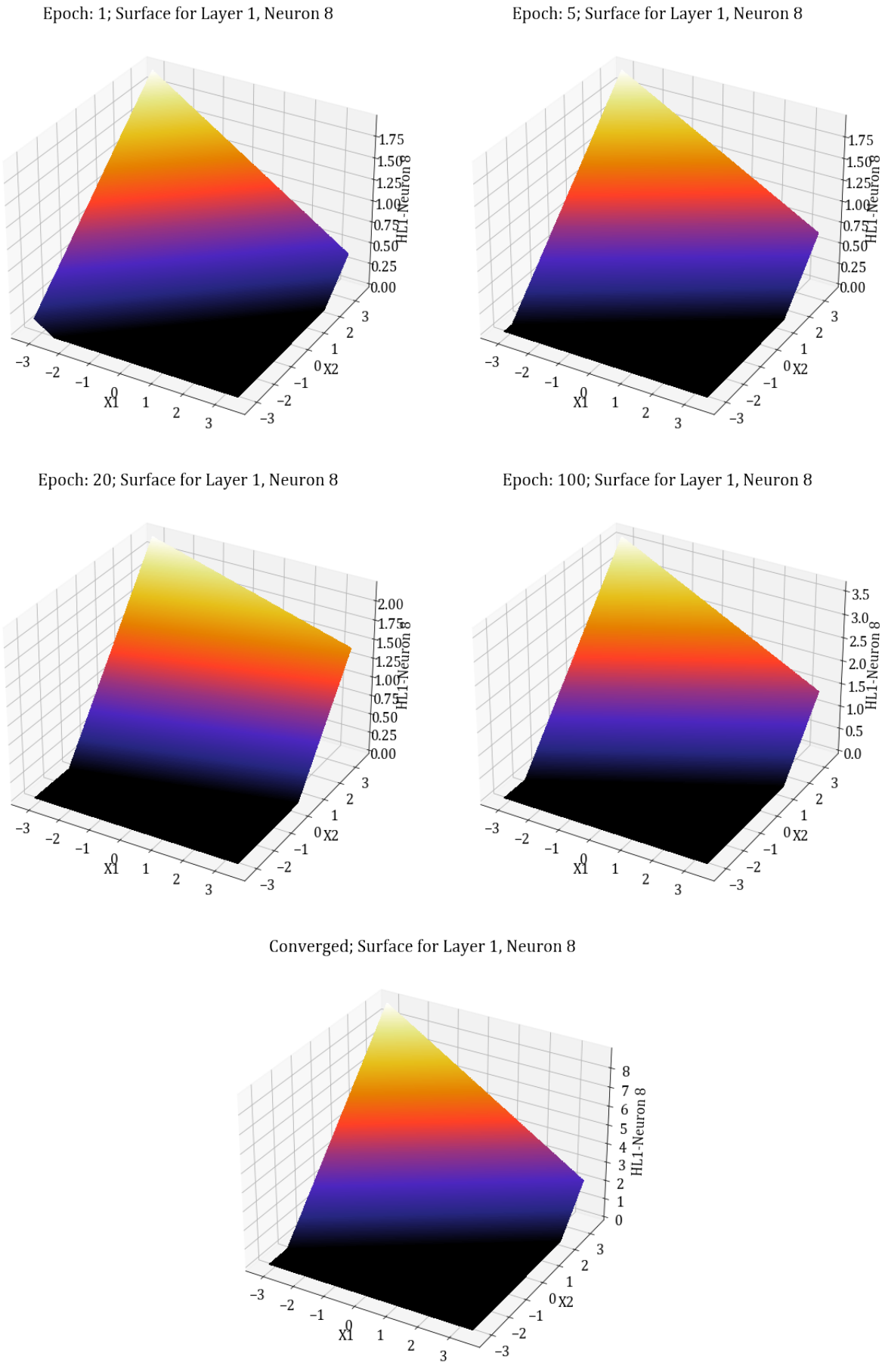


Figure 40: Surface Plots obtained for Hidden Layer 1, Neuron 8, across epochs.

2.1.4.9 Hidden Layer 2, Node 1

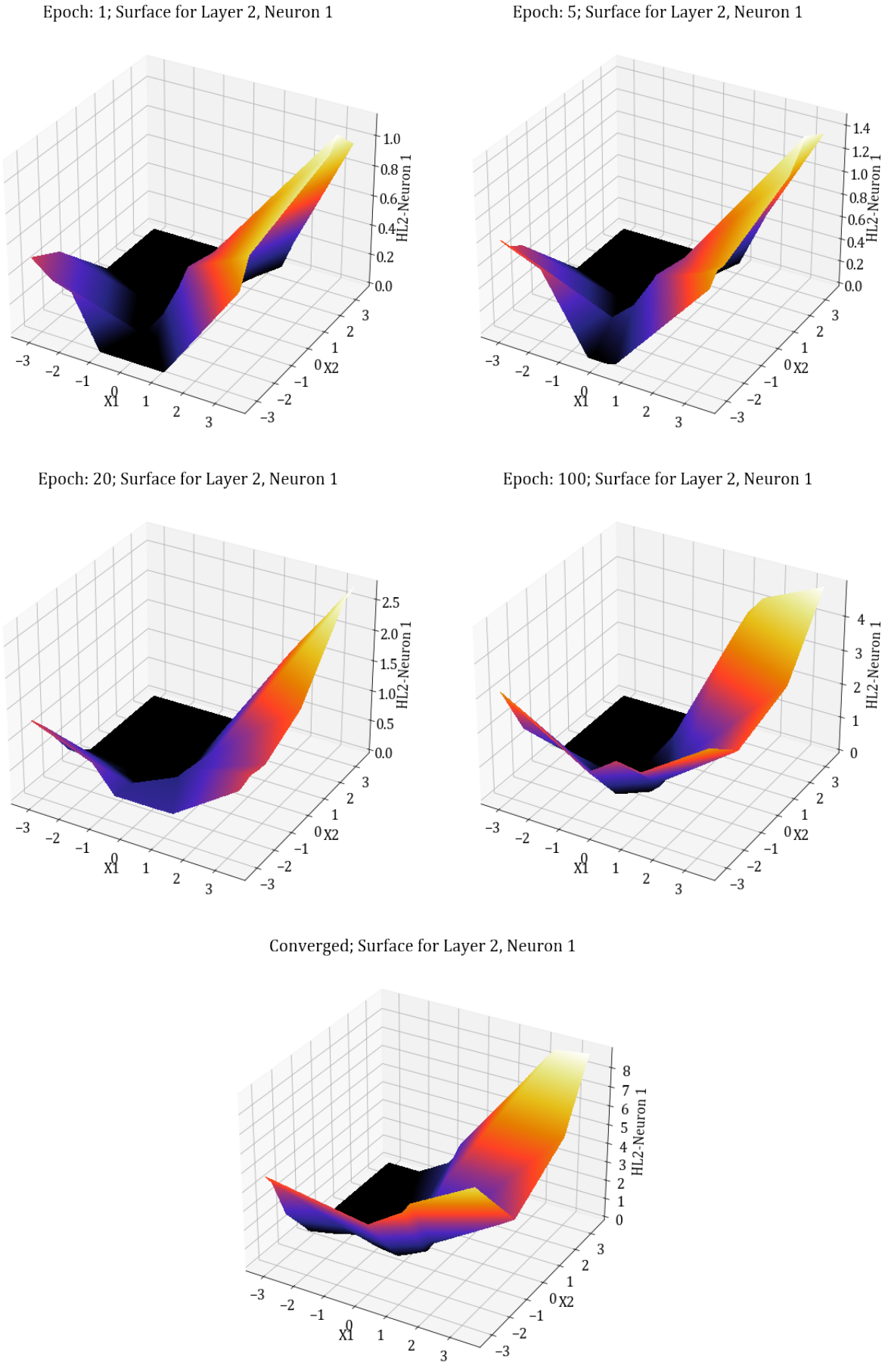


Figure 41: Surface Plots obtained for Hidden Layer 2, Neuron 1, across epochs.

