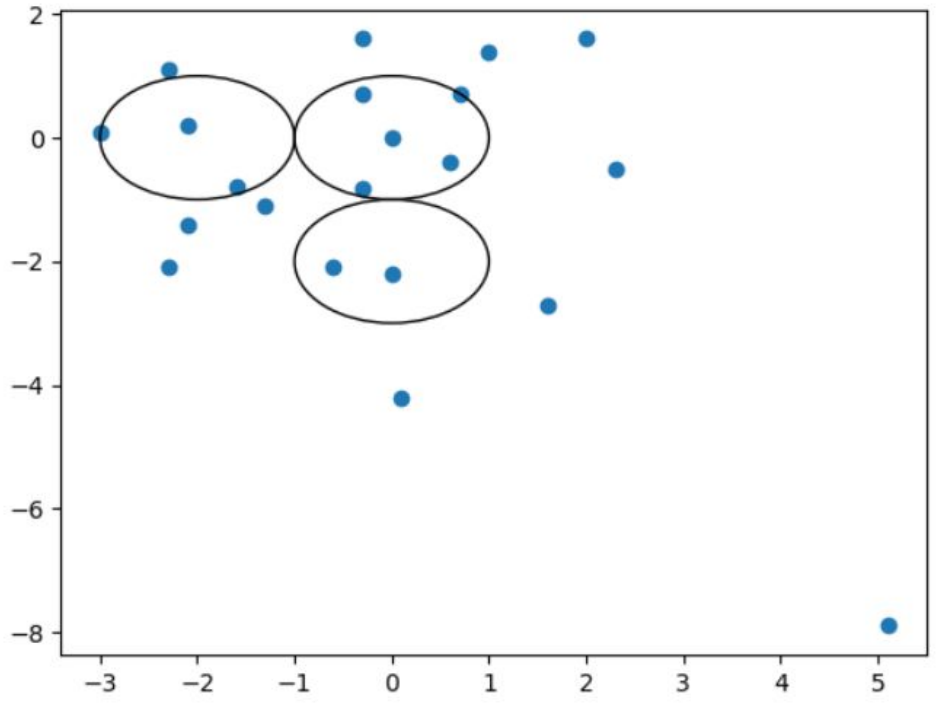
Neural Network

By Shahzab

Dataset Overview - Training Data (with true categories)



Data Visualization

Model Architecture:

- The model is a sequential model with three dense layers
- Each dense layer has different output shapes and parameters

Layer Details:

- The first dense layer has an output shape of (None, None, 1, 3) with 9 parameters
- The second dense layer has an output shape of (None, None, 1, 3) with 12 parameters
- The second dense layer has an output shape of (None, None, 1, 1) with 4 parameters

Training Epochs:

- The model was trained for 30 epochs (an epoch is one use of the full training dataset, so there were 30 cycles of parameter updates)

Prepossessing

Feature Scaling:
- Scale features to a similar range to prevent certain features from dominating others during model training
- Common scaling techniques include Min-Max scaling and Standardization (Z-score normalization)

Feature Encoding:
- Convert categorical variables into numerical representations suitable for machine learning algorithms
- Techniques include one-hot encoding for nominal variables and label encoding for ordinal variables

Train-Test Split:
- Split the dataset into training and testing sets to evaluate model performance on unseen data
- 80% for training and the remainder for testing
- Cross-validation can also be used to make the most out of existing data

Normalization vs. Standardization:
- Normalization scales features to a range between 0 and 1, while standardization transforms features to have a mean of 0 and a standard deviation of 1
- The choice between normalization and standardization depends on the algorithm's sensitivity to feature scales

Importance of Preprocessing:
- Proper preprocessing improves model performance, enhances interpretability, and reduces the risk of overfitting
- It ensures that the data meets the assumptions of the chosen machine learning algorithm

Model Architecture

Layer Types and Output Shapes:
- The model includes two types of layers: Dense layers with hyperbolic tangent (tanh) activation and a final Dense layer with a sigmoid activation
- The output shape of each layer is (None, None, 1, n), indicating the batch size (None), sequence length (None), single-dimensional input (1), and number of neurons (n)

Parameter Count:
- The total number of parameters in the model is 25, with all parameters being trainable

