CS 640 Programming Assignment #1

Ziqi Tan

U88387934

October 8, 2019

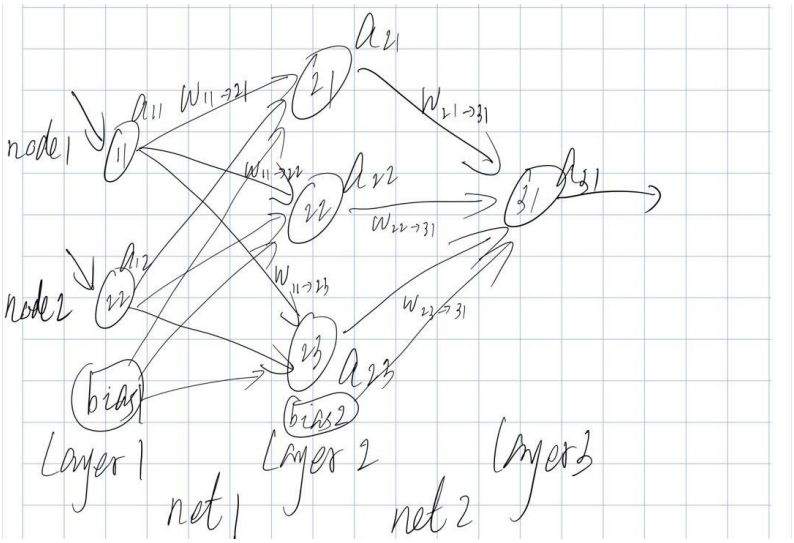
目录

Problem Definition

Given a dataset with X and Y, find a reflection f which satisfies Y = f(X).

Method and Implementation

Train a neural network with one hidden layer.



1. A brief workflow of how your neural network classifies data.

1. Get data from dataset.
2. Split data into k folds for cross validation.
3. Initialize model parameters (weight matrices, number of neurons in the hidden layer, epochs, lambda for regularization and learning rate).
4. For each fold of data:
    a) For each epoch:
        i. For each sample:
            1. Do forward propagation;
            2. Do back propagation;
5. Test the model.
6. Get result image, confusion matrix and performance scores.

2. An explanation of how your forward propagation works. (cite your code)

```
def forward(self, X):
    # Perform matrix multiplication and activation twice (one for each layer).
    # (hint: add a bias term before multiplication)
    # Add bias to X
    bias = np.ones((1, 1))
    X = np.concatenate((X, bias), axis=1)
    X = np.transpose(X)

    self.net1 = np.dot(self.w1, X)
    self.z1 = self.activate(self.net1)

    # Add bias to z1
    self.z1 = np.concatenate((self.z1, bias))
    # z1 is 5 by 1
    self.net2 = np.dot(self.w2, self.z1)
    self.z2 = self.activate(self.net2)

    return self.z2
```

3. An explanation of how your back propagation works. (cite your code)

```
def backpropagate(self, X, YTrue):
    # Compute loss / cost using the getCost function.
    # cost = self.getCost(YTrue, self.z2)

    # Compute gradient for each layer.
    YTrue = np.transpose(YTrue)
    diff = self.z2 - YTrue
    dloss_dw2 = np.dot(diff, np.transpose(self.z1))

    diff_trans = np.transpose(diff)
    d_activate = self.deltaActivate(self.z1)[:-1]
    bias = np.ones((1, 1))
    X = np.concatenate((X, bias), axis=1)
    dloss_dw1 = np.dot(np.multiply(np.transpose(np.dot(diff_trans, self.w2))[:-1], d_activate), X)

    # Update weight matrices.
    self.w2 = self.w2 - self.learningRate * dloss_dw2 - self.learningRate * self.regLambda * self.w2
    self.w1 = self.w1 - self.learningRate * dloss_dw1 - self.learningRate * self.regLambda * self.w1

    pass
```

Experiments

Dataset 1

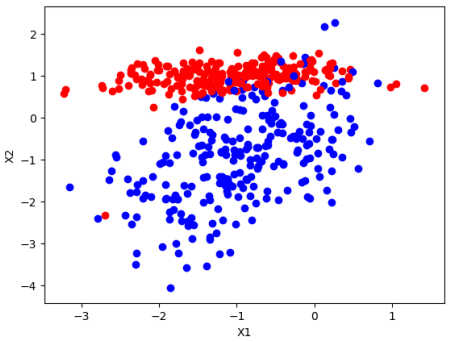
In dataset1, there are two binary classification problems, one of which is a linear classification problem and the other of which is a non-linear classification problem. Testing data is not given and cross validation is necessary.

