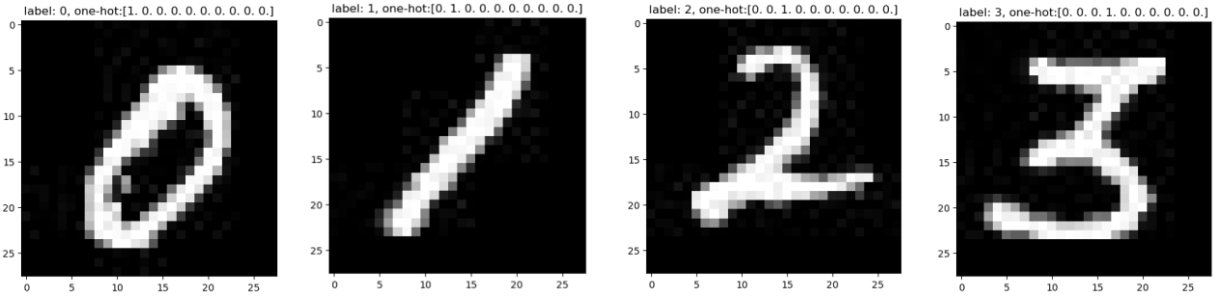
2022 NYCU Machine Learning – Homework2 Report

IEE 陳冠瑋 310510221 / Nov.23, 2022

1 Bayesian Linear Regression

1-of-*K* binary coding scheme for the target vector t



Random select 32 images as test data for each class and the remaining images as training data

```
Number of train data for each class:  [96.0, 96.0, 96.0, 96.0, 96.0, 96.0, 96.0, 96.0, 96.0, 96.0]
Number of test data for each class:  [32.0, 32.0, 32.0, 32.0, 32.0, 32.0, 32.0, 32.0, 32.0, 32.0]
```

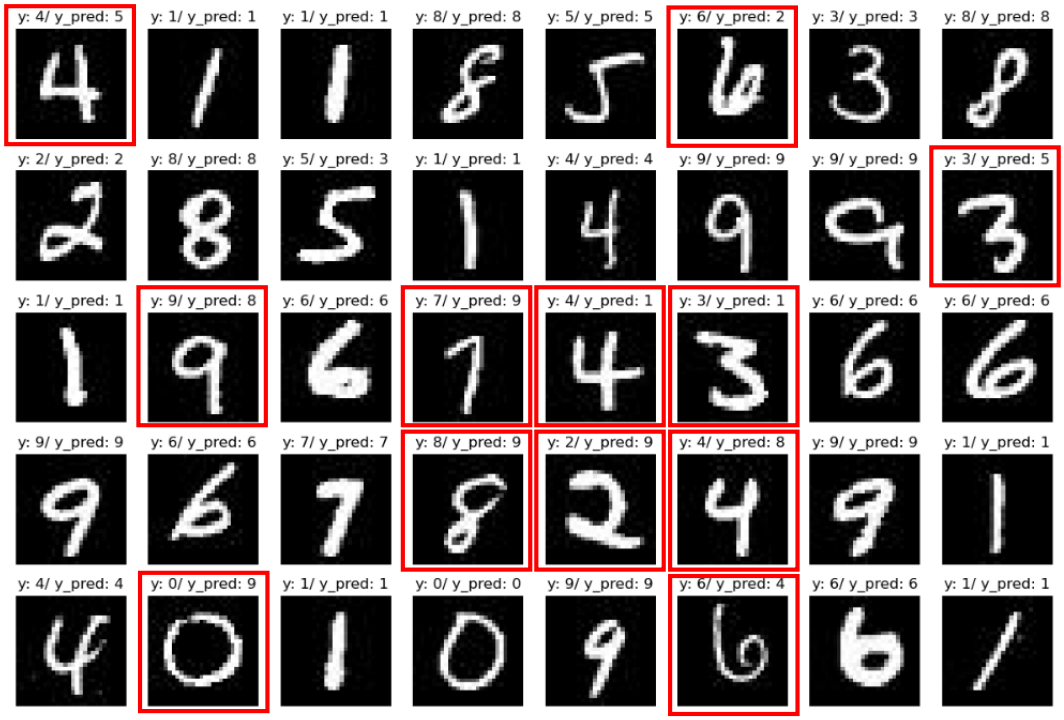
1.1 Least squares for classification

$$\min_{w} \sum_{i=1}^{n} (w^T x^{(i)} - y^{(i)})^2 \mathrm{w}$$

$$\text{Closed} - \text{form solution: } \mathrm{w} = (X^T \mathrm{X})^{-1} X^T y = A^+ y$$

```
X_pinv = np.linalg.pinv(X)
W = np.matmul(X_pinv, t)
```