2.1.4.10 Hidden Layer 2, Node 2

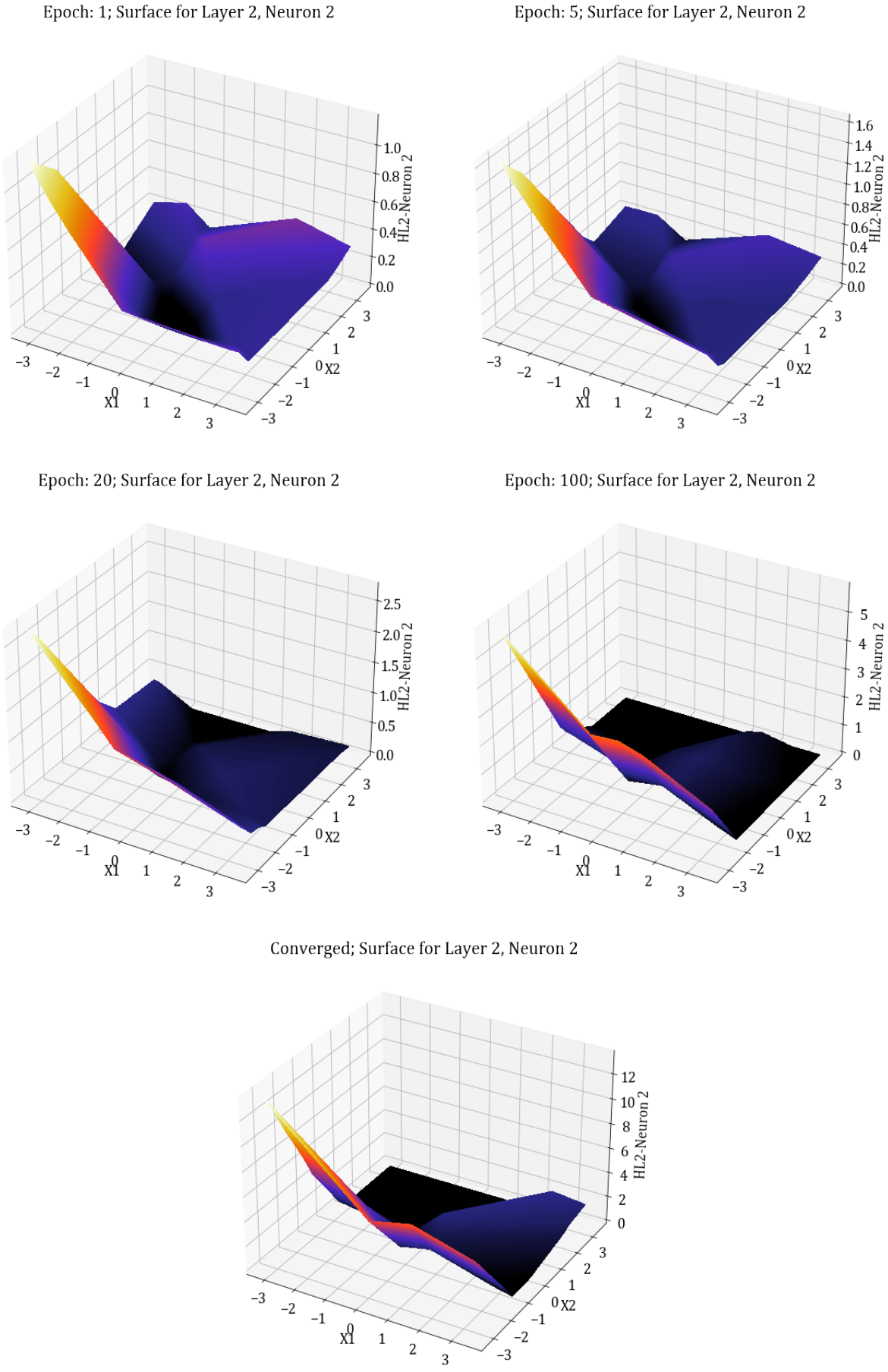


Figure 42: Surface Plots obtained for Hidden Layer 2, Neuron 2, across epochs.

2.1.4.11 Hidden Layer 2, Node 3

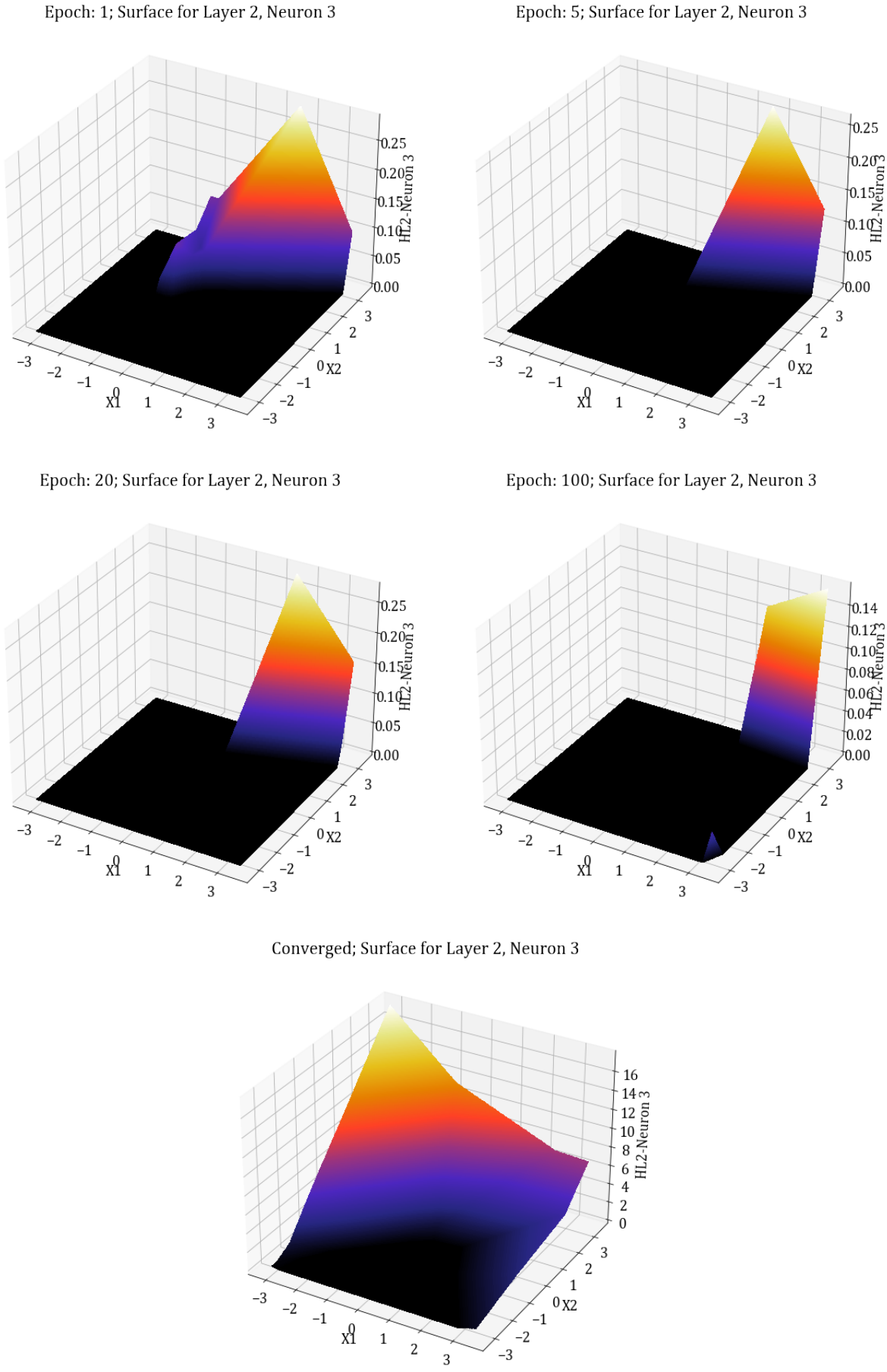


Figure 43: Surface Plots obtained for Hidden Layer 2, Neuron 3, across epochs.