Dataset2

In dataset2, there are 10 categories of data sample, which is also a classification problem. A training dataset and a testing dataset are provided separately. Thus, train the model with all the training data and test the model with the given testing data. Cross validation is not applied here.

Results

Dataset1 linear X-Y



Source image

Parameters

- Number of neurons in hidden layer: 4
- Lambda for regularization: 0.0000001
- Learning rate: 0.4
- Number of epochs: 100
- Cross validation k: 5
- Weight initialization: uniform distribution [-2,2]

In cross validation, we split the data into 5 folds: data0, data1, data2, data3 and data4.

Fold 1

Training data: data1, data2, data3 and data4.

Testing data: data0.

1. **Confusion matrix**

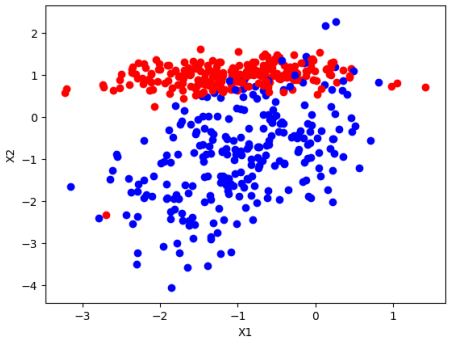
Confusion matrix

	Positive	Negative
True	41	54
False	0	5

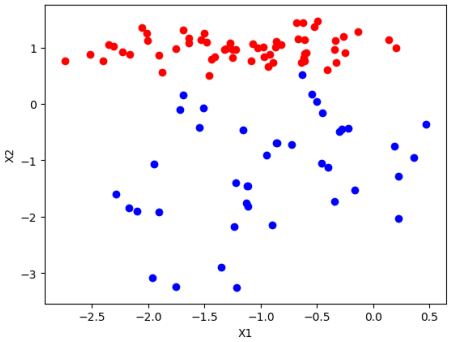
2. **Performance**

- Accuracy: 0.95
- Precision: 1.0
- Recall: 0.8913043478260869
- F1score: 0.9425287356321839

3. Visualization



Source image



Result

Fold 2

Training data: data0, data2, data3 and data4.
Testing data: data1.

1. Confusion matrix

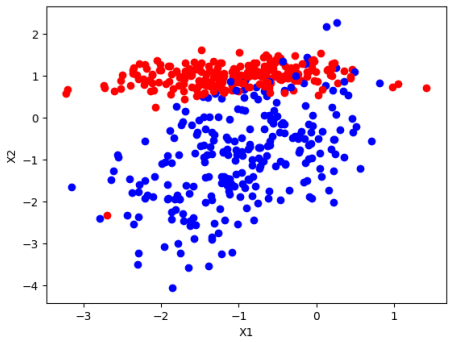
Confusion matrix

	Positive	Negative
True	47	44
False	2	7

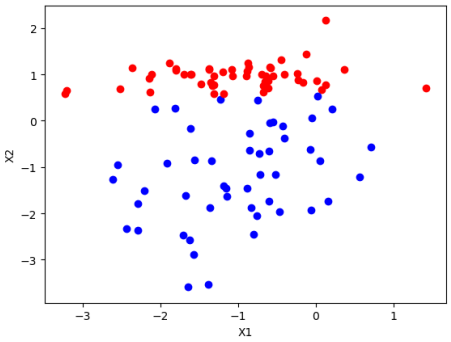
2. Performance

- Accuracy: 0.91
- Precision: 0.9591836734693877
- Recall: 0.8703703703703703
- F1score: 0.912621359223301

3. Visualization



Source image



Result

Fold 3

Training data: data0, data1, data3 and data4.

Testing data: data2.

1. **Confusion matrix**

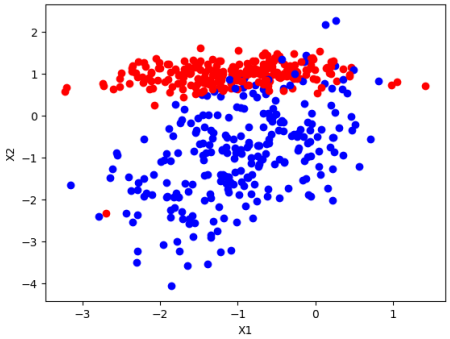
Confusion matrix

	Positive	Negative
True	37	59
False	2	2

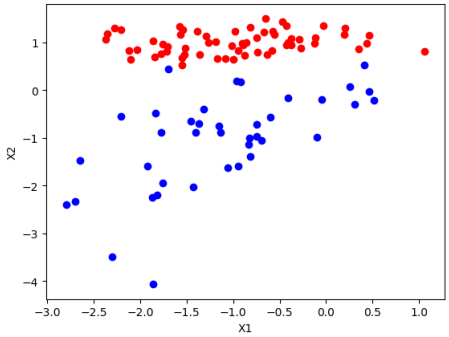
2. **Performance**

- Accuracy: 96.00%
- Precision: 94.87%
- Recall: 94.87%
- F1score: 94.87%

3. **Visualization**



Source image



Result

Fold 4

Training data: data0, data1, data2 and data4.