Predict Result (testing data)



Training & Testing L2 loss / Accuracy

```
[Training data]
Least Square Solution - L2 loss: 61.1699
Least Square Solution - Accuracy: 100.0000 %
-----------------------------------------
[Testing data]
Least Square Solution - L2 loss: 5451.0184
Least Square Solution - Accuracy: 52.8125 %
```

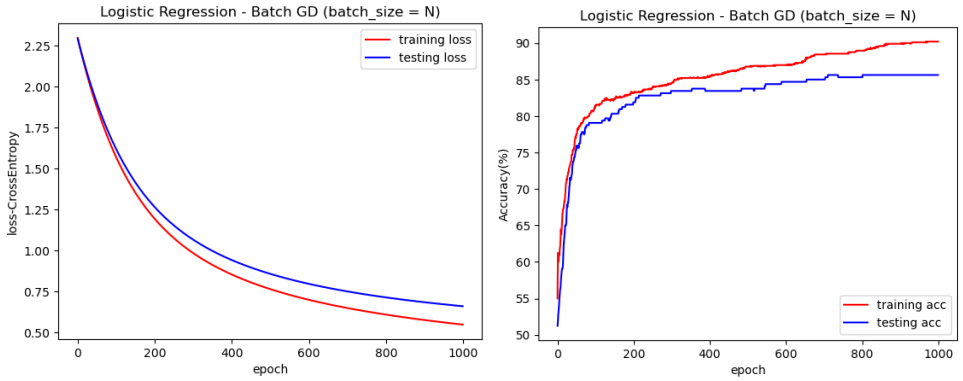
1.2 Logistic regression

Algorithms	Batch size	Iterations in one epoch
batch GD	N	1
SGD	1	N
mini-batch SGD	B	N/B

N: number of training data, B: batch size (can be selected by yourself)

Batch GD (Batch Gradient Descent)

(a)

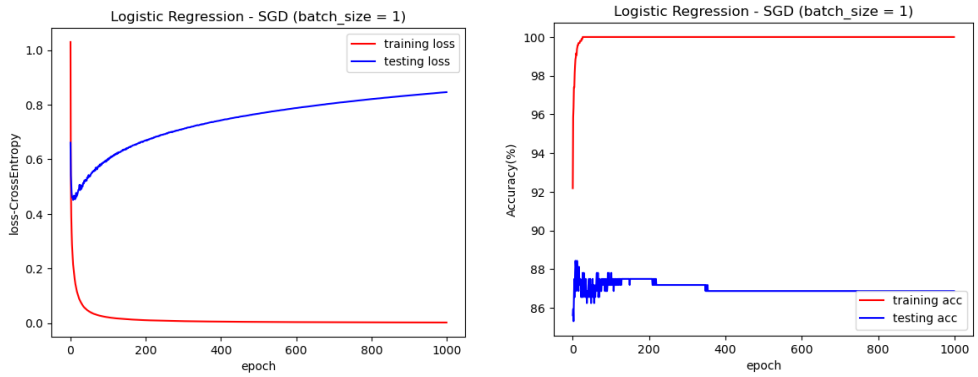


(b)

```
[Training]
Logistic Regression w/ BGD - Final Loss: 0.5494
Logistic Regression w/ BGD - Final Accuracy: 90.2083 %
-------------------------------------------
[Testing]
Logistic Regression w/ BGD - Final Loss: 0.6615
Logistic Regression w/ BGD- Final Accuracy: 85.6250 %
```

SGD (Stochastic Gradient Descent)

(a)

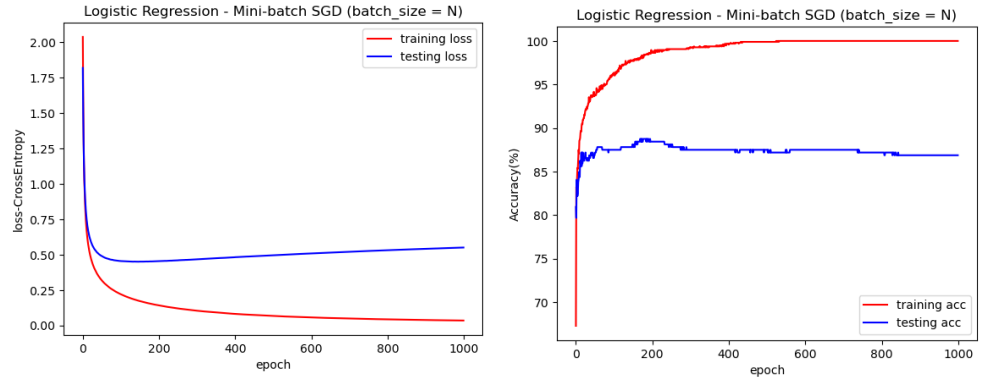


(b)

```
[Training]
Logistic Regression w/ SGD - Final Loss: 0.0022
Logistic Regression w/ SGD - Final Accuracy: 100.0000 %
-------------------------------------------
[Testing]
Logistic Regression w/ SGD - Final Loss: 0.8460
Logistic Regression w/ SGD- Final Accuracy: 86.8750 %
```