2.1.4.12 Hidden Layer 2, Node 4

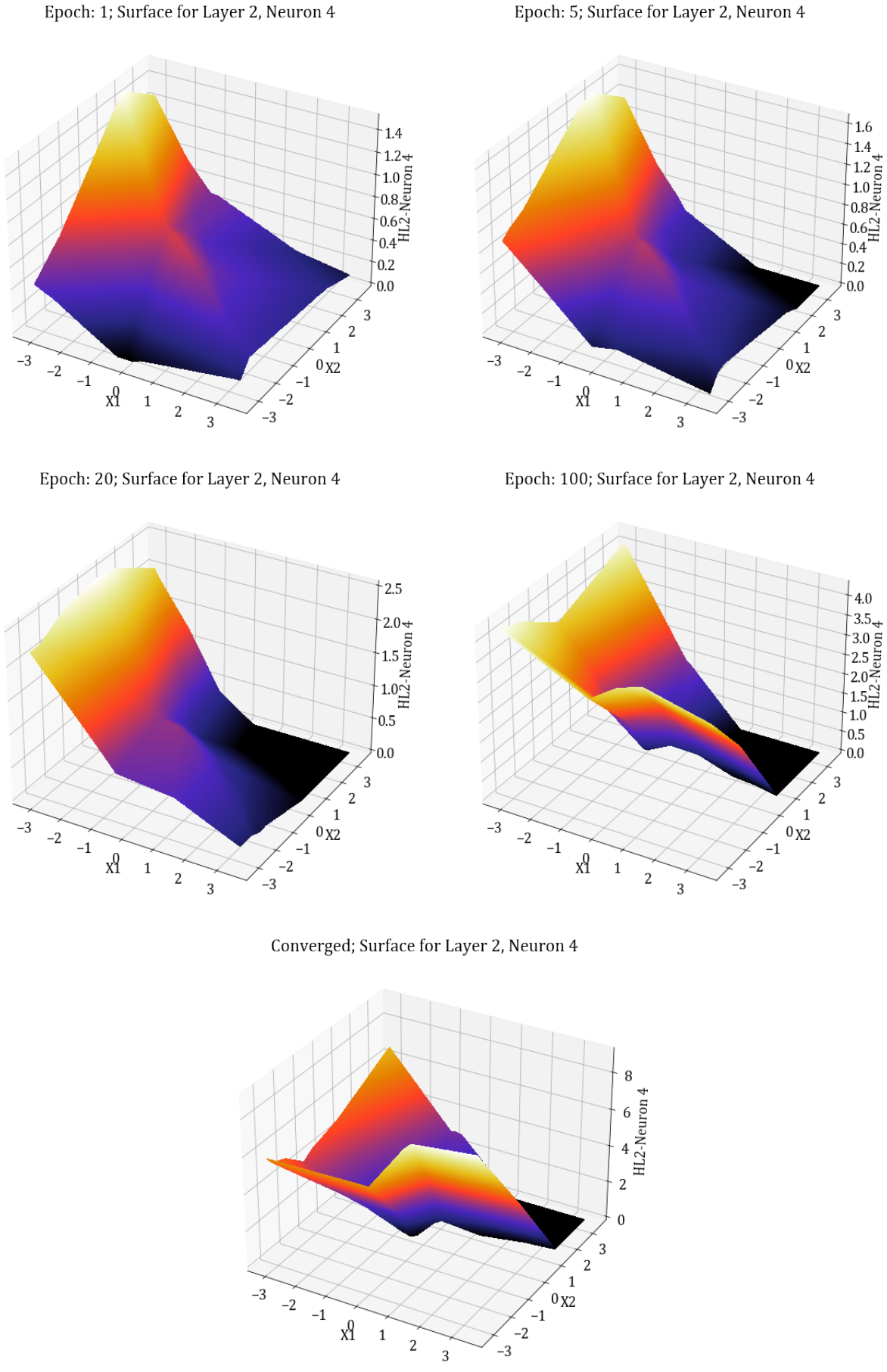


Figure 44: Surface Plots obtained for Hidden Layer 2, Neuron 4, across epochs.

2.1.4.13 Hidden Layer 2, Node 5

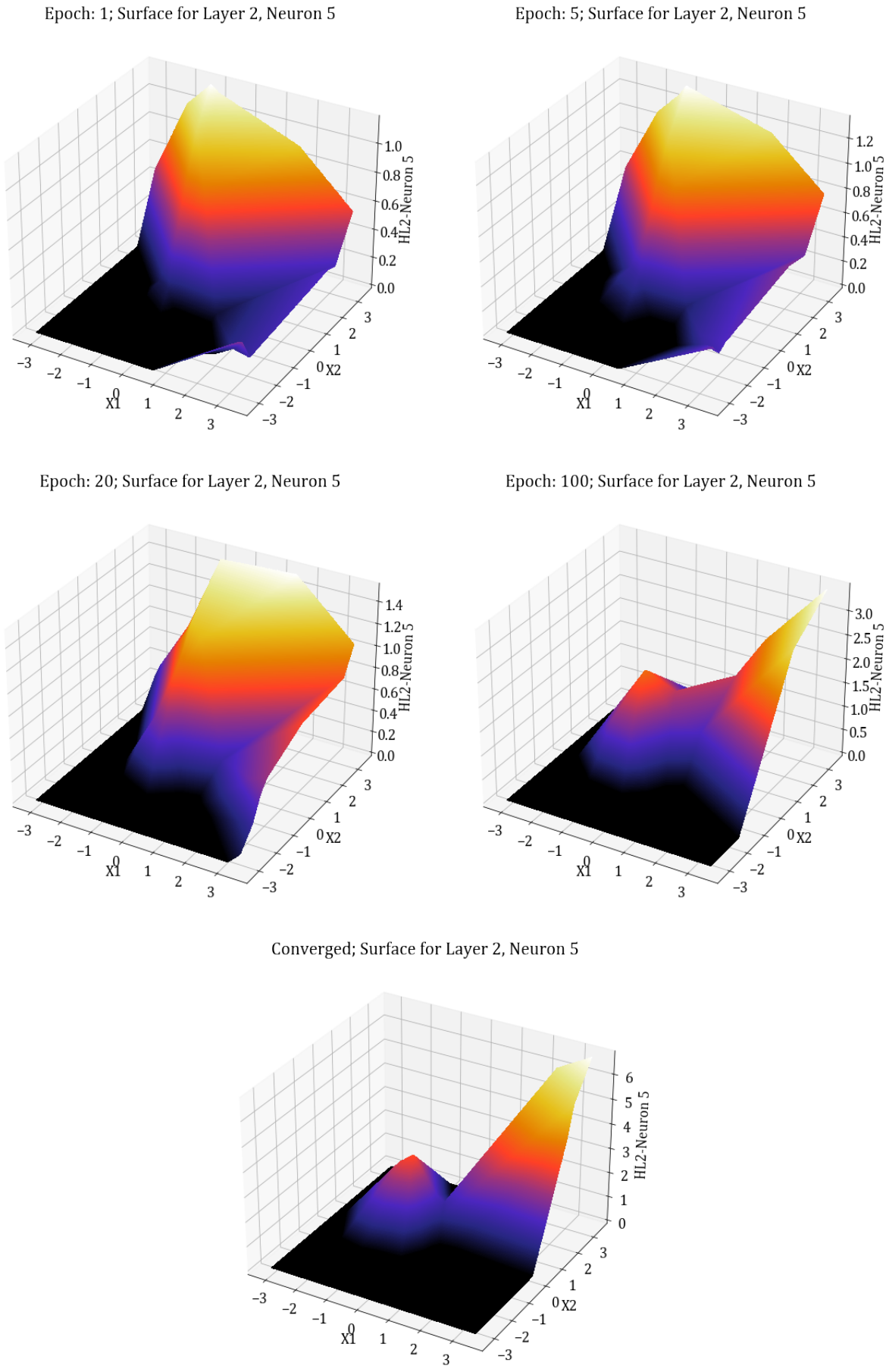


Figure 45: Surface Plots obtained for Hidden Layer 2, Neuron 5, across epochs.

2.1.4.14 Hidden Layer 2, Node 6

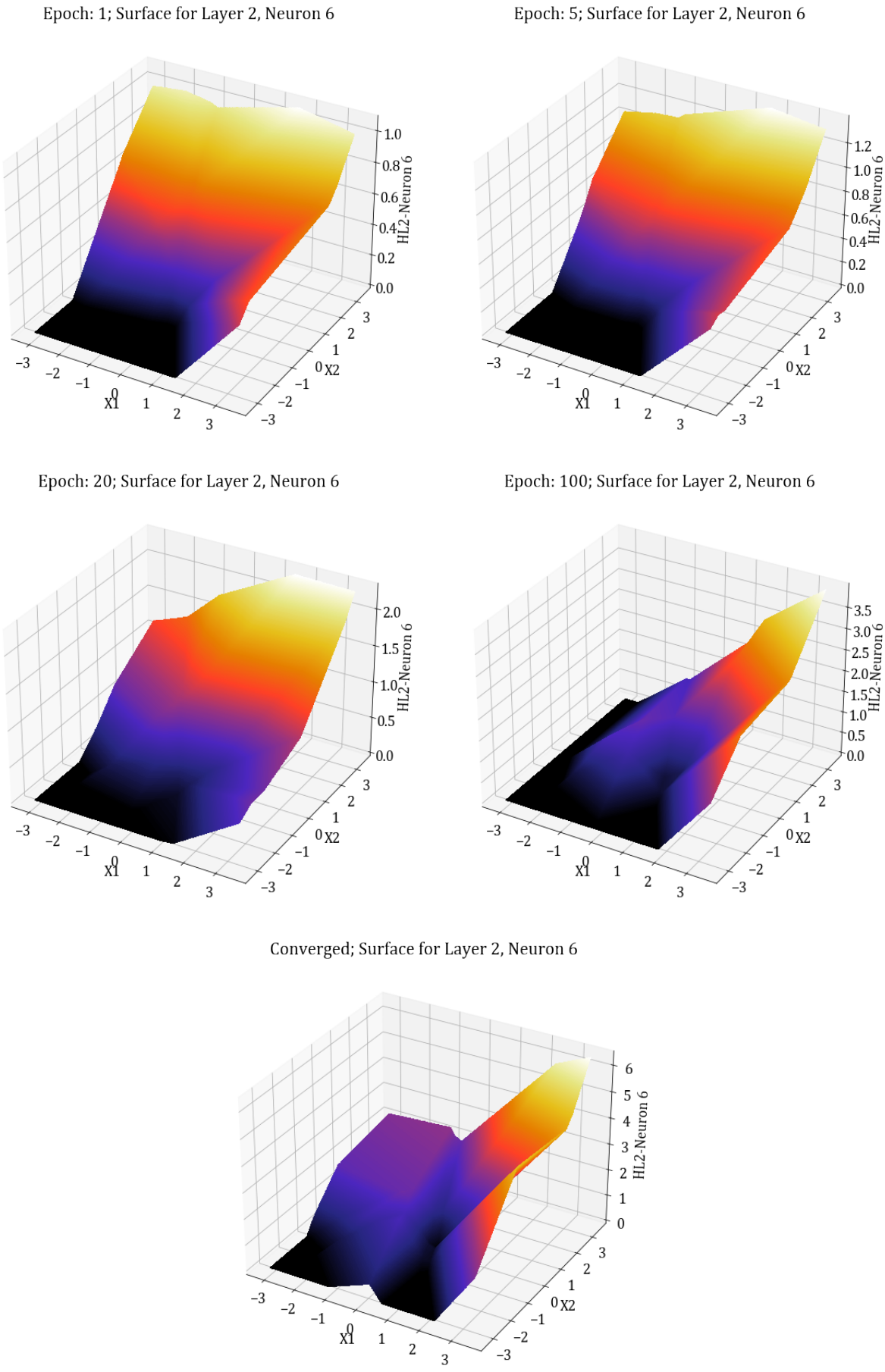


Figure 46: Surface Plots obtained for Hidden Layer 2, Neuron 6, across epochs.

