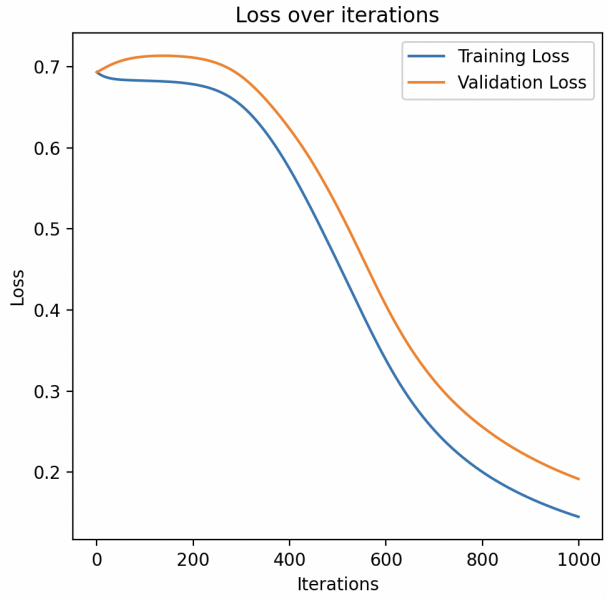
Deep Learning - Home Work 1

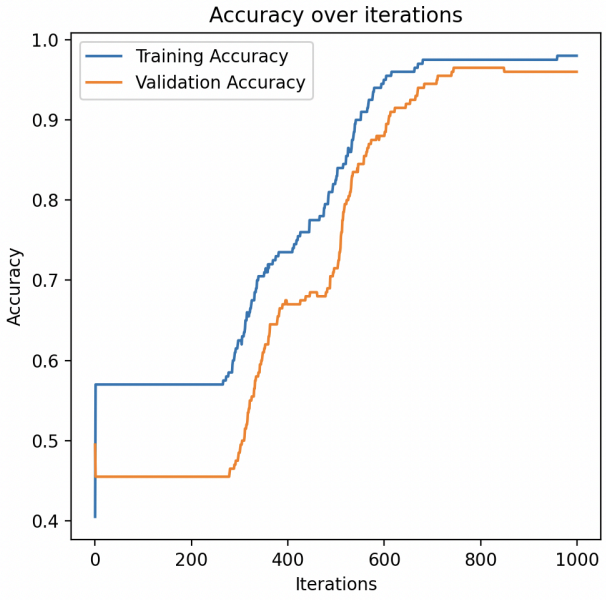
Naman Garg, Aaryansh Sahay, Rohan Gupta

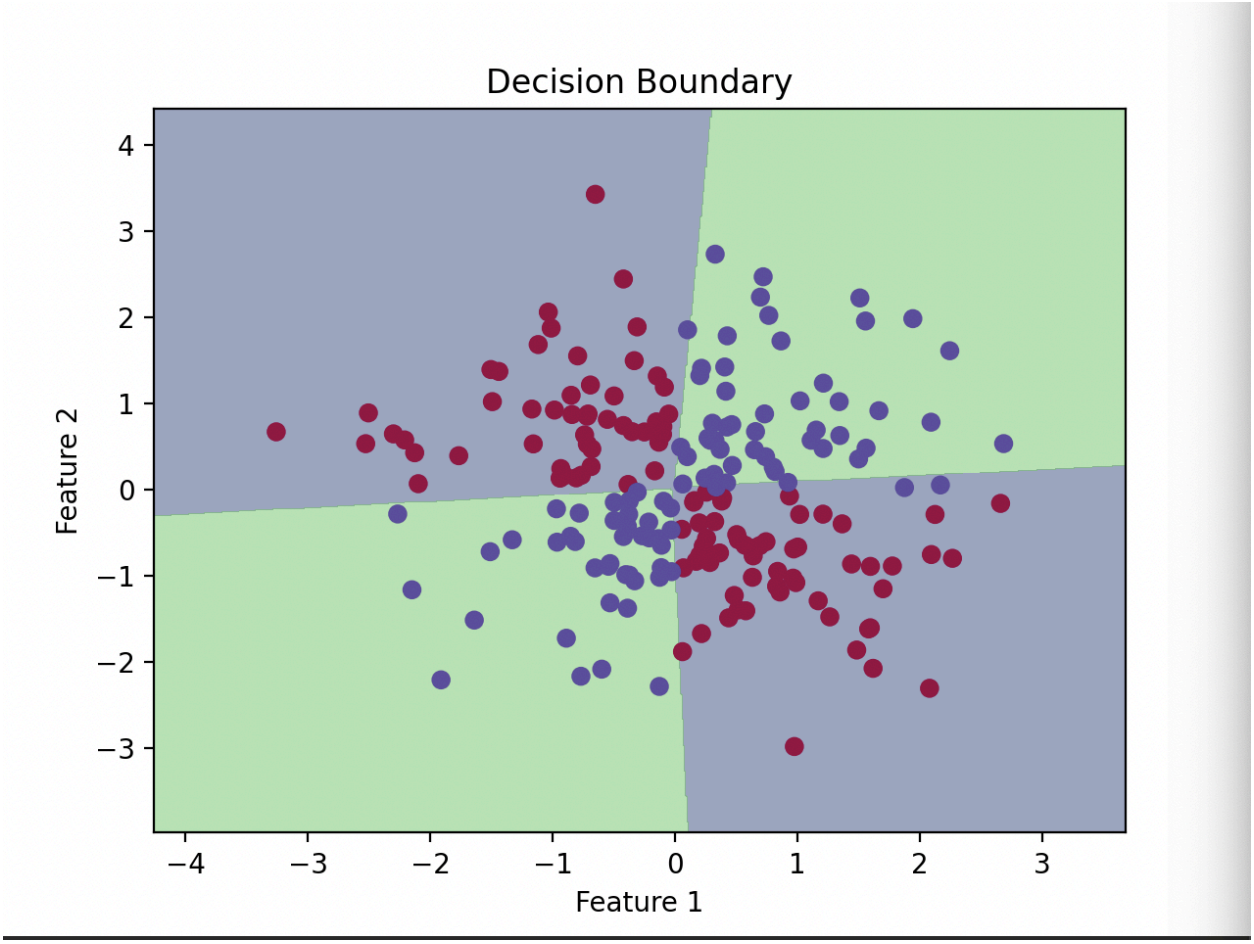
1. (4.0 points) Implement and train an MLP network as specified above using binary cross entropy as the loss function. Experiment with each of the four datasets to find the best number of nodes k in the hidden layer. For the best k for each dataset:
a. list hyper-parameters used in the model,
b. plot the learning curves for training and validation loss as a function of training epochs,
c. provide the final text accuracy, defined as the number of correct classifications divided by the total number of examples,
d. plot the learned decision surface along with observations from the test set, and
e. discuss any decisions or observations that you find relevant.

