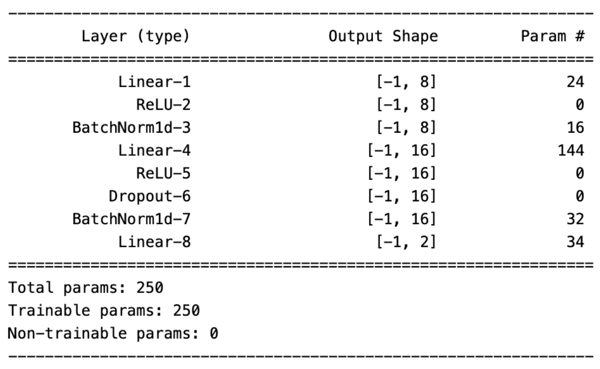
CSE229 – Non-linear classification under covariate shift

This homework will deal with nonlinear classification under *covariate shift* – the test set distribution is different from the training set distribution. We will use ensembling to improve test performance and confidence calibration.

(a) Architecture and hyper-parameter choices:

Architecture:

i. Fully Connected Layers: I have used two hidden layer of size 8, and 16.
ii. ReLU Activation: ReLU allows the model to learn complex, non-linear relationships in the data while avoiding the vanishing gradient problem. I have added ReLU after each Linear layer.
iii. Batch Normalization: I have added this layer after each ReLU to normalize the inputs to each layer. This helps in mitigating the internal covariate shift problem during training. It improves generalization by reducing the dependency on specific parameter initialization.
iv. Dropout: This is a regularization technique that randomly drops a certain percentage of units in a layer during training. This reduces overfitting and encourages the model to learn more robust representations. In PyTorch, I provided 0.1 probability to drop a unit in the second hidden layer.



```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Linear-1                    [-1, 8]              24
              ReLU-2                    [-1, 8]               0
       BatchNorm1d-3                    [-1, 8]              16
            Linear-4                   [-1, 16]             144
              ReLU-5                   [-1, 16]               0
           Dropout-6                   [-1, 16]               0
       BatchNorm1d-7                   [-1, 16]              32
            Linear-8                    [-1, 2]              34
================================================================
Total params: 250
Trainable params: 250
Non-trainable params: 0
----------------------------------------------------------------
```

Figure 1 Model Architecture

Hyperparameters:

i. Learning Rate: I have tried different learning rate (0.001 – 0.0001) and manually tried to identify the best that gives a linear boundary. The found learning rate of 0.00015 seems to be performing good.
ii. Number of Epochs: The choice of 100 epochs is arbitrary but reasonable, allowing the model to learn from the data for a sufficient duration without risking overfitting.
iii. Optimizer: I have used Adam optimizer. It adapts the learning rate for each parameter based on the estimates of the first and second moments of the gradients.

Loss Function:

i. The *BCEWithLogitsLoss* function is a suitable choice for binary classification problems where the model's output represents logits.

(b) Visualize:

Scatter plot of validation and test data, with all the classifier boundaries (each of ten models and the ensemble model) superimposed on top:

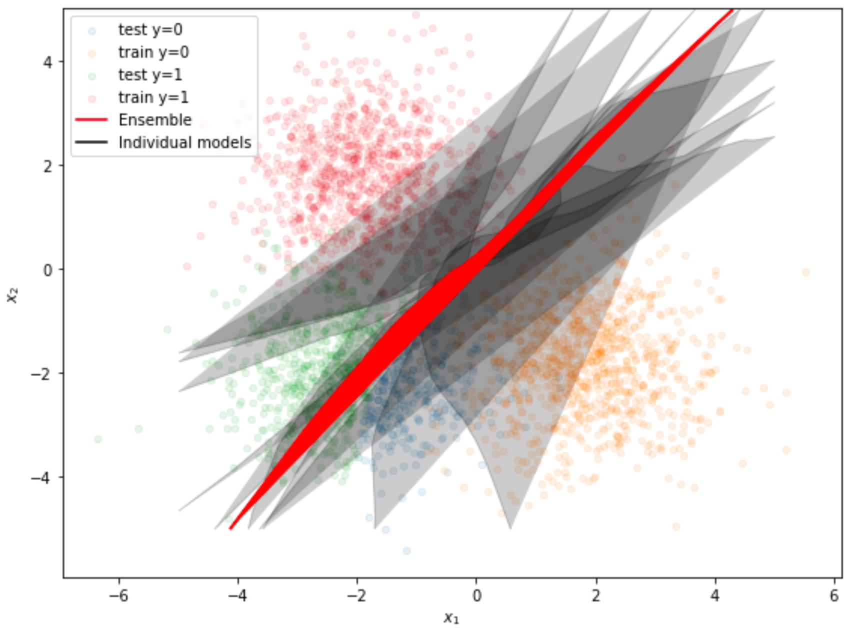


Figure 2 Classifier Boundaries

Confidence vs. Accuracy plot for each model and ensemble model:

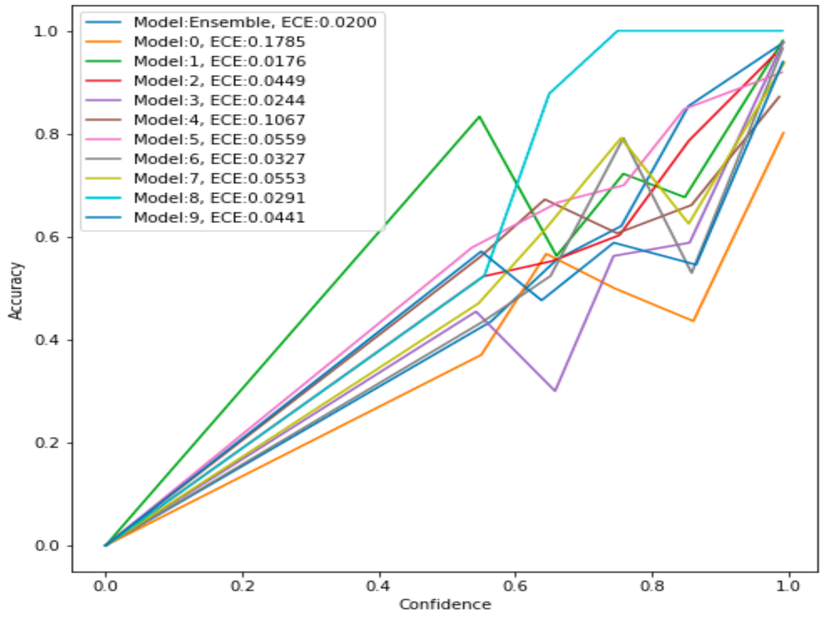


Figure 3 Confidence vs. Accuracy

(c) Calculate

Accuracy and expected calibration error for each model for both train/validation/and test data:

Table 1 Models vs. Ensemble

	Mean across models	Ensemble model
Train accuracy	0.993	0.998
Validation accuracy	0.991	0.996
Test accuracy	0.727	0.838
Train ECE	0.003	0.002
Validation ECE	0.004	0.003
Test ECE	0.209	0.059

(d) Discuss

I have used a neural network with two hidden layers, which is designed to capture non-linear relationships in the data. From Figure 2, we can observe that the models have identified regions as the classification boundaries, indicating non-linear relationships between the features and class labels. The ensemble of these models results in a thinner decision boundary, suggesting that the ensemble approach helps find a boundary that is close to linear. This boundary performs well on unknown test data with a different distribution. The effectiveness of the ensemble is reflected in Table 1, which shows 83.8% accuracy on the test data.

Batch normalization reduces covariate shift and improves gradient flow, while dropout reduces overfitting and encourages ensemble learning. These techniques help the model find and adapt to complex decision boundaries, resulting in good performance on unseen data.