2.1.4.15 Hidden Layer 2, Node 7

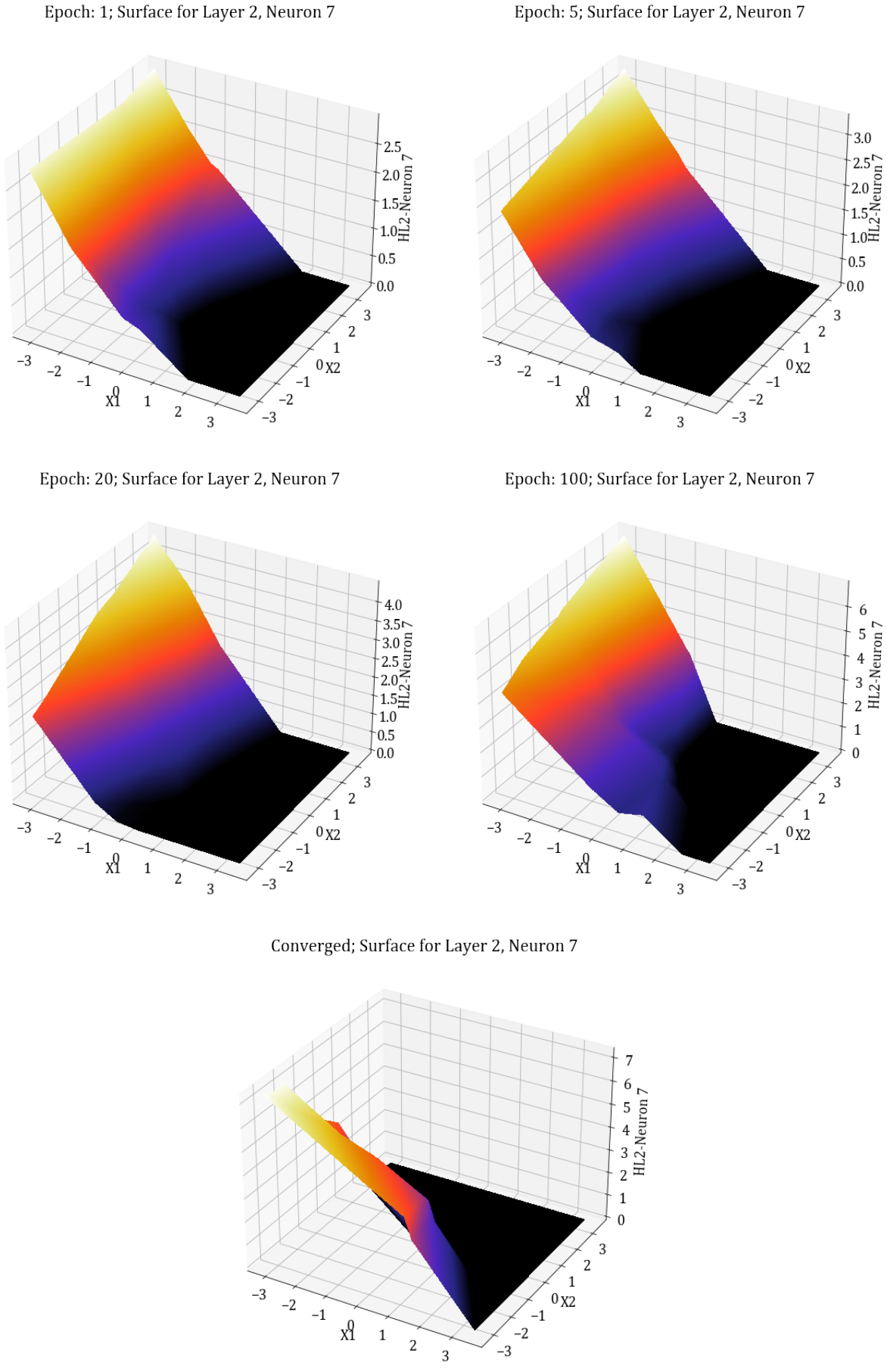


Figure 47: Surface Plots obtained for Hidden Layer 2, Neuron 7, across epochs.

2.1.4.16 Hidden Layer 2, Node 8

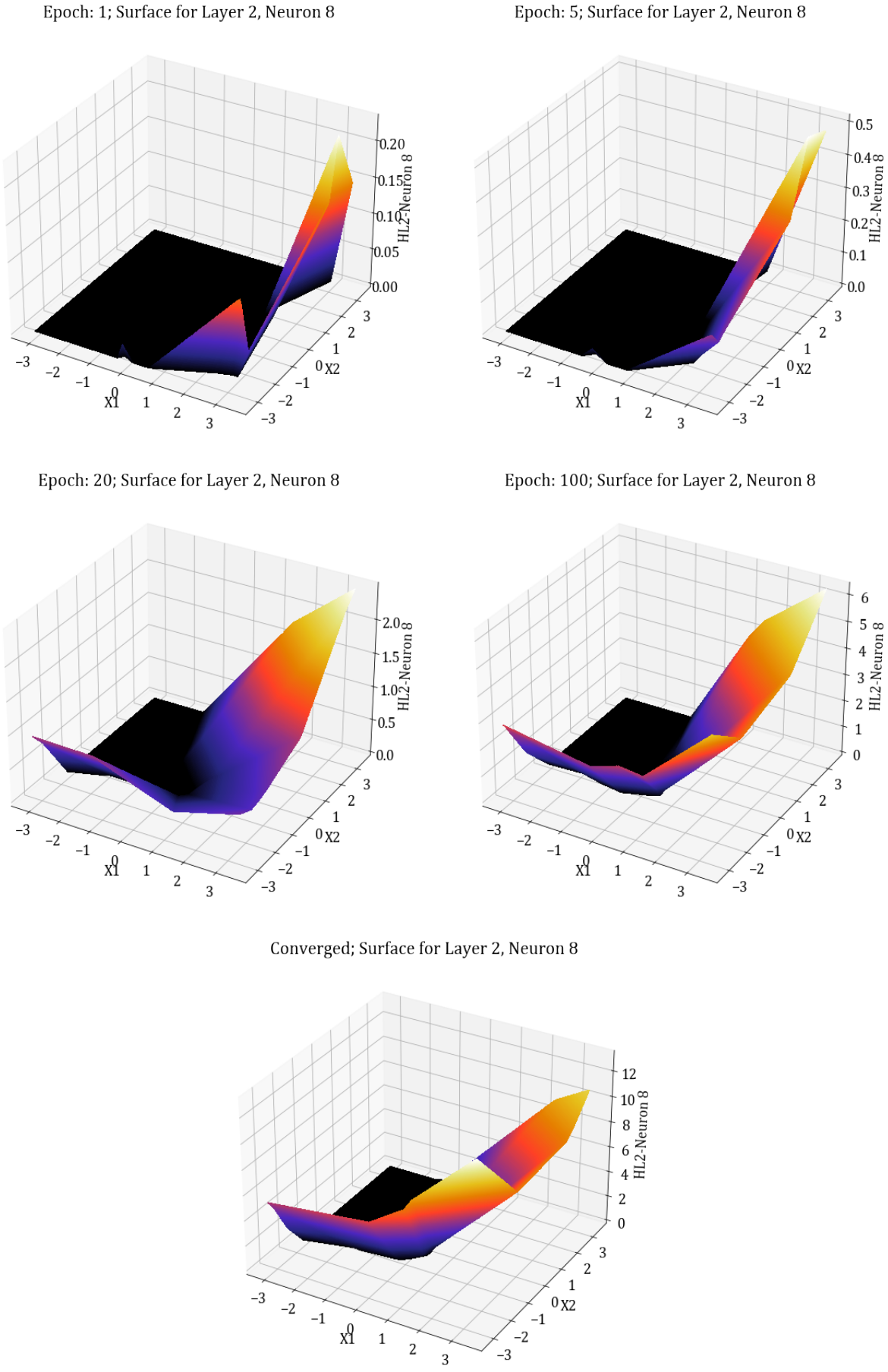


Figure 48: Surface Plots obtained for Hidden Layer 2, Neuron 8, across epochs.