1a-1d:
XOR DATASET:

{'hidden_layer_size': 16, 'learning_rate': 0.1, 'iterations': 1000, 'final_val_acc': 0.96, 'final_val_loss': 0.1917573914307001, 'test_acc': 0.975}





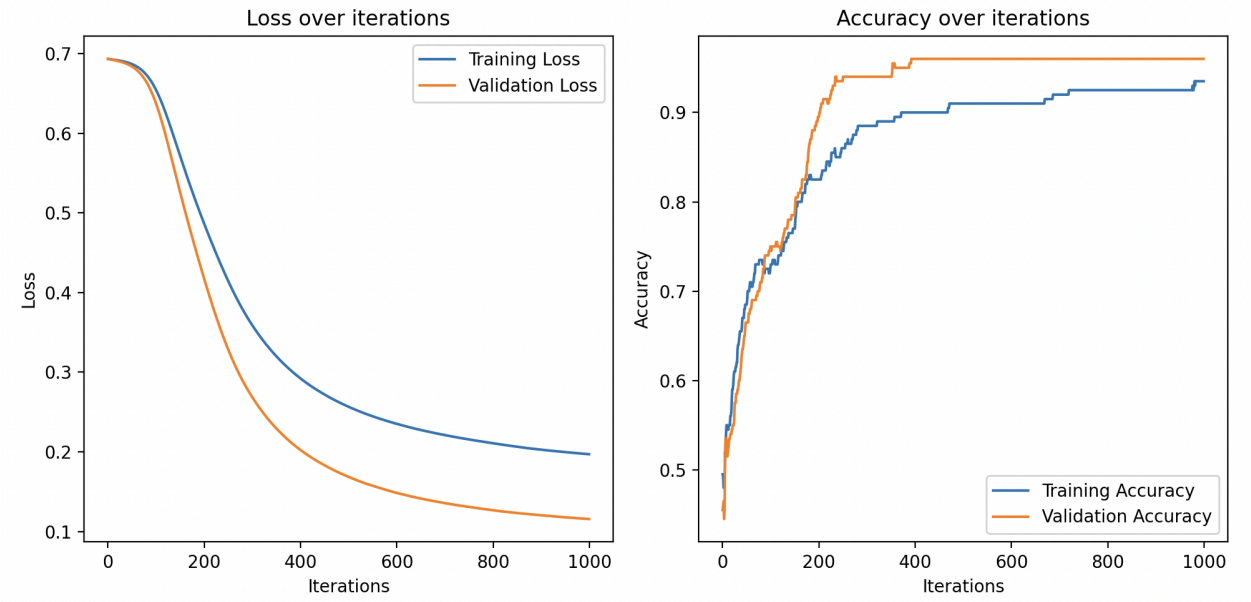


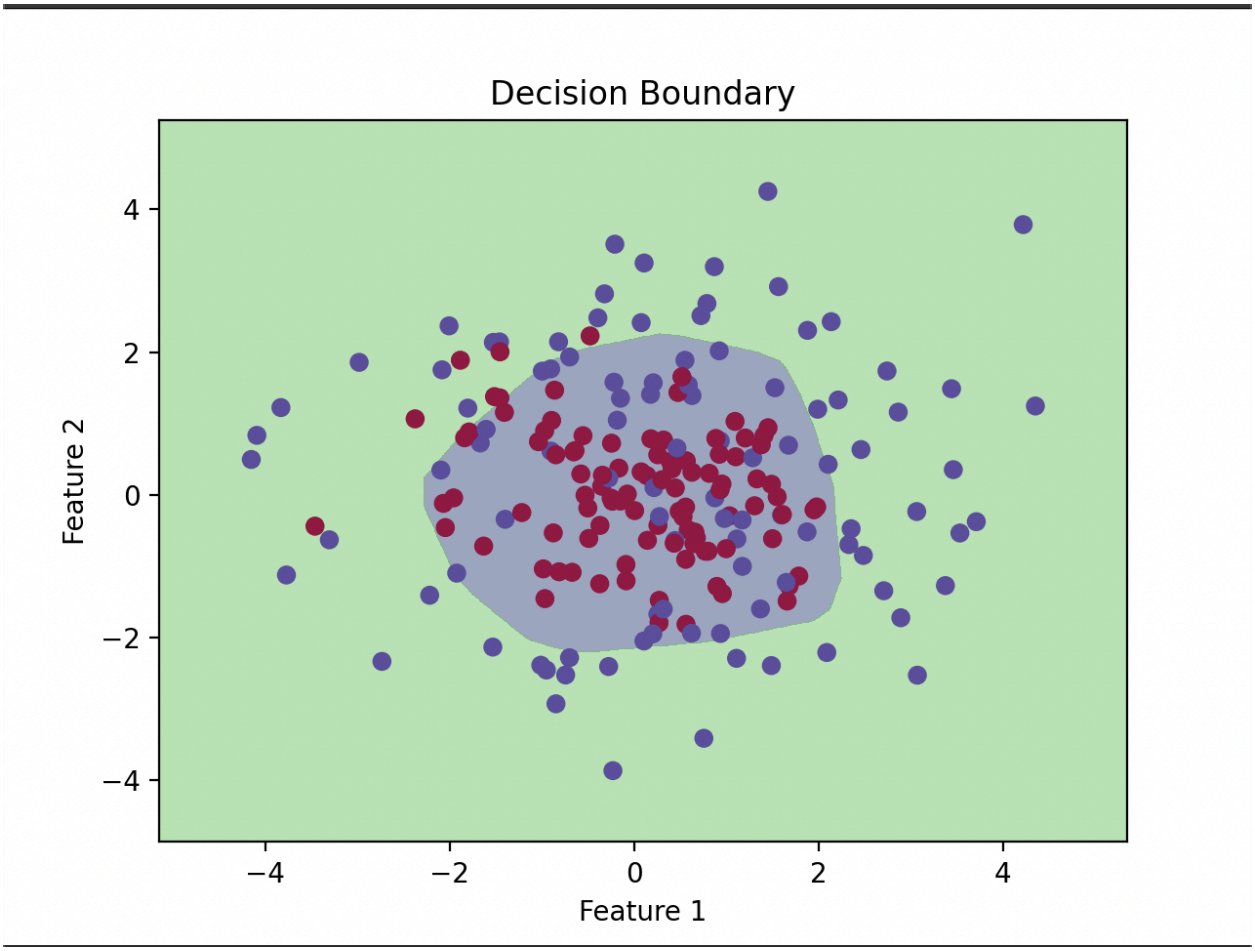
Observations:

- **Generalizability:** Model is generalizing well as test accuracy is near validation accuracy
- **Saturation of curve:** Accuracy curve saturates after 700 epoch this shows that model has learned enough or more complex model is required to learn further
- **Dataset Characteristics:** The XOR problem is a classic example where a linear model would fail, and the decision boundary plot confirms that the dataset has this non-linear structure.

Center_surround:

{'hidden_layer_size': 64, 'learning_rate': 0.1, 'iterations': 1000, 'final_val_acc': 0.96, 'final_val_loss': 0.11568116888929705, 'test_acc': 0.755}


Loss over iterations
Training Loss
Validation Loss
Loss
Iterations
Accuracy over iterations
Training Accuracy
Validation Accuracy
Accuracy
Iterations

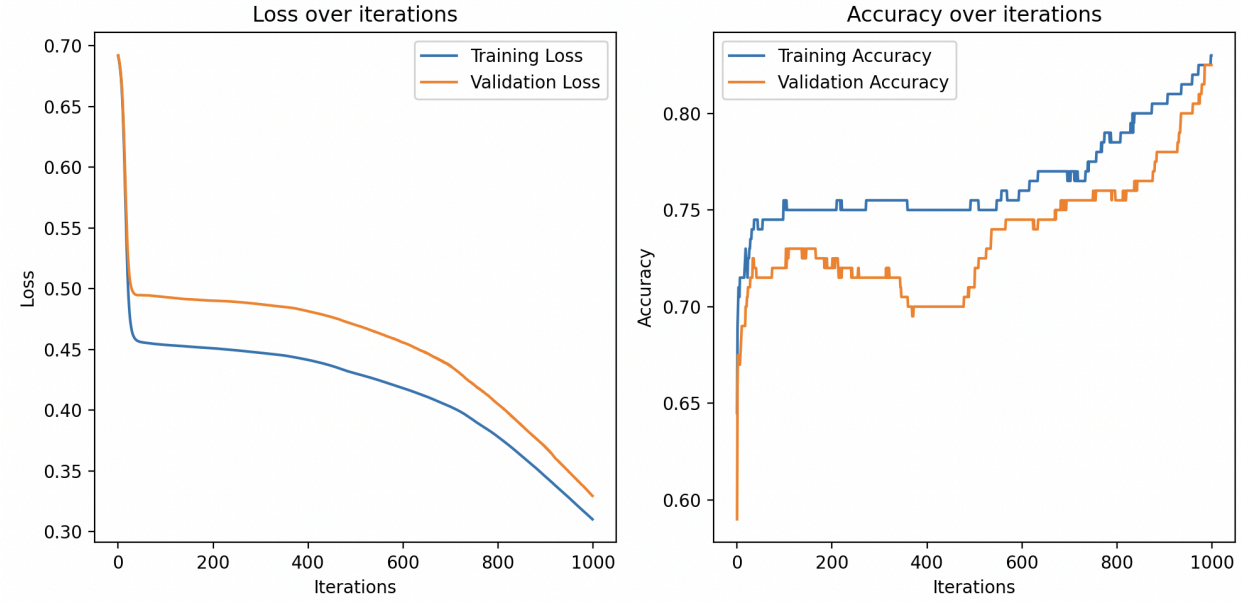

Decision Boundary
Feature 1
Feature 2

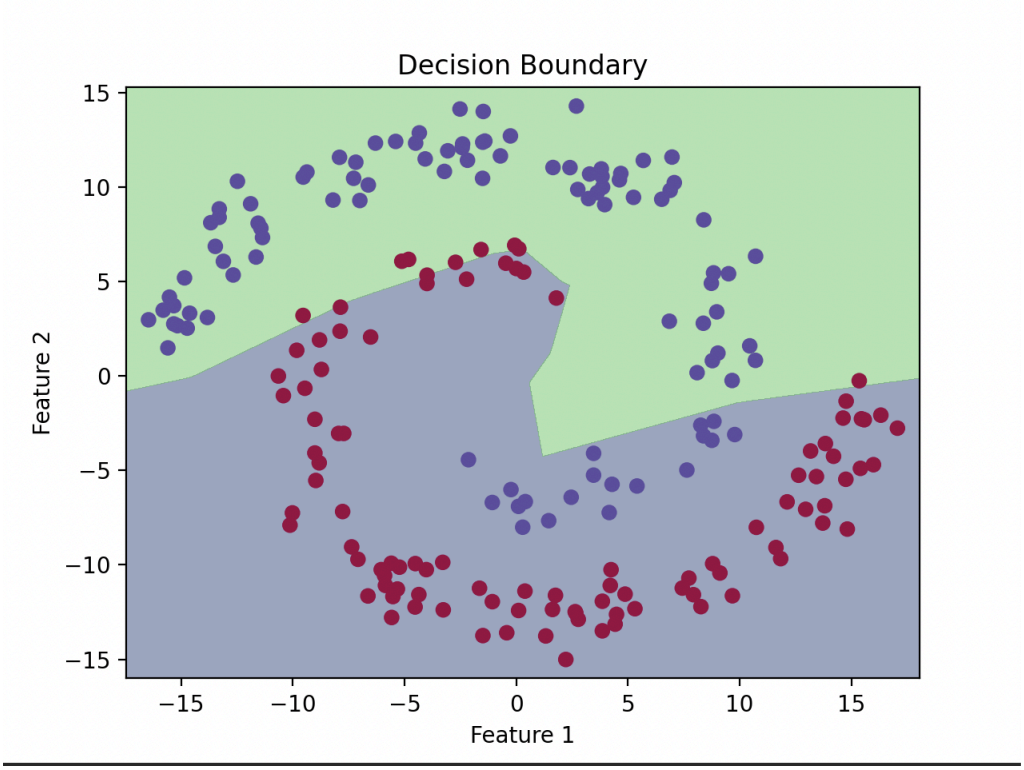
Observations:

- Model might be overfitting as observed by huge difference between validation accuracy and test accuracy
- Decision boundary also shows many test examples are wrongly classified.
- Proves that choosing parameters with best validation accuracy is not always the best approach

Spiral:

{'hidden_layer_size': 64, 'learning_rate': 0.1, 'iterations': 1000, 'final_val_acc': 0.825, 'final_val_loss': 0.3294961760952564, 'test_acc': 0.86}



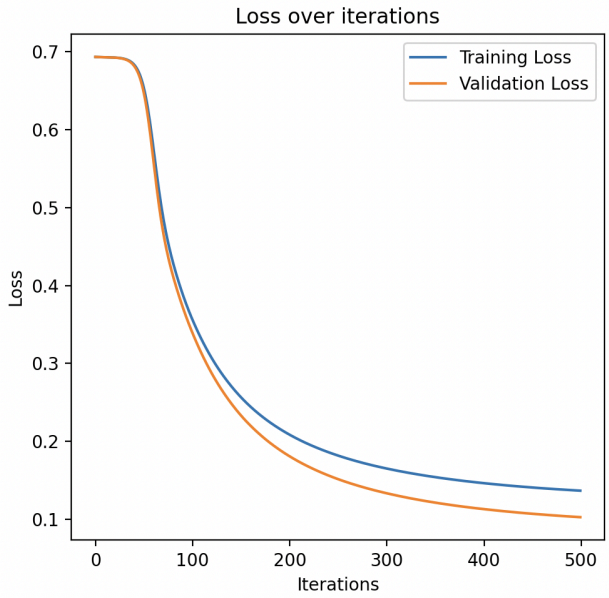


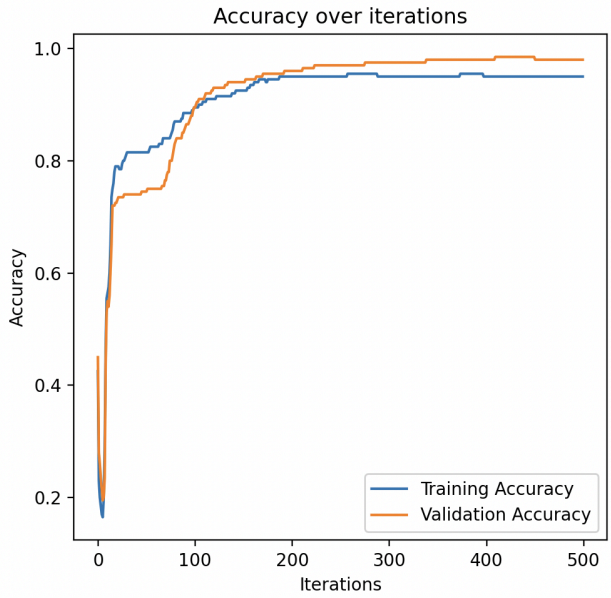
Observations:

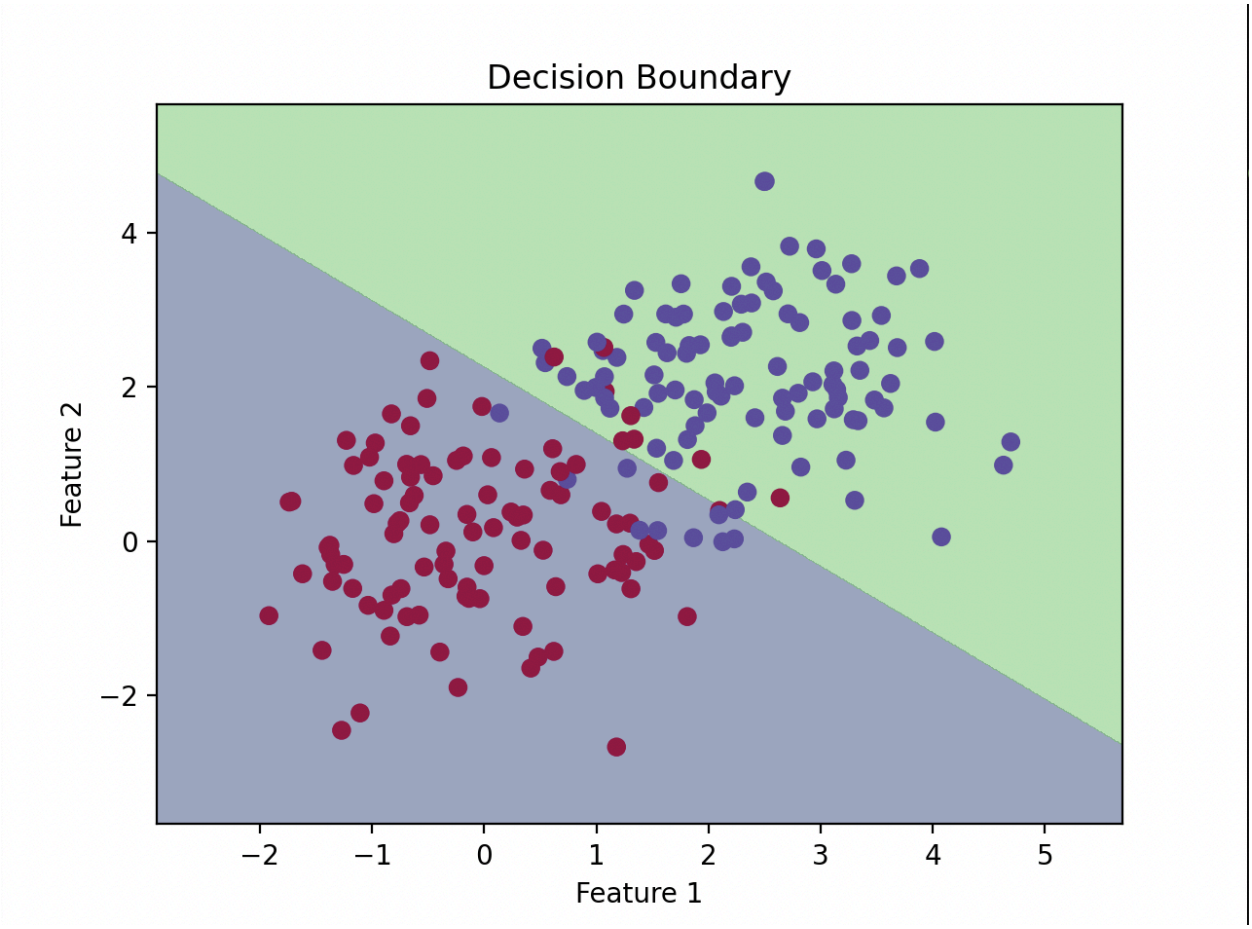
- Generalizing well as test accuracy and validation accuracy is comparable
- We can see training accuracy is not yet saturated. Further training might have provided better results.