Training:
- The model is trained using the binary cross-entropy loss function and stochastic gradient descent (SGD) optimizer
- Training is performed for 30 epochs with a batch size of 1

Training The Model

- The model was trained with the keras package, using binary cross entropy along with stochastic gradient descent
- The gradient is computed via the standard method- backpropagation, with the result being used by stochastic gradient descent
- There are two hidden layers with 3 nodes

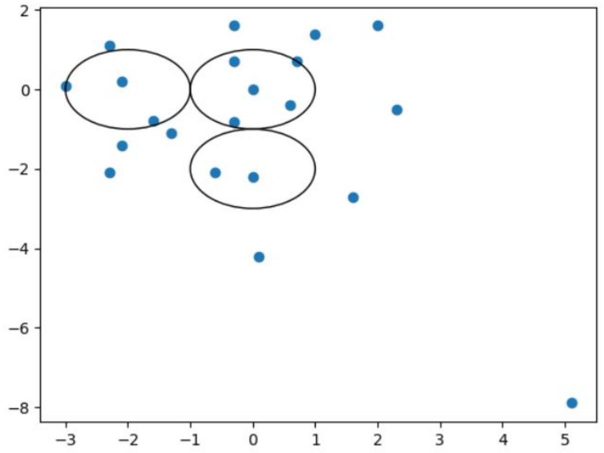
Loss Function and Optimizer

- Since the data is binary, the sigmoid activation function was used for the output layer and binary cross entropy loss
    - Note: This loss is just the negative log likelihood of the bernoulli random variable (so its equivalent to maximizing the log likelihood)
- The optimizer is stochastic gradient descent, which is a standard method for doing backpropagation
    - This is gradient descent but instead of using all the data points to estimate the gradient, one random point is used

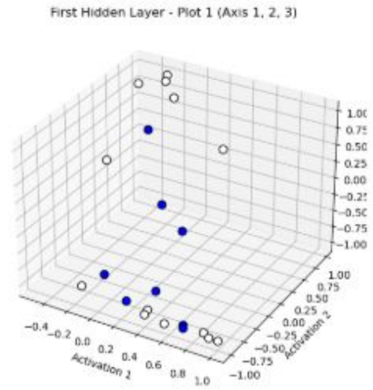
Model Evaluation

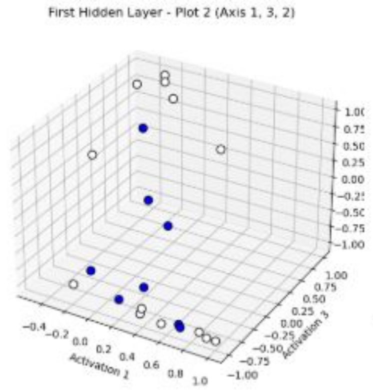
- The final model when trained on the data ended up assigning to all points to be 0, i.e. outside the 3 circles

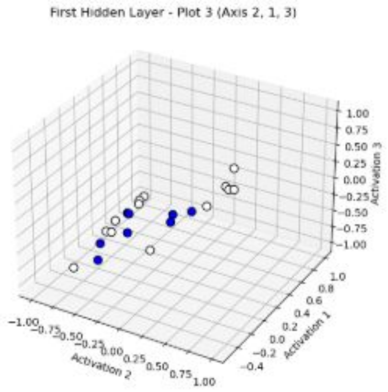
Recall the training data:

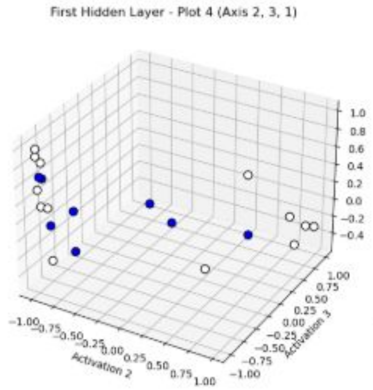


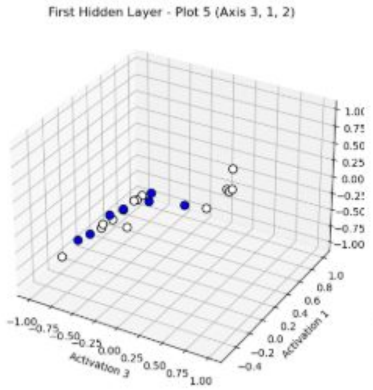
3-D Plots (Based on first hidden layer)