2.1.4.17 Output Layer, Node 1

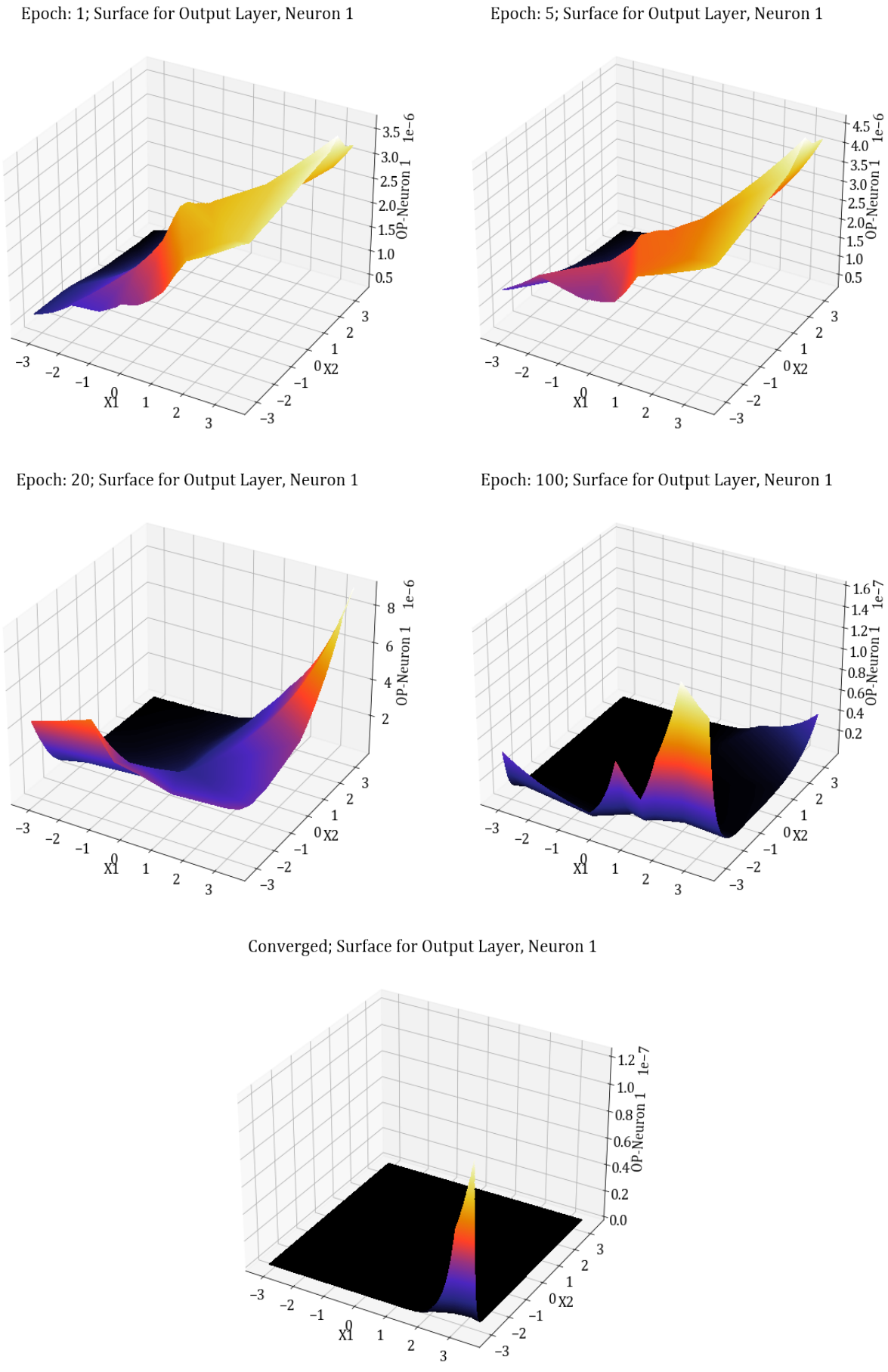


Figure 49: Surface Plots obtained for Output Layer, Neuron 1, across epochs.

2.1.4.18 Output Layer, Node 2

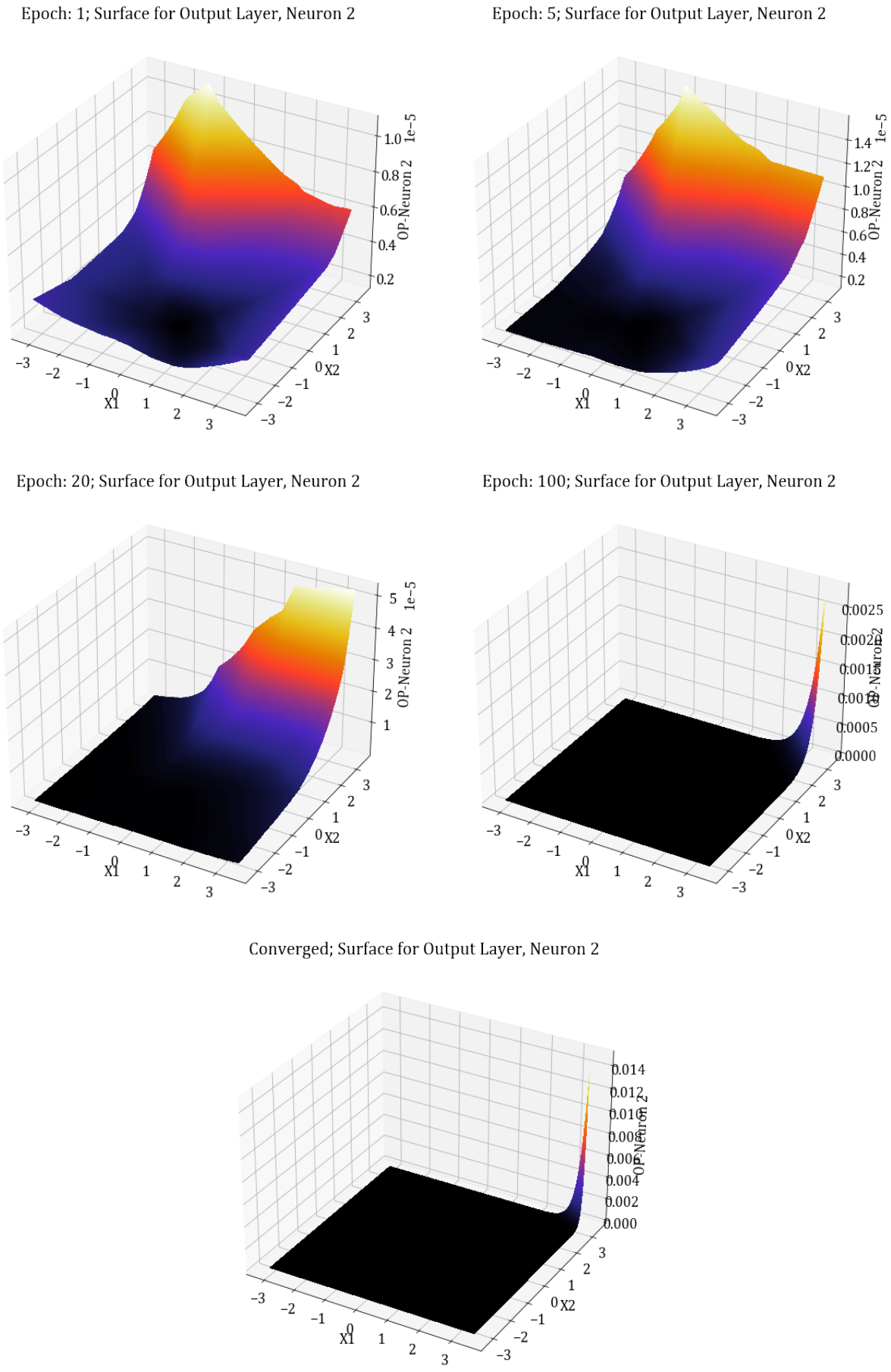


Figure 50: Surface Plots obtained for Output Layer, Neuron 2, across epochs.

