Assignment 6

314652021羅瑋翔

Written Assignment

We all know that if the distribution of the dataset is like normal distribution, then the approximation of GDA would be great even if the dataset is small. But the case in Assignment 4, is obviously unlike it.
Is there any other classification way to solve the parameters and perdict the unknown data well(not to use neural network)?

Is there any other way to define the loss function to make the error between true and prediction value while the dataset is not big(like the notation we used in PINNs)?

Program Assignment

Classification using GDA

Data Processing
I use the same methods as Assignment to transfer the xml file into csv file.
And there are three column in this csv file, longtitude, latitude and feature, respectuvely.
If temperature $!=$ -999, then feature = 1. Otherwise, feature = 0.

Data Partition
Training : 70%
Validation : 15%
Test : 15%

Notation of GDA
GDA model assumes that the conditional distribution of the features given the class label is multivariate normal:

$$p(x|y=0,1) \sim \mathcal{N}(\mu_k, \Sigma),$$

where μ_k and Σ are the class-specific mean vector and covariance matrix.

Using the Bayes' rule, we can derive the posterior distribution on y given x:

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)}$$

and the denominator is given by:

$$p(x) = p(x|y=1)p(y=1) + p(x|y=0)p(y=0)$$

Then we use the training data to solve the parameters in PDF of Gaussian distribution.
And use these parameters to predict the test data.

Accuracy
I use the method AUC(area under ROC curve) to evaluate the accuracy of the model.

- AUC = 1.0$\rightarrow$ Perfect Discriminator
- AUC = 0.5$\rightarrow$ Randomly Prediction
- AUC < 0.5$\rightarrow$ Model prediction in a reverse way

```
========================================
           Comparison Summary
========================================
QDA Test AUC: 0.9326
========================================
```
Figure 1. AUC of GDA discrimination model.

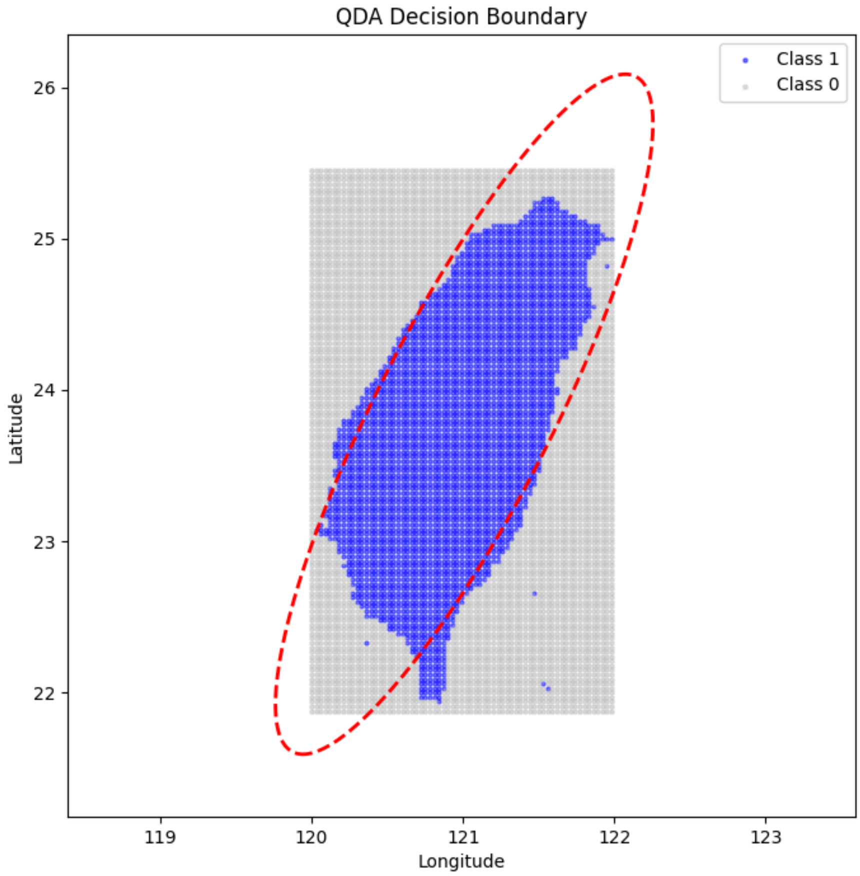

Figure 2. The valid/invalid data points and the decision boundary.

