Machine Learning 2021 Homework 2

310511048 張奕廷

1. Classification Problem

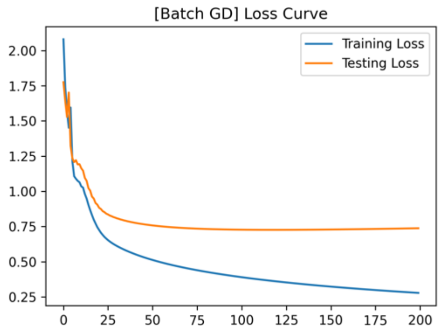
1. Least Squares for Classification

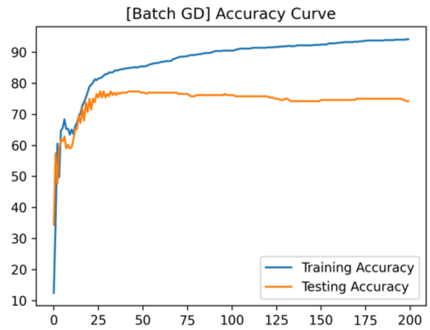
```
Train Accuracy: 1.0000, Train Loss:  0.0267
Test  Accuracy: 0.3008, Test  Loss: 35.0718
```

2. Logistic Regression

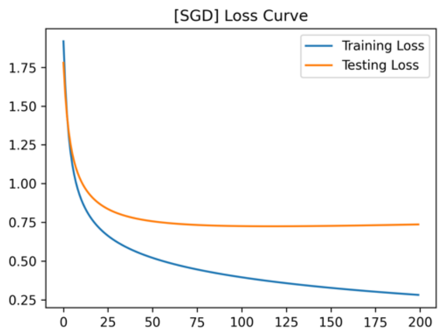
(a) Learning curves of the loss function and the accuracy.

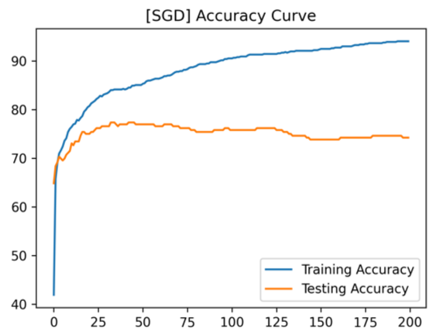
<u>Batch GD</u>



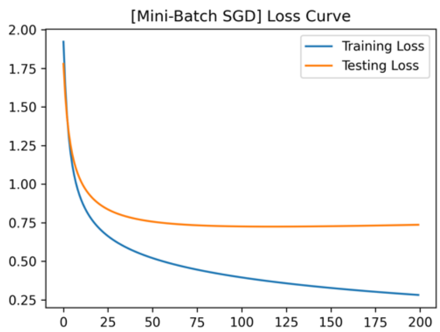


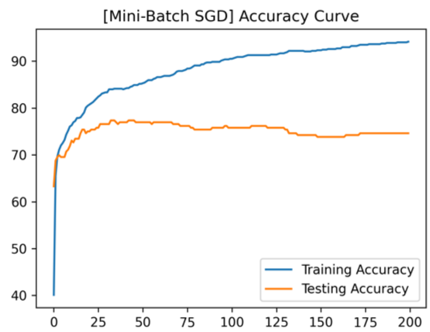
<u>SGD</u>





<u>Mini-Batch SGD</u>





(b) Final classification accuracy and loss value of training and testing data.

Batch GD

```
Train Acc: 94.7917, Train Loss:  0.2813
Test  Acc: 74.2188, Test  Loss:  0.7753
```

SGD

```
Train Acc: 94.1406, Train Loss:  0.2842
Test  Acc: 74.6094, Test  Loss:  0.7760
```

Mini-Batch SGD

```
Train Acc: 94.1406, Train Loss:  0.2841
Test  Acc: 74.6094, Test  Loss:  0.7759
```

(c) **Discussion.** The learning rate is set as 5×10^{-4}, and the batch size for Mini-Batch SGD is 32. I train all models for 200 epochs. From the learning curves shown above, I find that Batch GD has the most unstable behavior during the early stage of training process. On the other hand, SGD and Mini-Batch SGD behave similarly and are more stable during training. Also, Batch GD takes the shortest training time since it only updates the weight matrix once in an epoch, while SGD and Mini-Batch SGD update the weight matrix several times per epoch, thus requiring longer training time. However, I randomly split the training and test set several times and observe that the performance isn't always the best for any of the optimization algorithm.

3. **Discussion.** From the results in 1.1, we can observe that least squares model is severely overfitting, it correctly classifies every sample in the training set but acquires a terribly low accuracy on the test sets. I think this may due to the reasons that too many target classes are in this dataset, and these data has a relatively complex distribution. However, from the results in 1.2, logistic regression has outstanding performance comparing to least squares. Hence, I consider *we should apply more robust model such as logistic regression instead of least squares for classification problems*. Also, as the professor mentioned in the previous lecture, **least squares model is sensitive to outlier**, which means that the result may have a huge difference if the training set contains weird samples, this also offers us a glimpse of avoiding using least squares for classification.

2. Gaussian Process for Regression

1. Kernel function with polynomial basis function of order 2

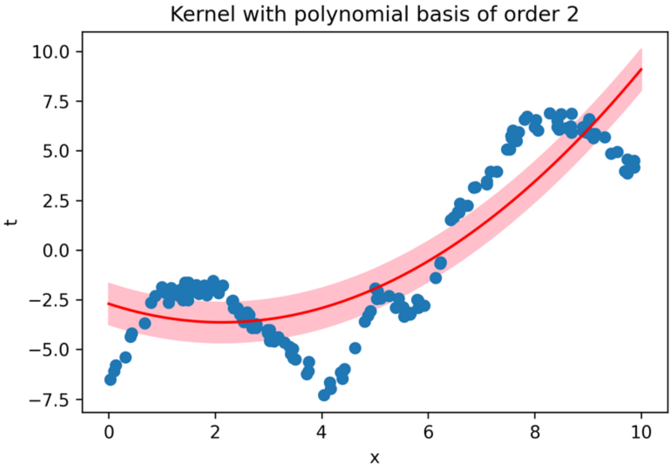
   Results are shown in 2.3 and 2.4.