2.1.4.19 Output Layer, Node 3

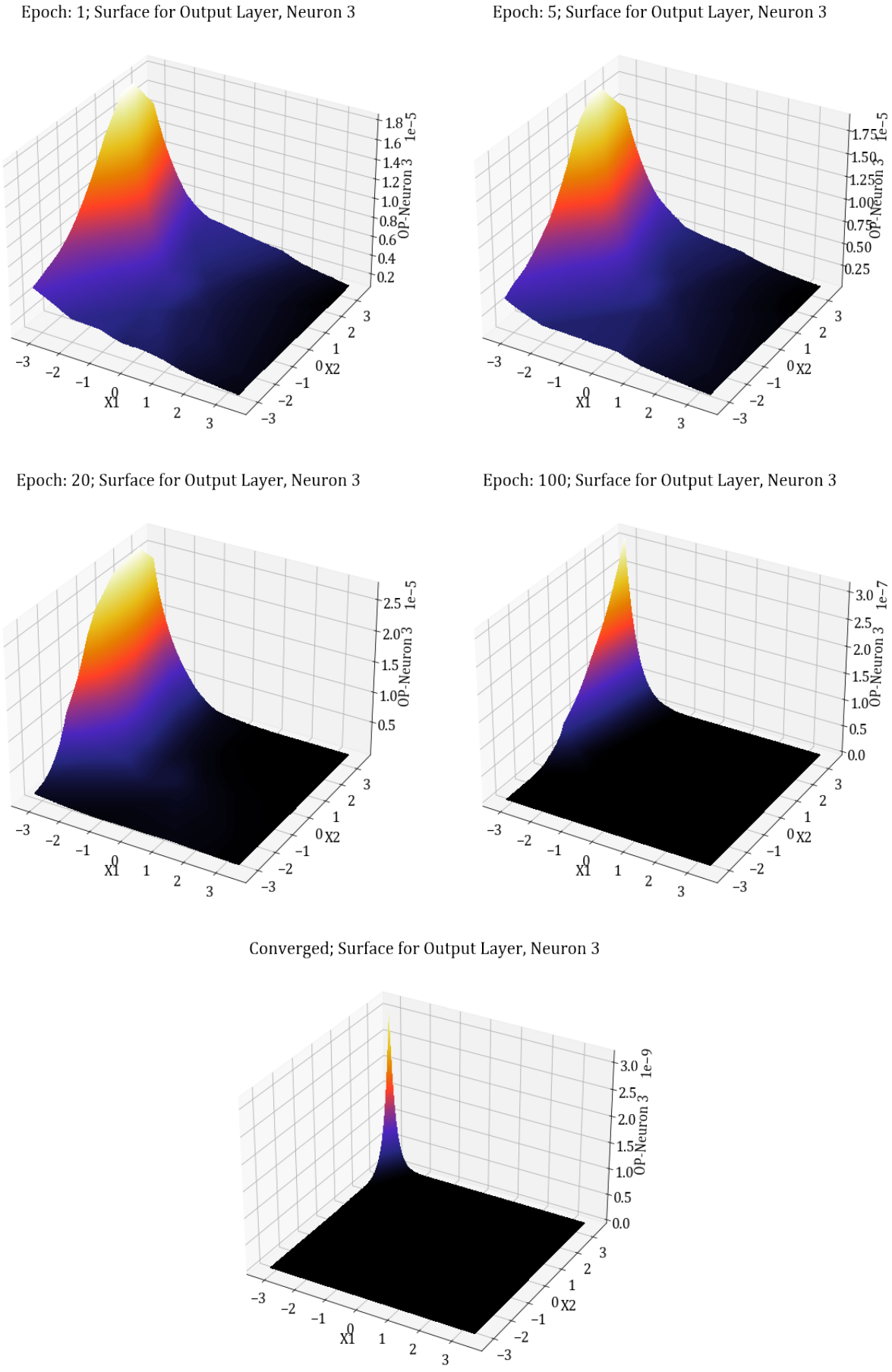


Figure 51: Surface Plots obtained for Output Layer, Neuron 3, across epochs.

From Figure 33-Figure 51, we observe the following:

- First hidden layer surface plot is non-linear (activation function is `ReLU`). However, the surfaces obtained are intersection of hyperplanes.
- Responses from the second layer is highly non-linear and the surfaces are very complex.
- The surface plot of the output neurons shows the selection cum localization of different classes in the latent space.

2.2 Non-Linear SVM

The module `sklearn.svm.SVC()` as described in section subsubsection 1.3.2 is used to build non-linear classifiers for dataset 1B with the parameter `decision_function_shape` set to `ovr`. The module can carry out multi-class classification hence individual pair-wise models need not be built. The kernels used for classification are:

- Polynomial Kernel-
  It is defined as $(\gamma < x, x' > +r)^d$. Where γ, r and d are the hyper-parameters.
- Gaussian Kernel or the Radial Basis function kernel-
  It is defined as $exp(-\gamma||x - x'||^2)$. Where γ is the hyper-parameter.

2.2.1 Polynomial Kernel

While using this kernel, a number of hyper-parameters need to be set:

- misclassification penalty term, `C`.
- Coefficient of the polynomials, γ.
- Constant term in the kernel, `r`.
- Degree of the polynomial, `d`.

Due to a large number of possible combinations of the above hyper-parameters, we use the function `GridSearchCV` from `sklearn.model_selection` to search for the optimal combination of hyper-parameters.

2.2.2 Optimal hyper-parameters and accuracy: Polynomial Kernel

The following values of hyper-parameters are tried resulting in $4 * 6 * 2 * 4 = 192$ different models.

- `C`: [1,10,100,1000]
- `degree`: [1,2,3,4,5,6]
- `coef0`: [10,100]
- γ: [1,0.1,0.01,"auto"]

The function `GridSearchCV` returns the validation and train accuracy for all the models in the form of a numpy array. Accuracy table for the 10 models with the highest validation accuracy is:

C	degree	gamma	Coef0	Train Accuracy	Validation Accuracy
1000	5	0.1	100	0.983	1.00
10	4	0.1	10	0.993	0.981
1	5	0.1	100	0.986	0.976
1	3	1	10	0.995	0.977
100	5	0.1	100	0.981	0.977
10	5	0.1	100	0.981	0.977
1	3	auto	100	0.980	0.975
1000	3	0.1	10	0.995	0.975
100	3	0.1	10	0.980	0.975
100	5	0.01	10	0.985	0.975

Table 10: Accuracy table for dataset 1B - SVM classifier with polynomial kernel

Since the validation accuracy is highest for [C:1000,degree:5,gamma:0.1,Coef0:100], those hyper-parameters are used to predict the class labels for the test dataset, obtaining an accuracy of **100%**. Those hyper-parameters are further used to obtain the confusion matrices and to plot the decision boundaries.

2.2.3 Confusion matrices for Train and Test data: Polynomial Kernel

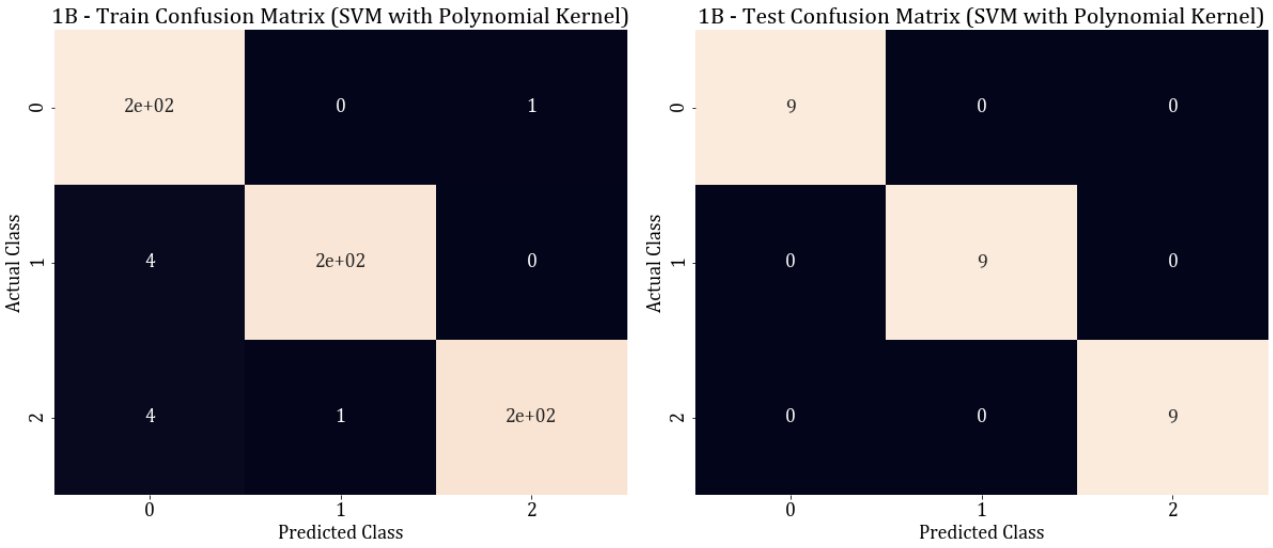


Figure 52: Confusion matrix for Train and Test data respectively

2.2.4 Decision Region: Polynomial Kernel

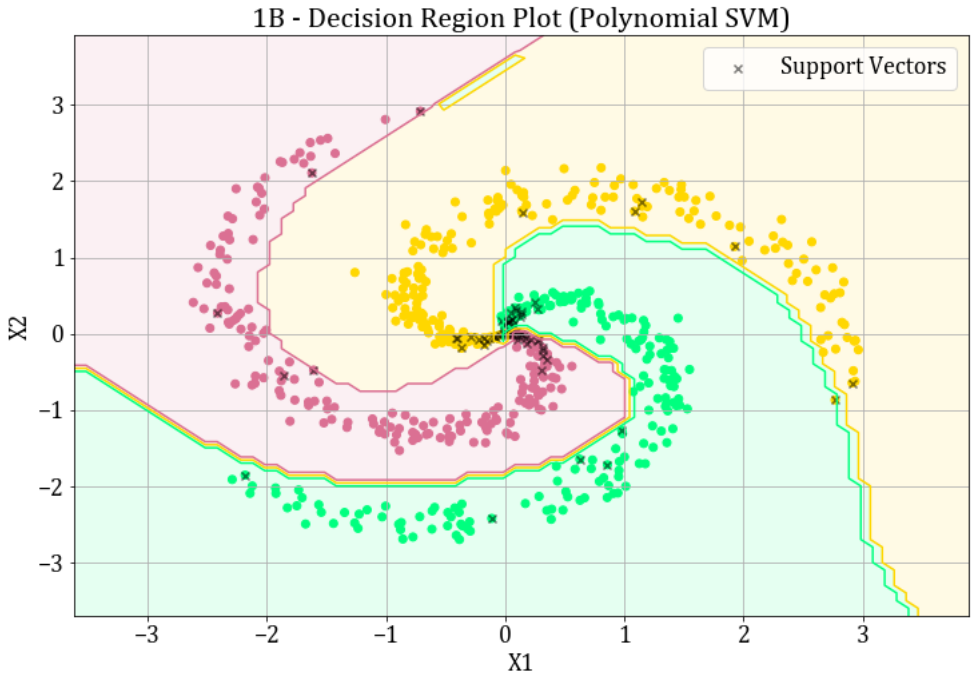


Figure 53: Decision Region Plot for dataset 1B

2.2.5 SVM with gaussian kernel

While using this kernel, the number of hyper-parameters to be set are comparatively less.