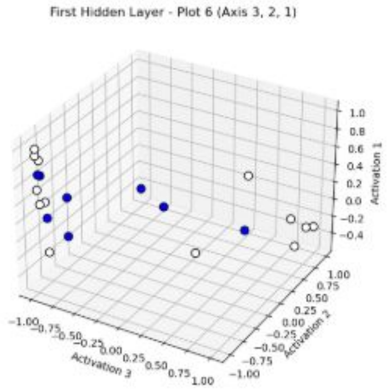






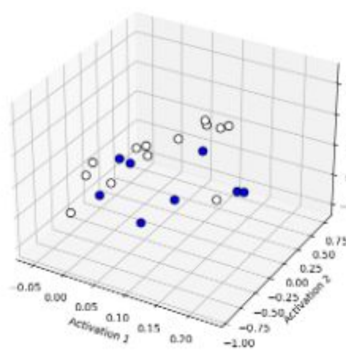




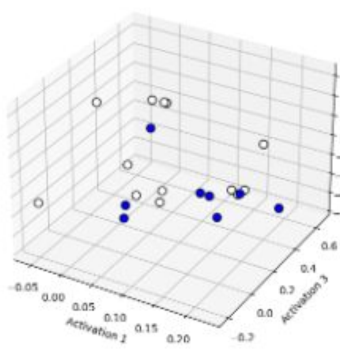


3-D Plots (Based on second hidden layer)

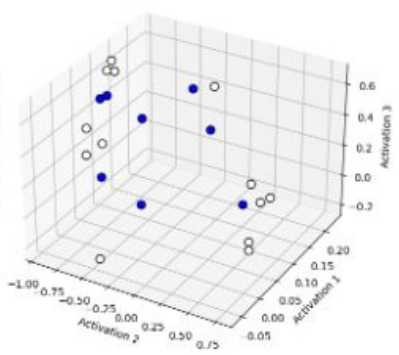
Second Hidden Layer - Plot 1 (Axis 1, 2, 3)



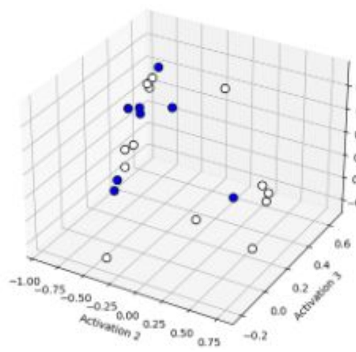
Second Hidden Layer - Plot 2 (Axis 1, 3, 2)



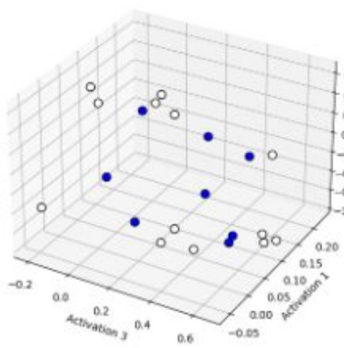
Second Hidden Layer - Plot 3 (Axis 2, 1, 3)



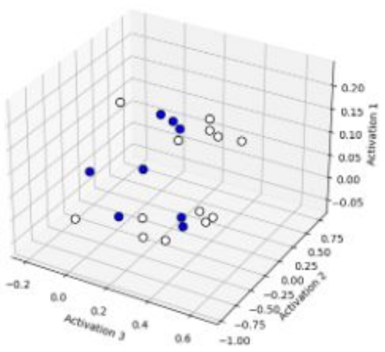
Second Hidden Layer - Plot 4 (Axis 2, 3, 1)