2. Exponential-quadratic kernel function

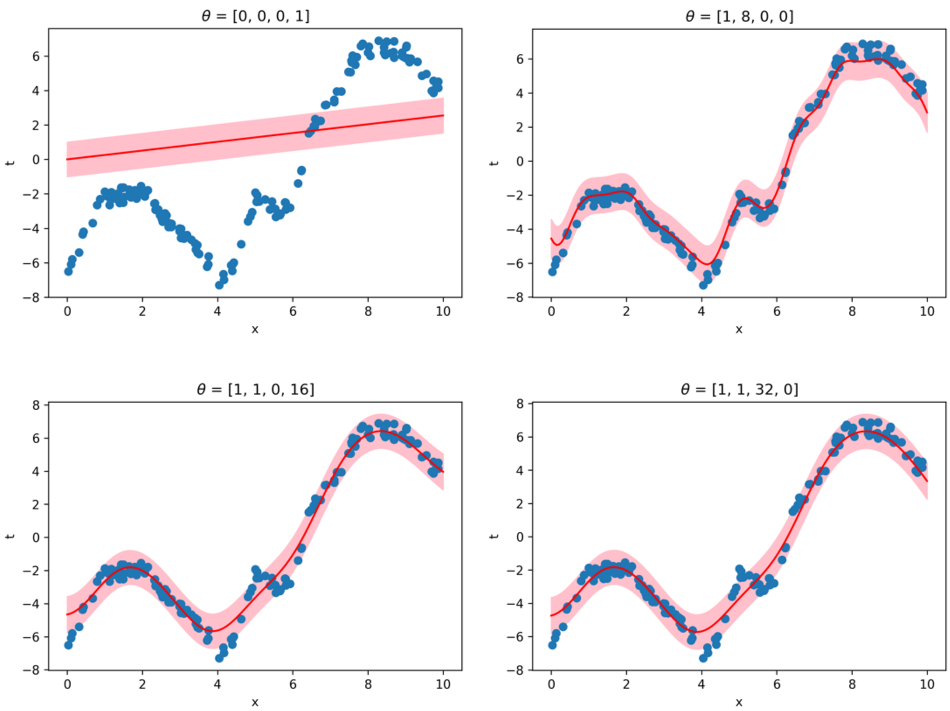
   Results are shown in 2.3 and 2.4.

3. Prediction results in 2.1 and 2.2

   Kernel function with polynomial basis function of order 2



Exponential-quadratic kernel function



4. Root-mean-square errors for both training and test sets in 2.1 and 2.2

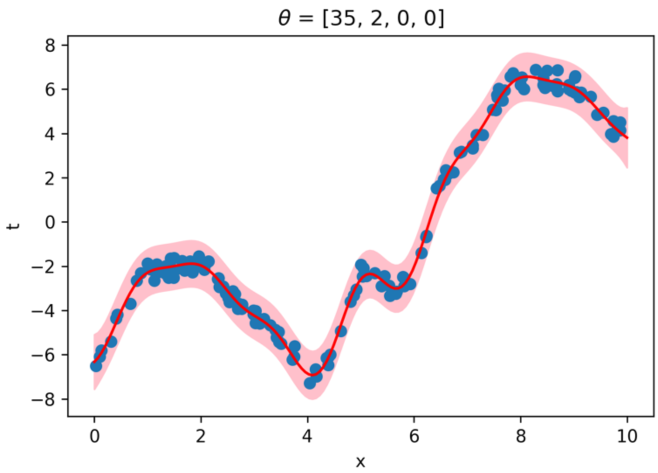
Kernel function with polynomial basis function of order 2

```
Training RMSE: 2.0664, Testing RMSE: 2.0971
```

Exponential-quadratic kernel function

```
Thetas: [ 0,  0,  0,  1], Training RMSE: 4.0827, Testing  RMSE: 3.9326
Thetas: [ 1,  8,  0,  0], Training RMSE: 0.4696, Testing  RMSE: 0.4930
Thetas: [ 1,  1,  0, 16], Training RMSE: 0.6045, Testing  RMSE: 0.5810
Thetas: [ 1,  1, 32,  0], Training RMSE: 0.6025, Testing  RMSE: 0.5866
```

5. I tune the hyperparameters θ in 2.2 by **trial and error**, and I find the best combination for this dataset is roughly $\theta = [35, 2, 0, 0]$.



Root-mean-square error for $\theta = [35, 2, 0, 0]$:

```
Training RMSE: 0.2737, Testing  RMSE: 0.3030
```

6. **Discussion.** From the results in 2.3, it is obvious that the first combination of θ has large root-mean-square error than the others, this is because it only remains the term $x_n^T x_m$ in the kernel function, which is too simple so that the model cannot fit well to the data. Also, the root-mean-square error between the third and fourth combination of θ only differs slightly, but the second one has a considerable improvement. This indicates that θ_2 and θ_3 don't have great impact to the model, while θ_1 can significantly influence the predicted results. Hence, I try to tune θ_0 and θ_1 in 2.5, and find out that the error decreases as both θ_0 and θ_1 increases. However, the test error starts to increase again when θ_0 exceeds about 35 or when θ_1 exceeds about 2, and the training error is still decreasing, which means that the model is overfitting.