Mini-Batch SGD (batch_size = 4)

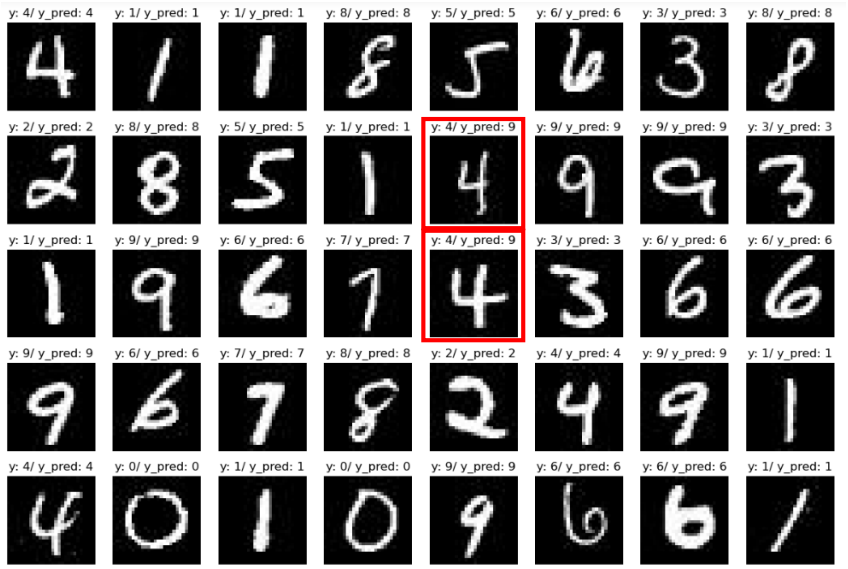
(a)



(b)

```
[Training]
Logistic Regression w/ MBGD - Final Loss: 0.0344
Logistic Regression w/ MBGD - Final Accuracy: 100.0000 %
-------------------------------------------
[Testing]
Logistic Regression w/ MBGD - Final Loss: 0.5503
Logistic Regression w/ MBGD- Final Accuracy: 86.8750 %
```

Predict Result (testing data)



(c) Based on your observation about the different algorithms (batch GD, SGD, and mini-batch SGD), please **make some discussion**.

Gradient descent(梯度下降法)是一種透過迭代來優化並更新參數的機器學習算法，用於降低 cost function，以便訓練出能夠更精準預測的模型。Batch GD 是一個 epoch 使用全部的 data 去計算損失函數，並更新一次參數(一個 epoch 更新一次參數)。SGD 是一次使用一個樣本(example)，然後算出損失函數，並更新參數(一筆 example 更新一次參數)。Mini-batch GD 就是介於 Batch GD 與 SGD 之間，會設定一個 batch size，一次拿 batch size 大小的 data 數量去計算損失函數，然後更新一次參數。

由於 Batch GD 是一個 epoch 更新一次參數，SGD 是每看一筆資料就更新一次參數。因此一個 epoch 的訓練時間，SGD 執行時間會比 Batch GD 來得長。

訓練一個 epoch 的所需的時間：SGD > Mini-batch GD > Batch GD

由於 SGD 是針對個體資料去對模型做優化，因此在學習率(learning rate)太大時，容易造成參數更新呈現鋸齒狀，learning curve 的 loss 與 accuracy 會有較大的震盪，除此之外，容易導致 overfitting 使模型泛化能力下降。Batch GD 是看完整個 dataset 之後做優化，因此 learning curve 會比較平滑，訓練比較有效率，模型泛化能力高，能夠適應 testing data。

模型收斂效率：Batch GD > Mini-batch GD > SGD

模型泛化能力：Batch GD > Mini-batch GD > SGD

由於 SGD 是一筆資料更新一次，所以一個 epoch 會更新比 Batch GD 還多次。因此，在相同 epoch 下 SGD 收斂較快。

相同 epoch 下模型收斂速度：SGD > Mini-batch GD > Batch GD

1.3 **Make some discussion** about the difference between the results of 1.1 and 1.2.