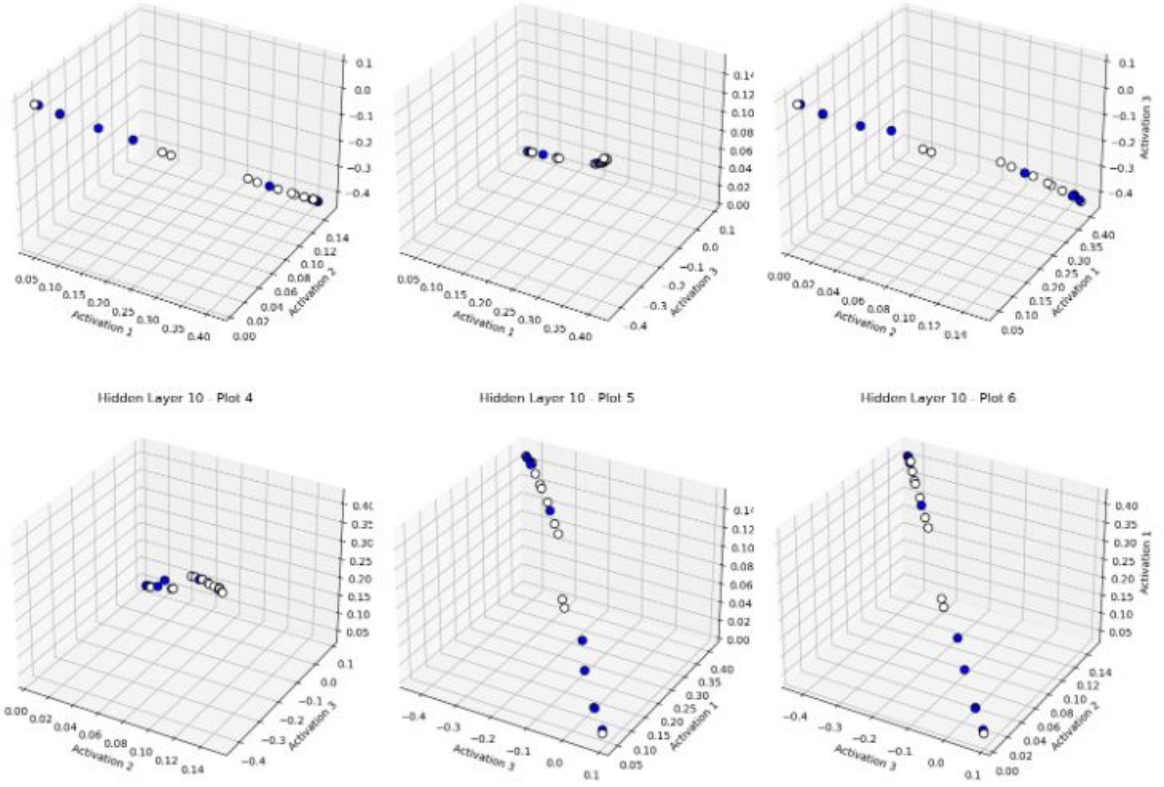
Second Hidden Layer - Plot 5 (Axis 3, 1, 2)



Second Hidden Layer - Plot 6 (Axis 3, 2, 1)



3-D Plots (if there’s a tenth hidden layer)

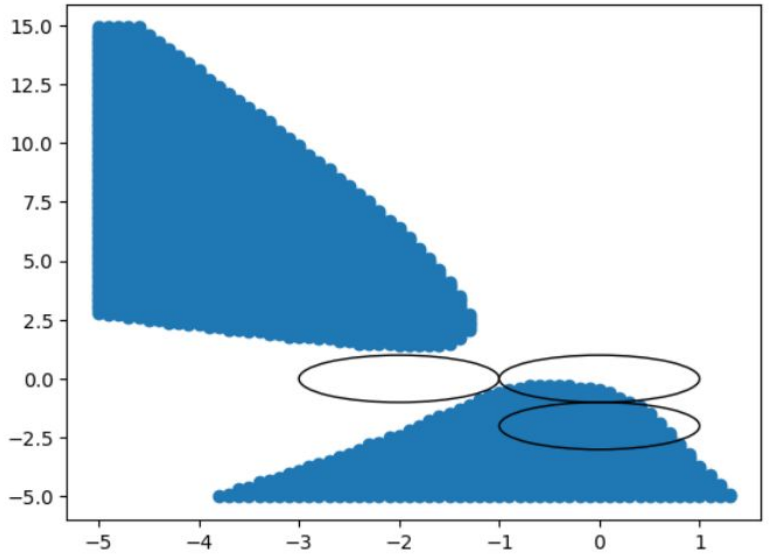


Conclusion About The 3-D Plots

- There’s a lot of movement of data points between first and second hidden layers but not clear what it might end up with if more hidden layers are added
- If there’s a tenth hidden layer, the data points don’t create a proper line with two class memberships not clearly separated
- Neural networks don’t seem to work well with data points with class memberships related to whether whether a point is outside or inside a shape