Two Gaussian:

{'hidden_layer_size': 4, 'learning_rate': 0.1, 'iterations': 500, 'final_val_acc': 0.98, 'final_val_loss': 0.1025378615263955, 'test_acc': 0.915}


Loss over iterations
Training Loss
Validation Loss
Loss
Iterations
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
100
200
300
400
500

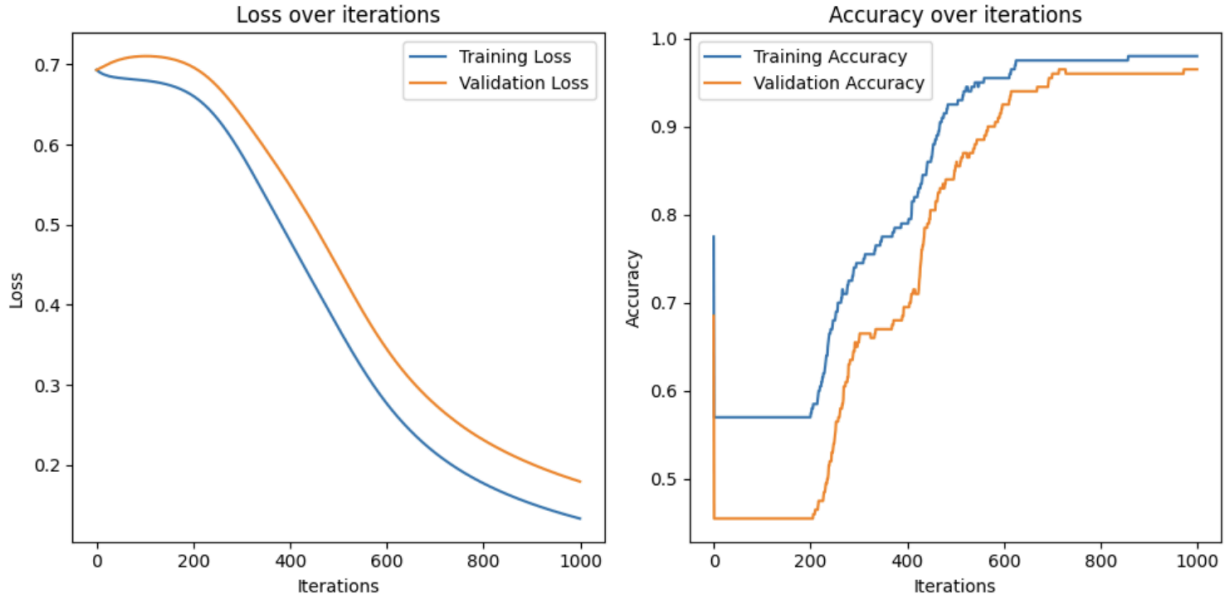

Accuracy over iterations
Training Accuracy
Validation Accuracy
Accuracy
Iterations
1.0
0.8
0.6
0.4
0.2
0
100
200
300
400
500


Decision Boundary
Feature 1
Feature 2
4
2
0
−2
−2
−1
0
1
2
3
4
5

Observations:

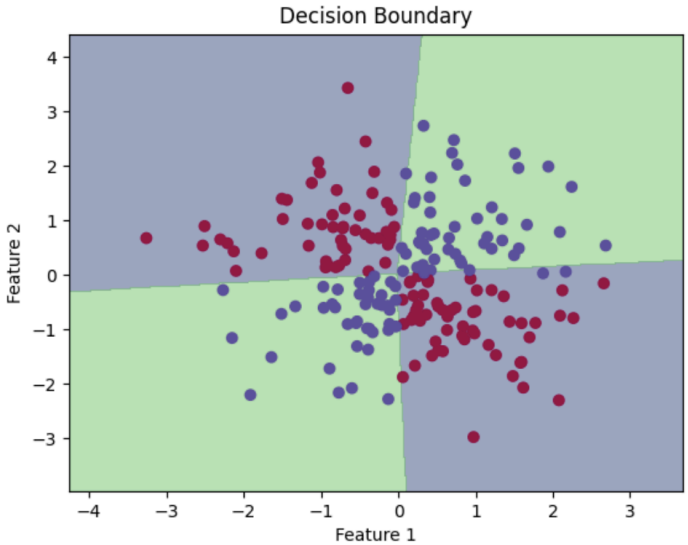
- Model is performing well on test data. However, it is not as accurate as training data.
- There might be signs of a little overfitting
- Test data seems noisy

2. **(2.0 points) Repeat Step 1 using mean squared error as the cost function.**
   - **XOR**
     - Best Hyperparameters
       - {'hidden_layer_size': 64, 'learning_rate': 0.1, 'iterations': 1000, 'final_val_acc': 0.965, 'final_val_loss': 0.17957187595097537}
     - Loss and Accuracy curves