Least squares method(最小平方法)主要要是透過最小化誤差的平方來求解出最佳參數。假設 X 有 Full rank，就有 closed-form soluton，但現實是 equation 的數量往往大於未知數的數量(m > n)，導致沒有解。所以利用矩陣運算，求出 x 的 psuedo inverse 來反推原本的線性方程式中的解。由於 Least squares method 是去擬合 traing dataset，再加上由於全部的運算都是線性的轉換，因此無法

學習到非線性關係。所以導致在 training 上有很好的結果，testing 的表現就很差，泛化能力差。

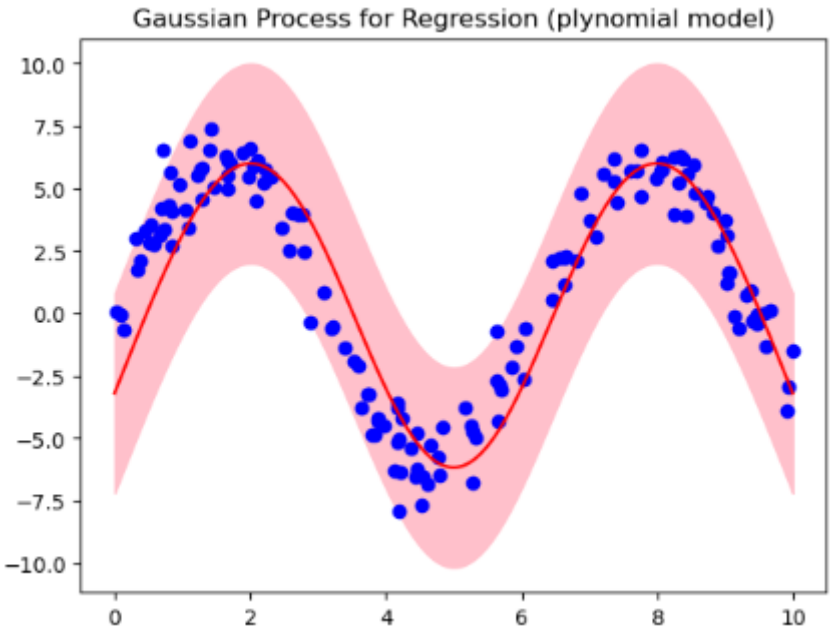
Gradient descent(梯度下降法)是透過迭代來尋找局部最佳解，透過一步一步更新權重來優化模型。使用 losgistic regression 使用 softmax 非線性函數，所以在非線性資料的表達上會比 Least squares method 還來的好一點。因此整體來看在 testing dataset 上，使用 logist regressin model 搭配 gradient descent 表現比 Least squares method 來的好。

```
[Training data]
Least Square Solution - L2 loss: 58.1939
Least Square Solution - Accuracy: 100.0000 %
-----------------------------------------
[Testing data]
Least Square Solution - L2 loss: 21908.7864
Least Square Solution - Accuracy: 45.0000 %
```

```
[Training]
Logistic Regression w/ MBGD - Final Loss: 0.0344
Logistic Regression w/ MBGD - Final Accuracy: 100.0000 %
-----------------------------------------
[Testing]
Logistic Regression w/ MBGD - Final Loss: 0.5503
Logistic Regression w/ MBGD- Final Accuracy: 86.8750 %
```

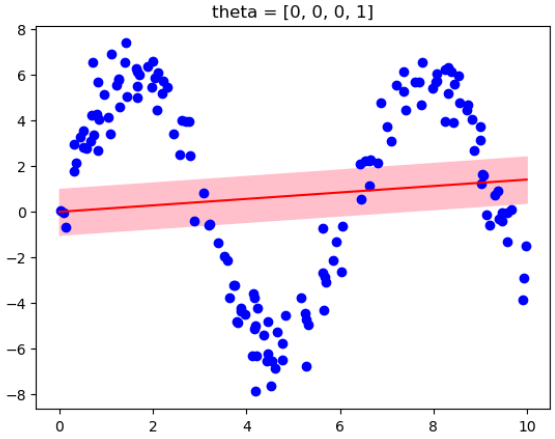
2 Gaussian Process for Regression

1.