- misclassification penalty term, C; Varied over [0.1,1,10,100,1000]
- Coefficient γ, varied over: [1,0.01,0.001,0.0001]

2.2.6 Optimal hyper-parameters and accuracy: Gaussian Kernel

Corresponding to different combinations of the above hyper-parameter values, 20 different models are trained. The training accuracy and cross-validation accuracy are obtained as follows:

gamma	C	Train Accuracy	Val. Accuracy	gamma	C	Train Accuracy	Val. Accuracy
1.00	1.0	99.16	100	0.001	0.1	51.67	53.96
1.00	0.1	99.16	98.41	0.001	1.0	51.50	53.96
1.00	10.0	99.66	98.41	0.001	10.0	51.50	60.31
1.00	100.0	99.66	98.41	0.001	100.0	54.17	61.90
1.00	1000.0	99.83	96.82	0.001	1000.0	84.67	80.95
0.01	0.1	51.16	55.56	0.0001	0.1	51.50	53.97
0.01	1.0	53.17	61.90	0.0001	1.0	51.50	53.97
0.01	10.0	84.83	80.95	0.0001	10.0	51.17	53.97
0.01	100.0	86.83	80.95	0.0001	100.0	51.50	60.32
0.01	1000.0	92.16	87.30	0.0001	1000.0	50.83	60.32

Table 11: Accuracy table for data set 1B: SVM with Gaussian Kernel

The validation accuracy is highest for [C:1.0, gamma:1.0]. Hence those hyper-parameters are used for further calculations. Accuracy on test data with those hyper-parameters is obtained to be **100%**.

2.2.7 Confusion matrices for Train and Test data: Gaussian Kernel

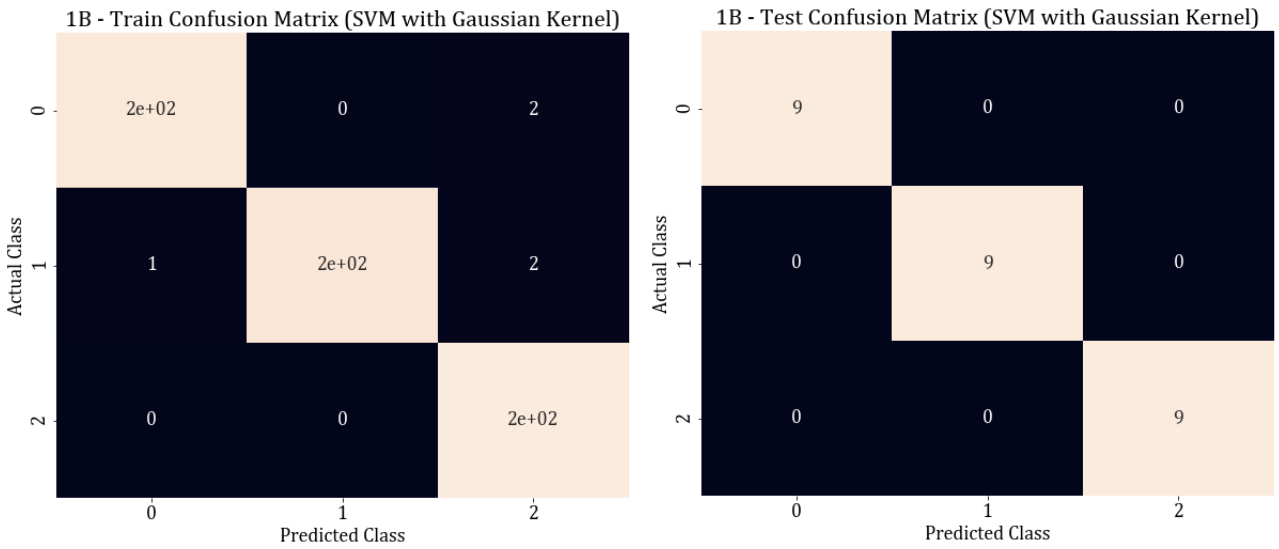


Figure 54: Confusion Matrix for Train and Test data respectively

2.2.8 Decision Region plot: Gaussian Kernel

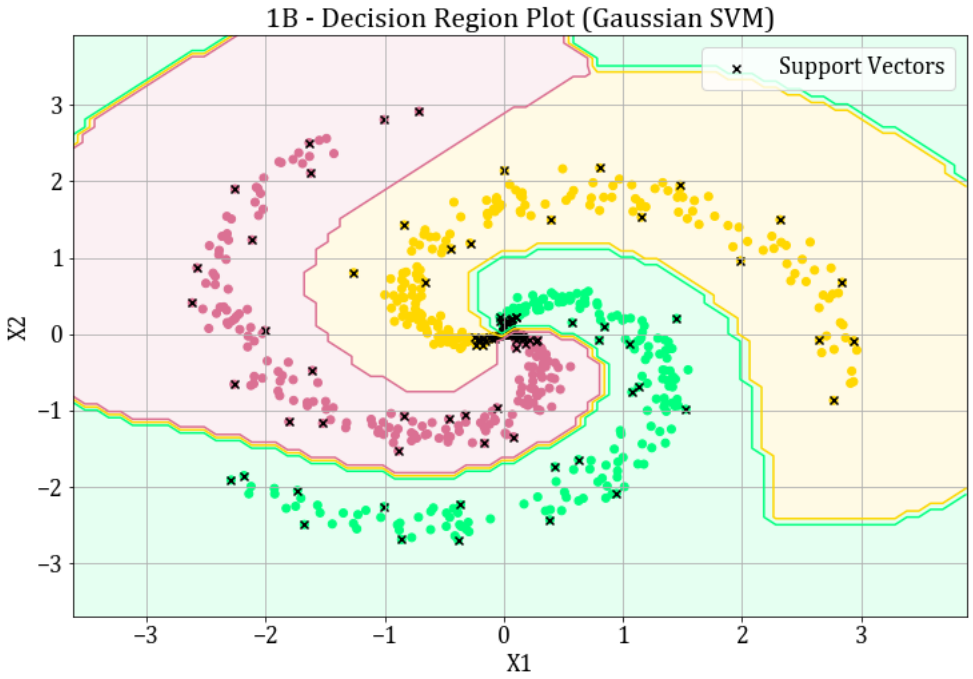


Figure 55: Decision region plot for dataset 1B: SVM with Gaussian Kernel

2.2.9 Observations

- The distinguishing feature between Figure 55 and Figure 53 is the behavior at the edges. Also, the decision boundary obtained using polynomial kernel has a discontinuity in the region $x_1 \in (-0.5, 0.5)$ and $x_2 \in (3, 4)$
- The distribution of support vectors is very distinct. In polynomial kernel, the support vectors are majorly near the center. While in the gaussian kernel, the support vectors are more uniformly distributed.

3 Dataset 2A

This dataset contains data for five classes. The classes assigned for our team are: `coast`, `highway`, `mountain`, `opencountry` and `tallbuilding`. These classes were assigned the class label 0, 1, 2, 3 and 4 respectively. The classes are non-linearly separable and the dimension of the feature space is 24.

3.1 MLFFNN

An initial parameter search by varying the parameters of the MLFFNN alone was done. However, the accuracy of the model couldn't be improved. In order to increase the accuracy of the model, PCA was used to reduce the dimensionality of the model.

The hyperparameters varied and sweeped for are - `PCA # components` `MLFFNN hidden layer size`, `MLFFNN batch size`, `MLFFNN learning rate` and `MLFFNN L2 regularization` α. They were varied as follows:

- `pca__n_components`: `list(range(1,25))`
- `mlp__hidden_layer_sizes`: (10,10), (25,25), (50,50), (75,75)
- `mlp__batch_size`: 50, 100, 200
- `mlp__alpha`: 0.01, 0.001
- `mlp__learning_rate`: `constant`, `adaptive`, `invscaling`

Hence, a total of 1728 parameter combinations were sweeped for and analyzed.