Regression

Implementation

In **combine_model.py**

$C(\vec{x})$: a classification tree model like what I've done in Assignment, but I make it deeper and wider to obtain a better decision boundary then what I've done in Assignment 4. And 70% training, 15% validation and 15% testing. And use threshold= 0.5 to improve the accuracy.

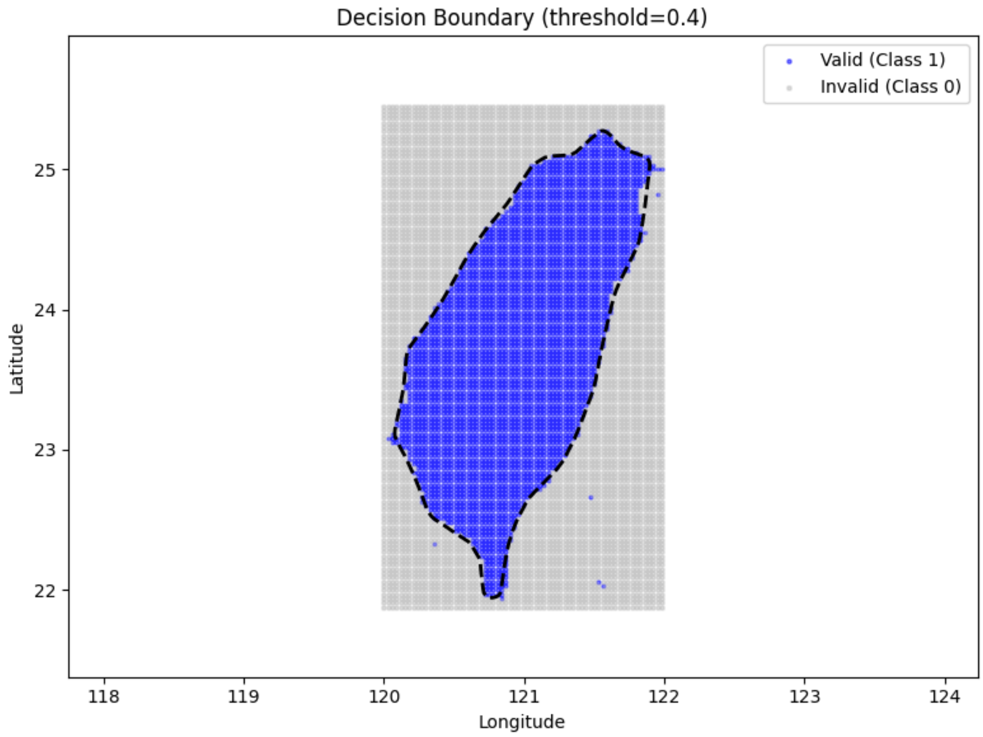

Figure 3. The decision boundary we obtained by piesewise smooth function model.

$R(\vec{x})$: 5 hidden layers, 64 neurons in each layer, and use ReLU activation function, Adam optimizer, MSE loss function. This model only trained on valid points(temperature $! = -999$).

```
with torch.no_grad():
    c_probs = torch.sigmoid(c_model(x).squeeze())
    c_pred = (c_probs >= threshold).float()
    r_pred = r_model(x).squeeze()
    h_pred = c_pred * r_pred + (1 - c_pred) * invalid_value
return h_pred
```

The above code is the way I build the combined function, where c_probs, c_pred, r_pred and h_pred are the probability, and prediction obtained from classification model, predicted temperature obtained from regression model and the last is the predicted temperature in this function.