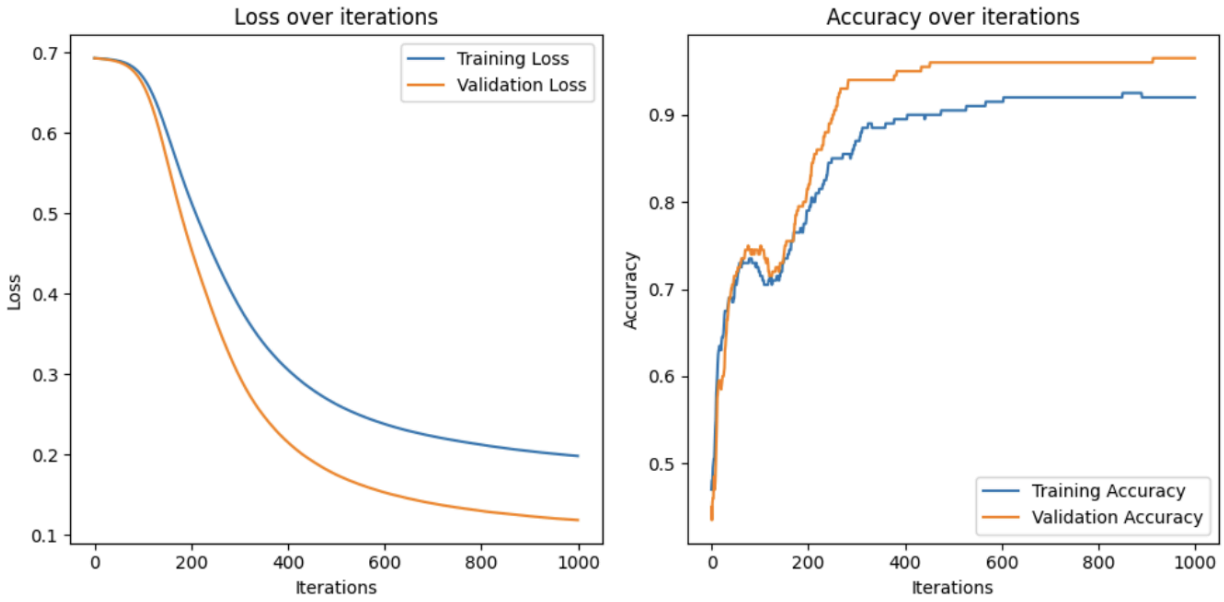
     - Accuracy
       - Test Accuracy 0.975

     - Decision Boundary



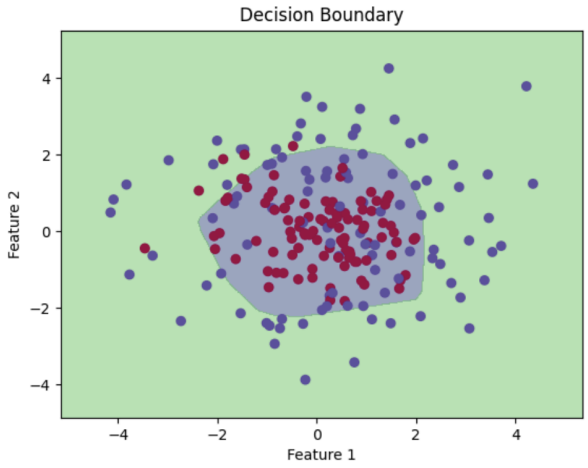
- Observations
  - Model is generalizing well as test and validation accuracy is similar
  - Model stops learning after about 800th iteration so we can apply early stop to save computation or overtraining
  - Model is linear but the decision boundary suggests that data is not linearly separable

- **Center Surround**
  - Best Hyperparameters
    - {'hidden_layer_size': 32, 'learning_rate': 0.1, 'iterations': 1000, 'final_val_acc': 0.965, 'final_val_loss': 0.11855163641030014}
  - Loss and Accuracy Curves



  - Accuracy
    - Test Accuracy 0.76

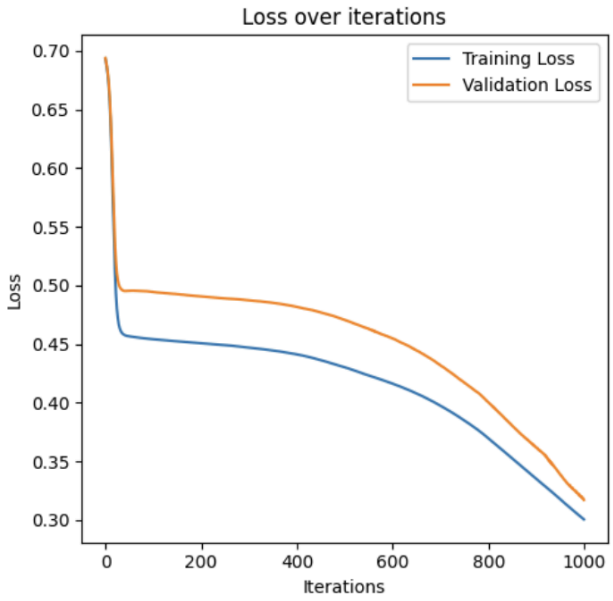
  - Decision Boundary

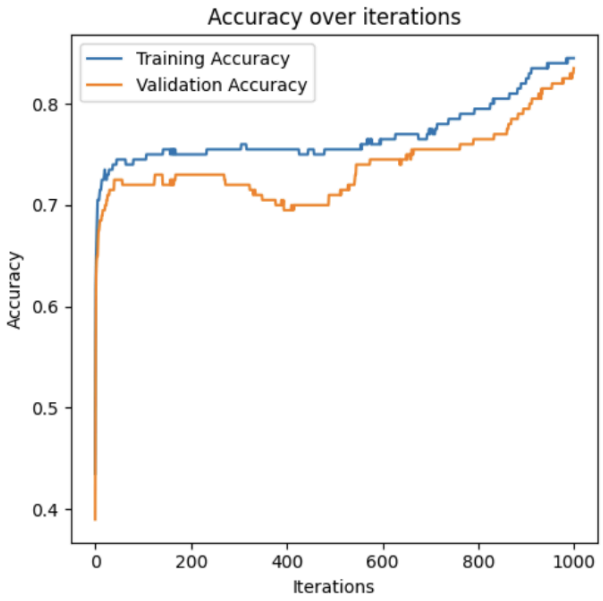


- Observations
  - Model might be overfitting as observed by huge difference between validation accuracy and test accuracy
  - Decision boundary also shows many test examples are wrongly classified.
  - Proves that choosing parameters with best validation accuracy is not always the best approach
  - Not much difference in accuracy upon changing the loss function.