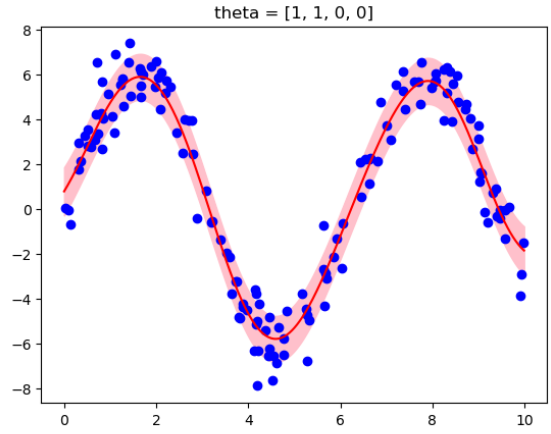



```
[RMS Error]
Training data:  1.0344234037145343
Testing data:   26.58374398546768
```

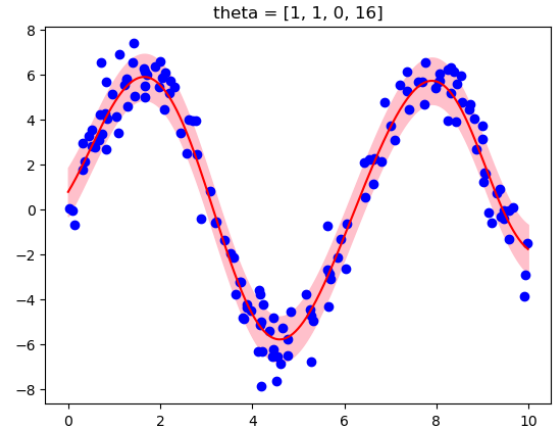
2.



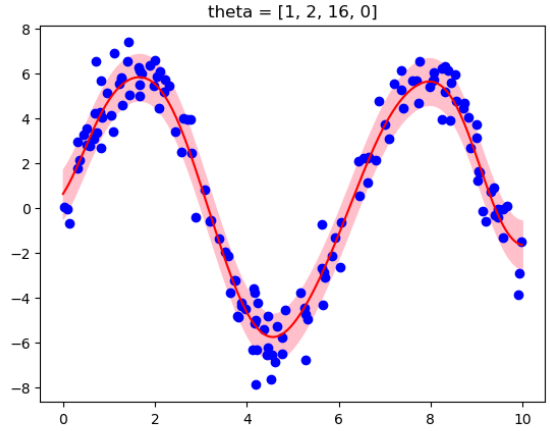

```
theta = [0, 0, 0, 1]
[RMS Error]
Training data:  4.33557961185875
Testing data:  4.378509014389943
```




```
theta = [1, 1, 0, 0]
[RMS Error]
Training data:  0.9563642676105912
Testing data:  0.9600463323661679
```

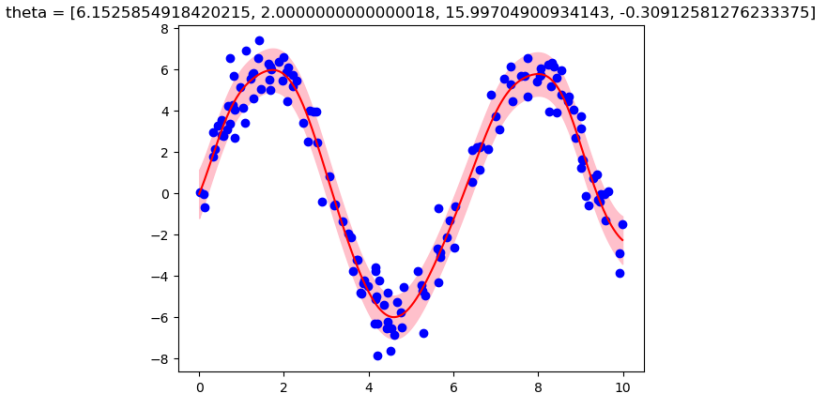



```
theta = [1, 1, 0, 16]
[RMS Error]
Training data:  0.9572824402560011
Testing data:  0.9624735743601458
```




```
theta = [1, 2, 16, 0]
[RMS Error]
Training data:  0.9504982228584732
Testing data:  0.9707675411307138
```

3.




```
theta = [6.1525854918420215, 2.0000000000000018, 15.99704900934143, -0.30912581276233375]
[RMS Error]
Training data:  0.9146228745326511
Testing data:  0.9227026013628404
```

4.

根據以上實驗可以發現：

首先，使用 Exponential-quadratic kernel function 配合適當的 hyperparameters，在本次實驗最終結果是優於使用 Polynomail model。另外，當 theta 越複雜時，可以更擬合 training data，且 RMS error 也會跟著變低。其中，第 2 組與第 3 組的表現差不多，多了一個 theta4 卻對結果沒造成太大的影響。最後，使用 Automatic relevance determination (ARD)，可以對 hyperparameter 進行 fine-tune，讓模型有更好的結果。