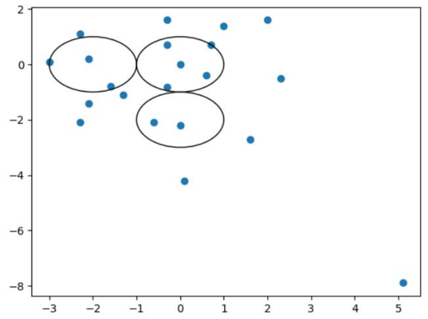
Visualizing Decision Boundary

Final Model Decision Boundary (blue=1, white=0)



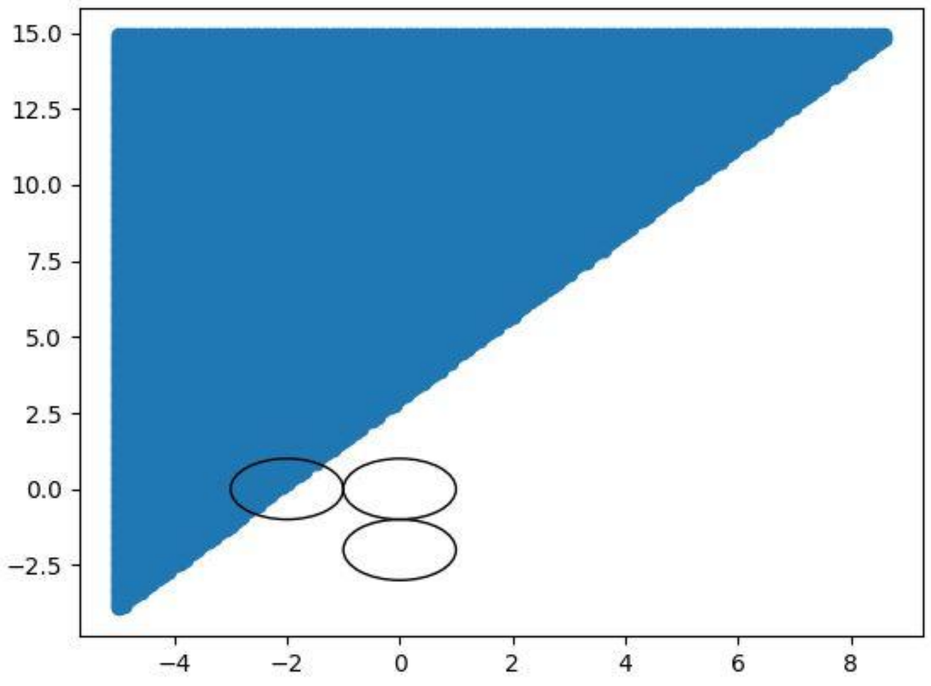
Note: True value is 1 for inside circle, 0 otherwise

Training Data



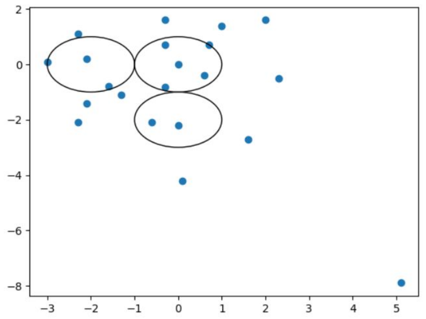
Understanding intermediate model (no hidden layers)