- **Spiral**
  - Best Hyperparameters
    - {'hidden_layer_size': 64, 'learning_rate': 0.1, 'iterations': 1000, 'final_val_acc': 0.835, 'final_val_loss': 0.3173107747437642}

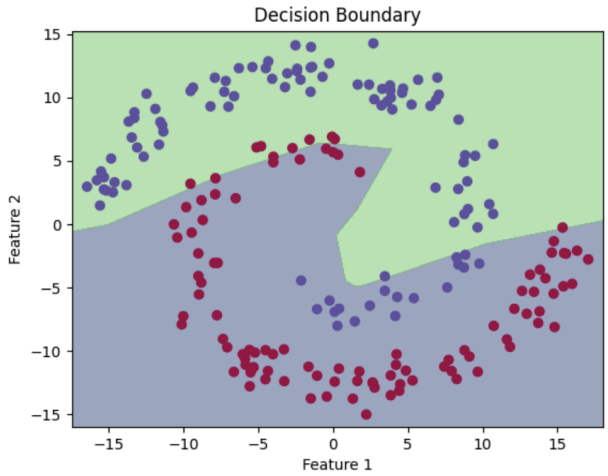
  - Loss and Accuracy Curve





- Accuracy
  - Test Accuracy 0.865

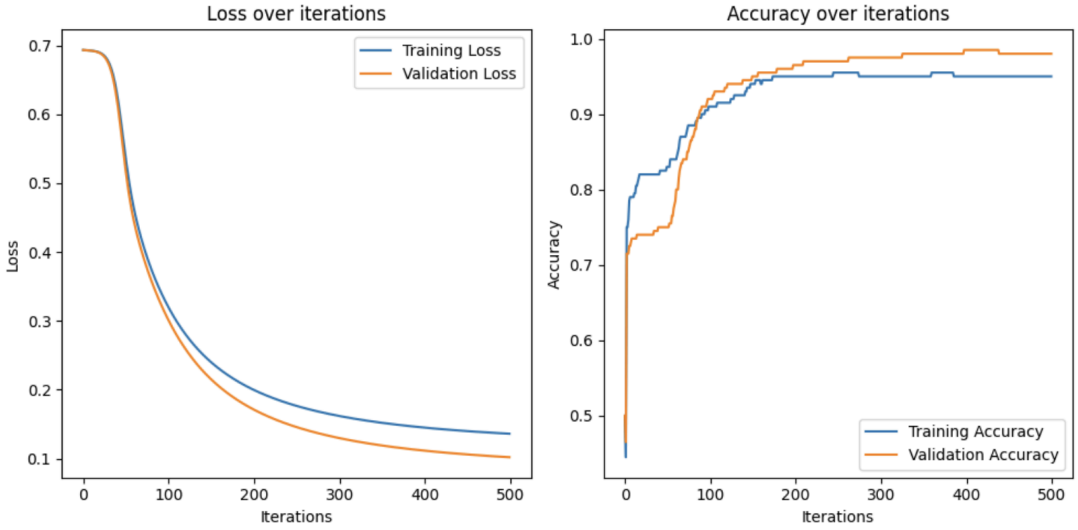
- Decision Boundary



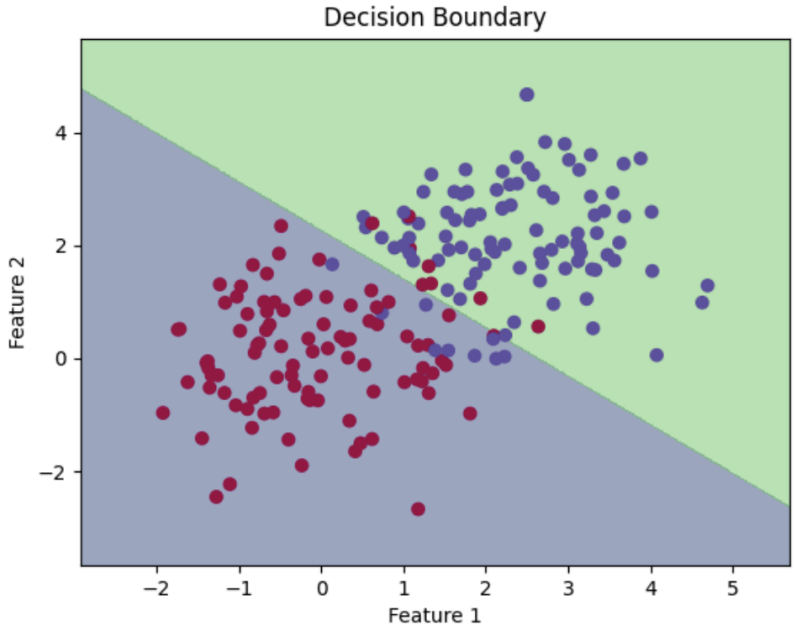
- Observations
  - Generalizing well as test accuracy and validation accuracy is comparable
  - We can see training accuracy is not yet saturated. Further training might have provided better results.

- **Two Gaussians**
  - Best Hyperparameters
    - {'hidden_layer_size': 4, 'learning_rate': 0.1, 'iterations': 500, 'final_val_acc': 0.98, 'final_val_loss': 0.10187663227349279}