Model Evaluation

This hybrid model did a good job on classification, with AUC = 0.9989, but it did terrible on regression one, I'm thinking if there exist any other way can learn the data better.

```
[Classification] Epoch 1/50 - Train Loss: 0.2942 | Val Loss: 0.1533 | Val AUC: 0.9920
[Classification] Epoch 5/50 - Train Loss: 0.0842 | Val Loss: 0.0933 | Val AUC: 0.9952
[Classification] Epoch 10/50 - Train Loss: 0.0681 | Val Loss: 0.1028 | Val AUC: 0.9948
[Classification] Epoch 15/50 - Train Loss: 0.0668 | Val Loss: 0.0625 | Val AUC: 0.9980
[Classification] Epoch 20/50 - Train Loss: 0.0648 | Val Loss: 0.0707 | Val AUC: 0.9975
[Classification] Epoch 25/50 - Train Loss: 0.0549 | Val Loss: 0.0570 | Val AUC: 0.9982
[Classification] Epoch 30/50 - Train Loss: 0.0551 | Val Loss: 0.0627 | Val AUC: 0.9980
[Classification] Epoch 35/50 - Train Loss: 0.0590 | Val Loss: 0.0659 | Val AUC: 0.9984
[Classification] Epoch 40/50 - Train Loss: 0.0527 | Val Loss: 0.0716 | Val AUC: 0.9974
[Classification] Epoch 45/50 - Train Loss: 0.0489 | Val Loss: 0.0666 | Val AUC: 0.9978
[Classification] Epoch 50/50 - Train Loss: 0.0459 | Val Loss: 0.0435 | Val AUC: 0.9988
[Classification] Test AUC: 0.9989
[Regression] Epoch 1/500 - Train Loss: 492.3066 | Val Loss: 480.6255 | Val MAE: 20.9236
[Regression] Epoch 50/500 - Train Loss: 13.2567 | Val Loss: 14.0261 | Val MAE: 2.8684
[Regression] Epoch 100/500 - Train Loss: 11.4251 | Val Loss: 12.2733 | Val MAE: 2.6437
[Regression] Epoch 150/500 - Train Loss: 9.2846 | Val Loss: 10.8563 | Val MAE: 2.3734
[Regression] Epoch 200/500 - Train Loss: 8.6209 | Val Loss: 10.3193 | Val MAE: 2.2870
[Regression] Epoch 250/500 - Train Loss: 8.2623 | Val Loss: 10.2542 | Val MAE: 2.3893
[Regression] Epoch 300/500 - Train Loss: 8.0199 | Val Loss: 9.7078 | Val MAE: 2.2092
[Regression] Epoch 350/500 - Train Loss: 7.8762 | Val Loss: 9.3977 | Val MAE: 2.1996
[Regression] Epoch 400/500 - Train Loss: 7.8622 | Val Loss: 9.2841 | Val MAE: 2.2322
[Regression] Epoch 450/500 - Train Loss: 7.6491 | Val Loss: 9.0230 | Val MAE: 2.1149
[Regression] Epoch 500/500 - Train Loss: 7.5601 | Val Loss: 9.0407 | Val MAE: 2.1468
[Regression] Test MAE: 10.883°C, RMSE: 15.912°C
```

Figure 4. The train loss, val loss and val AUC during training(classification). The train loss, val loss and val MAE during training(regression).

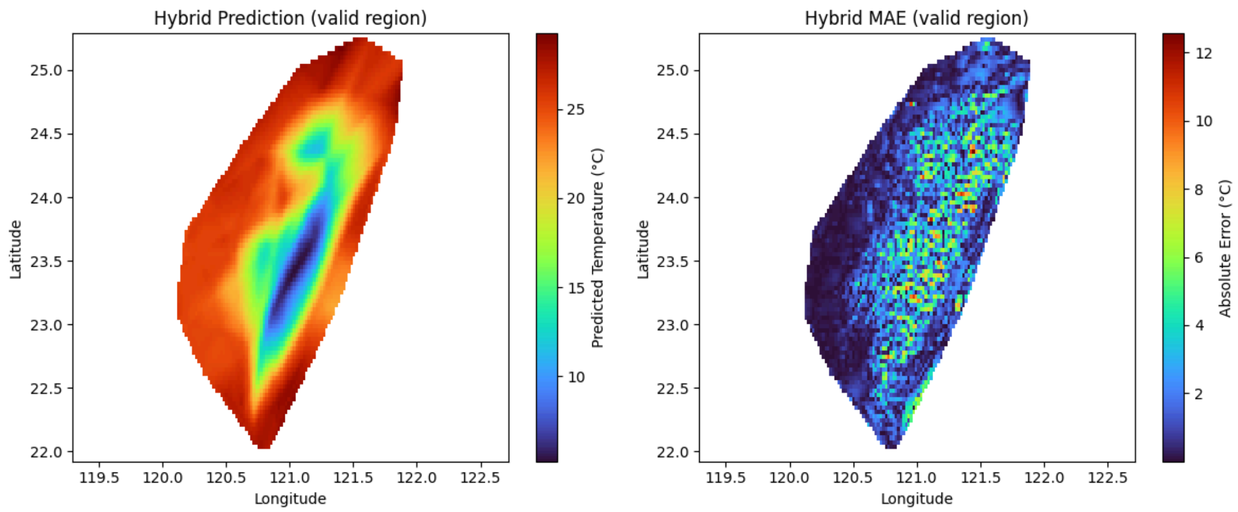

Figure 5. The prediction and the MAE between true and predicted temperature of the piesewise smooth function.