0 Hidden Layer Decision Boundary (blue=1, white=0)



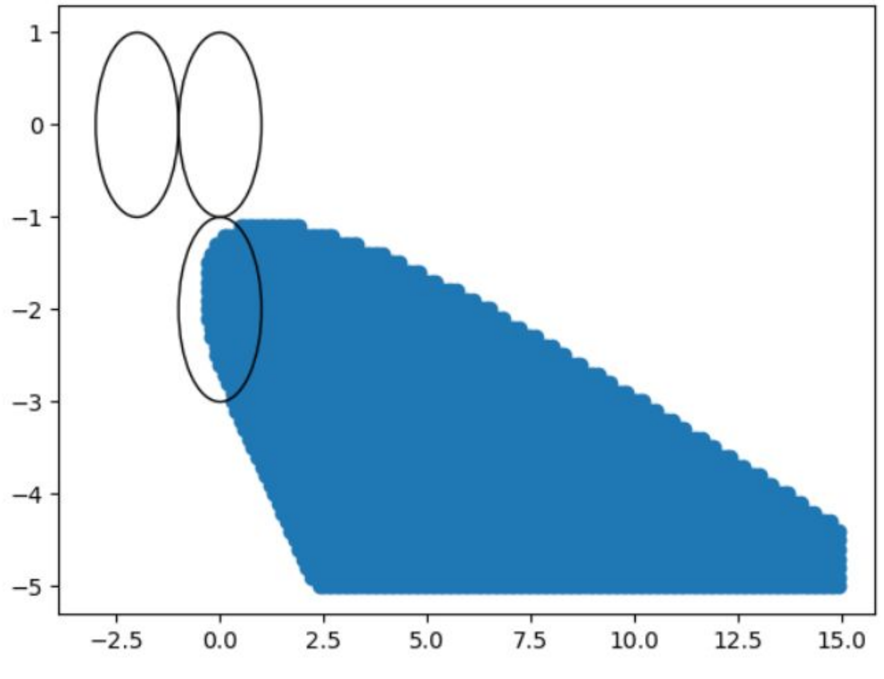
Note: True value is 1 for inside circle, 0 otherwise

Training Data



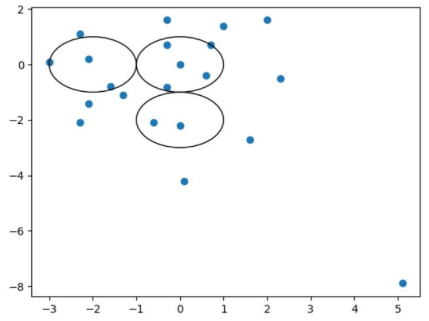
Understanding Intermediate Model (1 hidden layer)

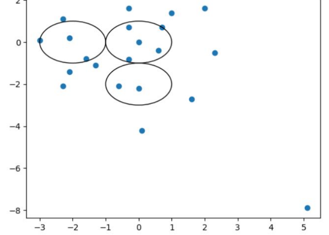
1 Hidden Layer Decision Boundary (blue=1, white=0)