3.1.1 Classification Accuracies

The classification accuracies on the training and validation datasets (30% of the `dev.csv`) are as follows:

# Components	# Neurons	Learning Rate	Batch Size	α	Accuracy	Validation Accuracy
16	(50, 50)	invscaling	auto	0.001	79.02	52.23
16	(50, 50)	adaptive	auto	0.001	79.02	52.23
16	(50, 50)	constant	auto	0.001	79.02	52.23
20	(50, 50)	constant	auto	0.010	82.93	51.82
20	(50, 50)	adaptive	auto	0.010	82.93	51.82
20	(50, 50)	invscaling	auto	0.010	82.93	51.82
8	(75, 75)	invscaling	100	0.010	75.45	51.82
19	(25, 25)	adaptive	auto	0.001	68.05	51.42
19	(25, 25)	constant	auto	0.001	68.05	51.42
19	(25, 25)	invscaling	auto	0.001	68.05	51.42

Table 12: Best 10 Train and Validation Accuracies obtained after performing a `GridSearch` on 1728 parameter combinations.

3.1.2 Best Model

The parameter combination were additionally sorted based on minimum fitting time (least fitting time - first) and the model that gave the best accuracy the fastest (and potentially the most minimal model that best fits the data), was chosen. Hence the best parameter combination chosen is:

- n_components: 16
- hidden_layer_sizes: (50, 50)
- batch_size: 200
- alpha: 0.001
- learning_rate: invscaling

The classification accuracy of the best model on the testing data is: **49.06%**.

The confusion matrices obtained are as follows:

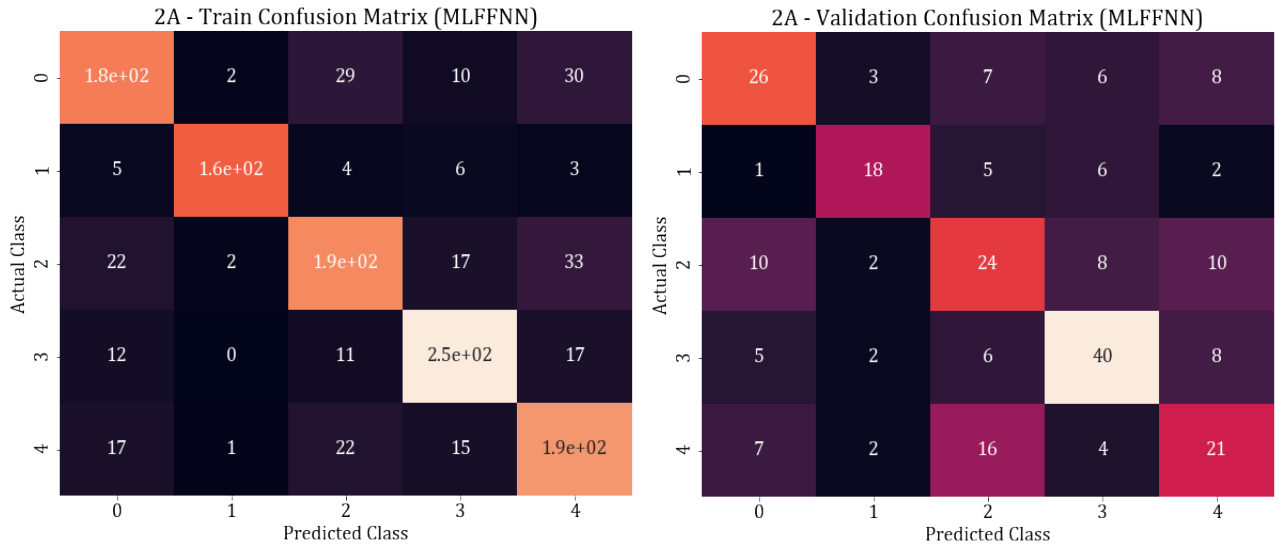


Figure 56: Training and Validation confusion matrices obtained for the best parameter combination, on the left and right respectively.

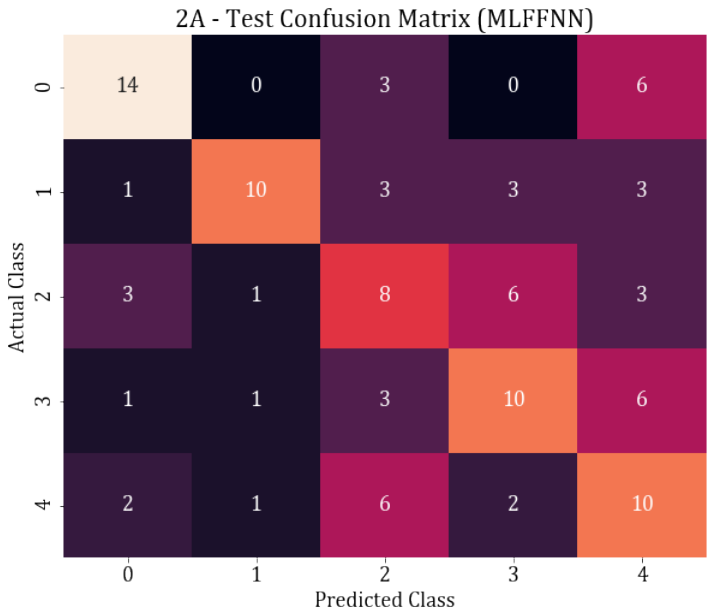


Figure 57: Testing confusion matrices obtained for the best parameter combination.

3.2 Gaussian-kernel SVM

Similar to the section 2.2.5, an svm classifier with Gaussian kernel is built. However, the accuracy obtained with varying C and gamma is quite low, therefore other parameters of the svm.SVC() function (like tolerance, class_weight etc) are also altered to check for better solutions. We find that altering those extra parameters do not affect the accuracy and so are not listed in the accuracy table.

3.2.1 Optimal Hyper-parameter and accuracy

Approximately 1500 models were trained, performance of few of those models is as listed below (in order of decreasing validation accuracy)

C	Gamma	Train Accuracy	Validation Accuracy
10	1	0.80	0.51
1000	0.1	0.79	0.51
1000	scale	1.00	0.49
100	auto	0.59	0.45
100	0.01	0.55	0.43
0.1	scale	0.51	0.41
1	0.1	0.47	0.39

Table 13: Accuracy table for data set 2a- SVM with gaussian kernel

The best hyper-parameter settings are: [c=1.0, gamma=1.0]. Test accuracy with those hyper-parameters is obtained to be **0.547**

3.2.2 Confusion matrix: Gaussian kernel

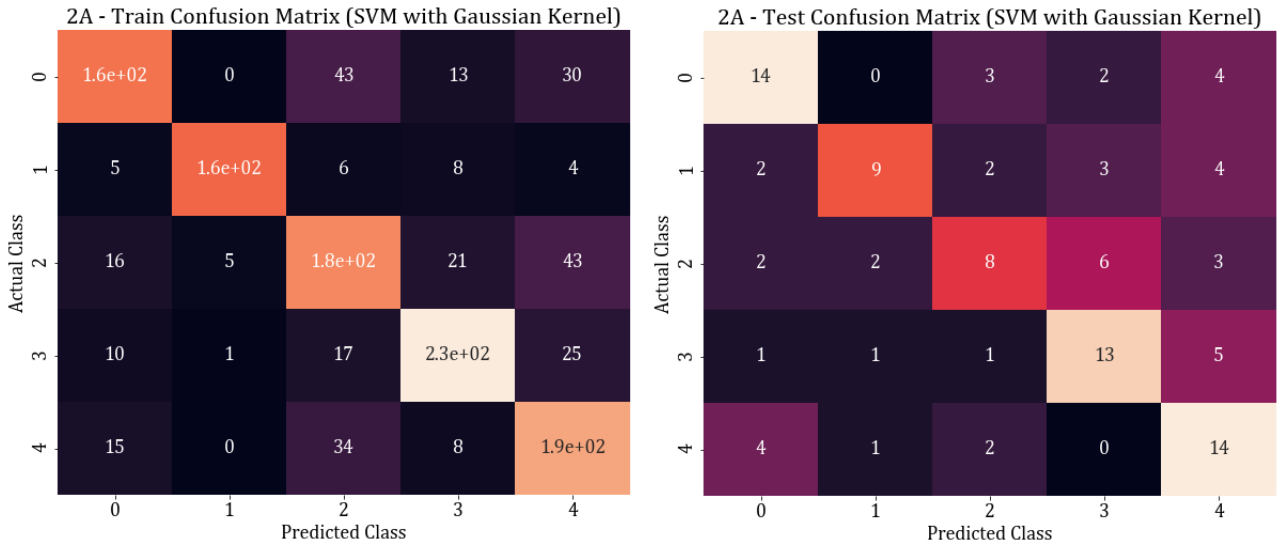


Figure 58: Confusion matrix for Train and Test data respectively, dataset 2a

3.2.3 Conclusion

The accuracy obtained with SVM classifier with a gaussian kernel is low. However, it does not improve on carrying out dimension reduction or scaling of the feature vectors. Moreover, the covariance between the feature vectors is very low,below 0.2 for majority while the covariance is maximum at 0.64 for few pairs of feature vectors.