  - Loss and Accuracy Curves



  - Decision Boundary



  - Accuracy
    - Test Accuracy 0.915

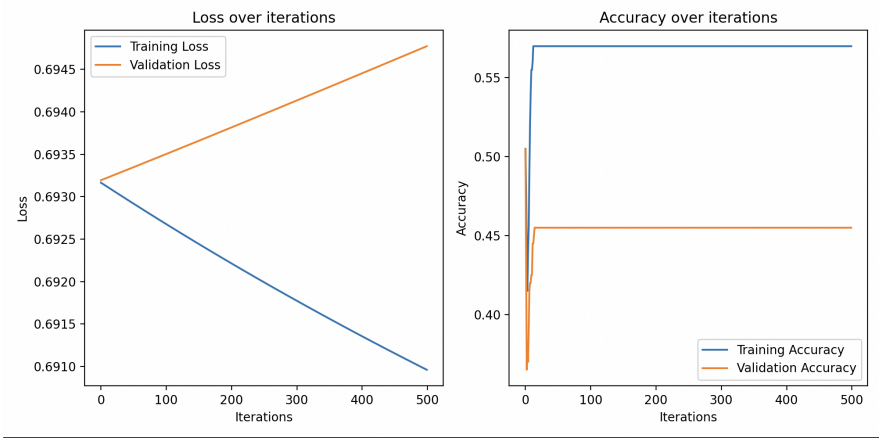
  - Observations

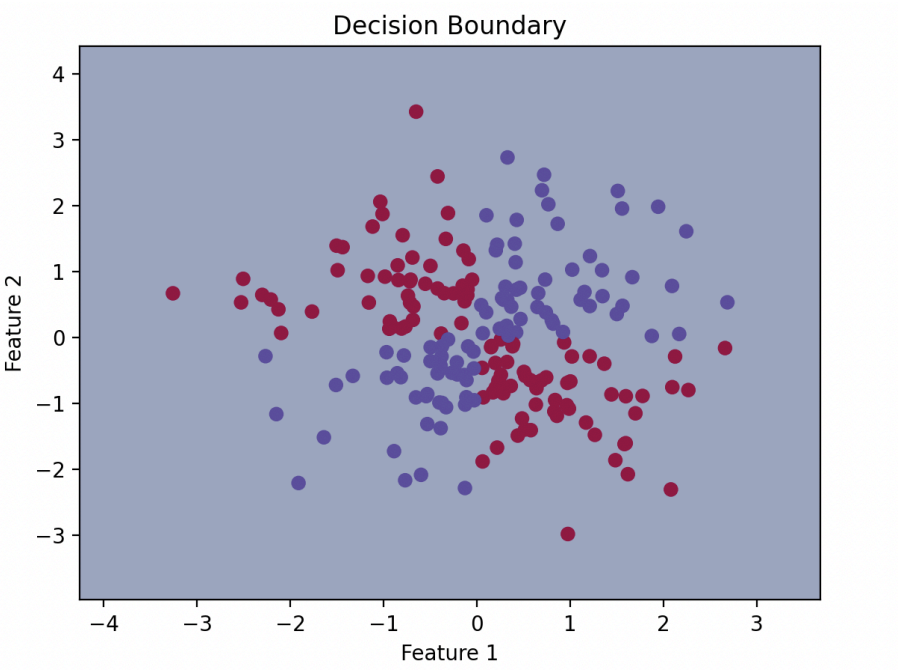
- Model is performing well on test data. However, it is not as accurate as training data.
- There might be signs of a little overfitting (.91 Val > .90 Test)
- Test data seems noisy

3. (2.0 points) Select the worst-performing model (dataset, cost function and number of hidden nodes) from the above experiments and plot decision boundaries learned for each node in your hidden layer. Discuss why you selected this instance and speculate about factors contributing to poor performance.

XOR data:

{'hidden_layer_size': 32, 'learning_rate': 0.001, 'iterations': 500, 'final_val_acc': 0.455, 'final_val_loss': 0.6947731682843576, 'test_acc': 0.54}





Factors contributing to bad performance:

- Too low learning rate
- Need for more training
- Inaccurate hidden layer size (not enough nodes to learn features properly)
- Maybe using better loss function can help

4. (1.0 points) Discuss how you might encourage your model to learn "feature maps" that could improve the performance of the instance selected for Step 3.

To enhance the model's ability to learn more effective feature maps, particularly for the instance selected in Step 3, a multifaceted approach focusing on hyperparameter optimization and model architecture refinement is essential. Here are several strategies that can be implemented to achieve this:

- **Optimal Learning Rate Search:** An optimal learning rate is crucial for efficient training. It should be neither too high, which may cause the model to overshoot minima, nor too low, which could result in slow convergence. Techniques such as

learning rate scheduling or adaptive learning rate methods (e.g., Adam, RMSprop) can dynamically adjust the learning rate during training to find a balance between convergence speed and stability.

- **Extended Training Duration:** Increasing the number of training epochs allows the model more iterations to adjust its weights and biases towards minimizing the loss function. However, care must be taken to avoid overfitting. Employing early stopping mechanisms can mitigate this by halting training when the validation loss stops improving.
- **Hidden Layer Architecture Adjustment:** Experimenting with different numbers of hidden layers and neurons can significantly impact the model's capacity to learn complex feature representations. For the XOR problem, which is non-linearly separable, adding more hidden layers or adjusting the number of neurons can enable the model to capture more complex patterns. However, this should be done judiciously to prevent overfitting.

5. (BONUS 2.0 points) Implement the approach specified in Step 4. Discuss and present results that illustrate "why" your approach did or did not enhance performance. Note: You may use PyTorch (or another deep learning package) to implement this part of the assignment.

Implementation and Outcome

We will try the following hyperparameters and see what works best for us:

- hidden_layer_sizes = [4, 8, 16, 32, 64]
- learning_rates = [0.1, 0.01, 0.001, 0.0001]
- iteration_counts = [100, 500, 1000]

We experimented with a range of configurations for hidden layer sizes, learning rates, and iteration counts. Ultimately, the configuration that yielded the best results featured:

Hidden Layers: 16
Learning Rate: 0.1
Iterations: 1000

This configuration dramatically improved the model's performance, achieving a final validation accuracy of 0.96, a validation loss of 0.1917573914307001, and a test

accuracy of 0.975. These results indicate a significant enhancement in the model's ability to learn and generalize the XOR problem.

Analysis of Improvement

The improved performance can be attributed to several key adjustments:

- **Increased Learning Rate:** Moving from a learning rate of 0.001 to 0.1 likely allowed the model to make more significant updates to its weights during training, thus escaping potentially poor local minima faster.
- **Optimized Hidden Layer Size:** Reducing the hidden layer size from 32 to 16 nodes might have helped the model to focus on learning more meaningful representations without overfitting or underfitting, which is crucial for a relatively simple problem like XOR.
- **Extended Training Iterations:** Increasing the number of iterations to 1000 provided the model with more opportunities to learn from the data, refining its decision boundaries over time.