Note: True value is 1 for inside circle, 0 otherwise

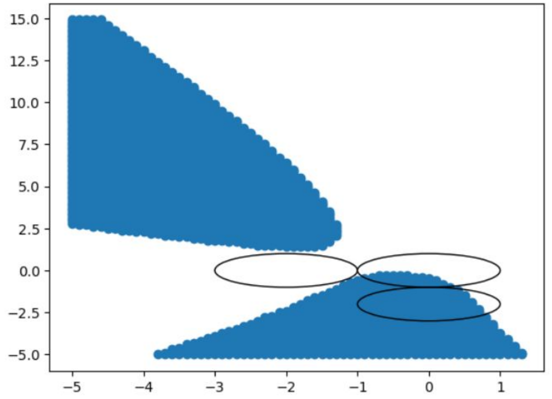
Training Data

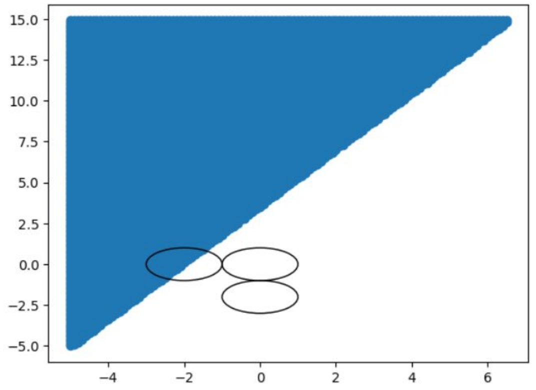


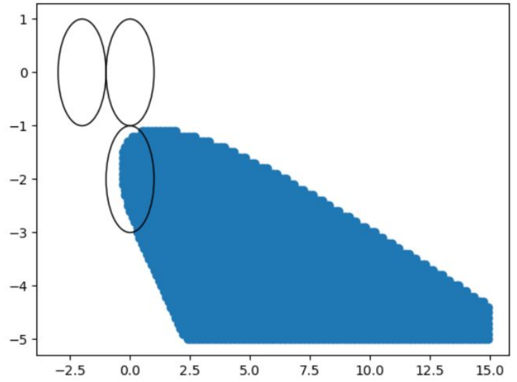


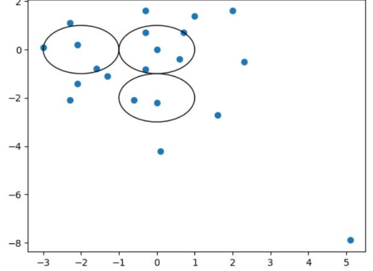
Conclusions about the models

- At least for this case, adding extra hidden layers allows for an additional curved region (no hidden layers means purely linear boundary, one layer allows for one “bell curved” like surface, two layers has two of them)
- In general the models are odd given the training data, since it misclassifies a large minority of the points between each iteration (even if it does slightly better each time)





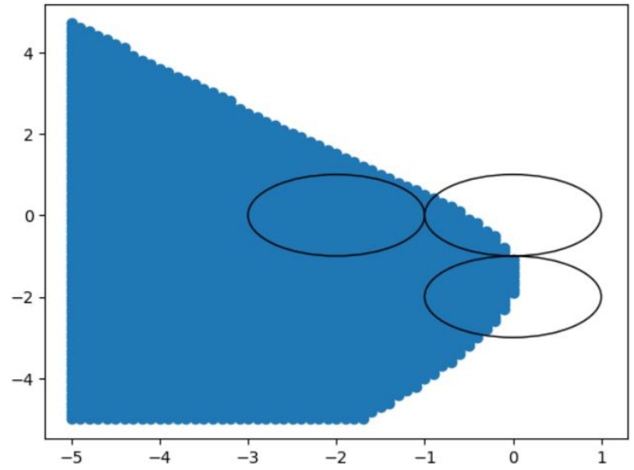




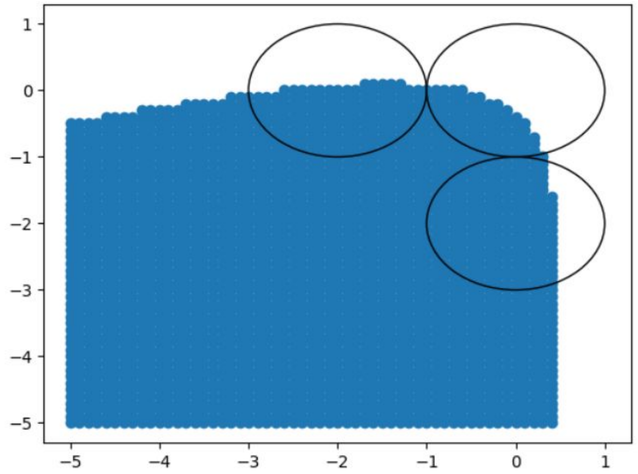
More Hidden Layers

Note: 10 and 20 hidden layers produced models that classify all points as outside the circle.

3 Hidden Layers



5 Hidden Layers



Overfitting

- It doesn't appear as though the model is overfitting the dataset via its predictions, the prediction accuracy is quite low on the training dataset
- However if one looks at how the model builds up through the intermediate models, there’s a possible overfit on certain characteristics of the training dataset- to get the shapes that were shown
- Training the model on other datasets gives similar results (overfitting in the sense that the boundaries are nothing like the true ones, and it does poorly on test datasets), even if prediction accuracy on training dataset is not very high

Conclusion

- Each hidden layer introduces additional non-linear transformations to the data, enabling the model to capture intricate patterns and relationships that may not be discernible with fewer layers
- Can show signs of overfitting, particularly if the model's capacity exceeds the complexity of the dataset or if the training data is limited
- When examining intermediate models during training, it's possible to observe signs of overfitting

Questions?