Testing data: data3.

1. **Confusion matrix**

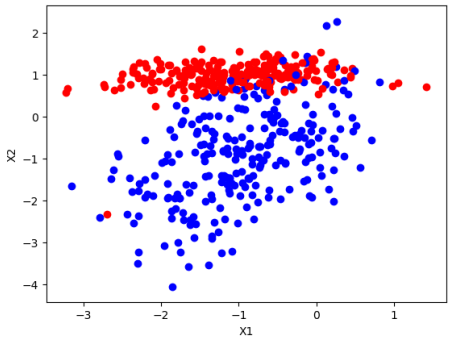
Confusion matrix

	Positive	Negative
True	47	44
False	0	9

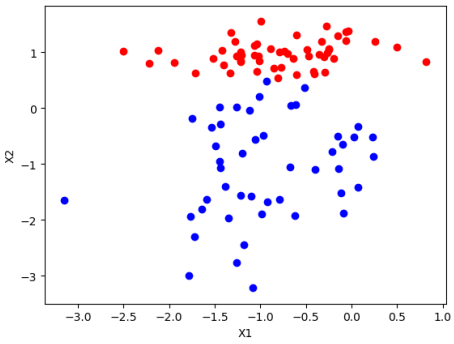
2. **Performance**

- Accuracy: 91.00%
- Precision: 100.00%
- Recall: 83.93%
- F1score: 91.26%

3. **Visualization**



Source image



Result

Fold 5

Training data: data0, data1, data2 and data3.

Testing data: data4.

1. **Confusion matrix**

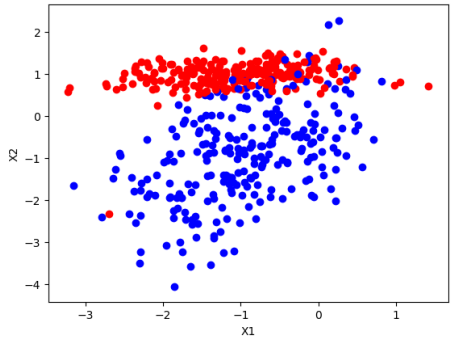
Confusion matrix

	Positive	Negative
True	47	45
False	0	8

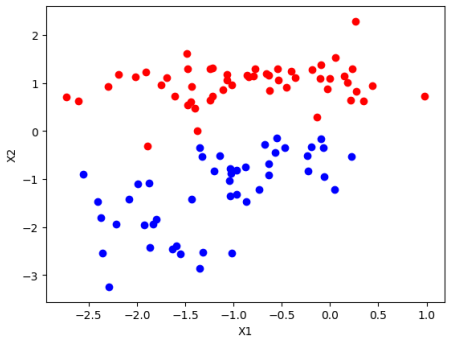
2. **Performance**

- Accuracy: 92.00%
- Precision: 100.00%
- Recall: 85.45%
- F1score: 92.16%

3. **Visualization**



Source image



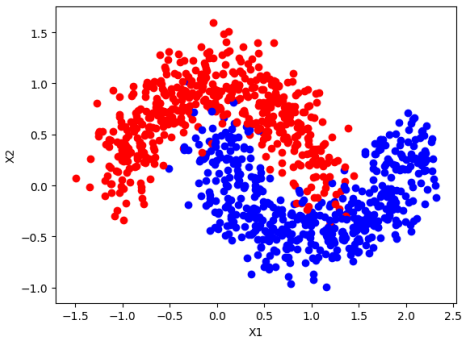
Result

Overall performance

- Accuracy: 93%
- Precision: 98.16%
- Recall: 88.08%

- F1score: 92.76%

Dataset1 Non-linear X-Y



Source image

Different number of neurons

Let **NNodes** be the number of neurons in the hidden layer.

In this part, number of neurons in hidden layer will be various. These numbers will be shown as follows: 75, 5, 10, 20, 30.

Parameters

- Lambda for regularization: 0.0000001
- Learning rate: 0.01
- Number of epochs: 300
- Cross validation k: 5
- Weight initialization: normal distribution N(0,3)

Result:

Lambda	5	10	20	30	75
Accuracy	89.00%	89.60%	92.20%	95.30%	95.80%

NNodes: 75

Fold 1:

Training data: data1, data2, data3 and data4.

Testing data: data0.

1. **Confusion matrix**

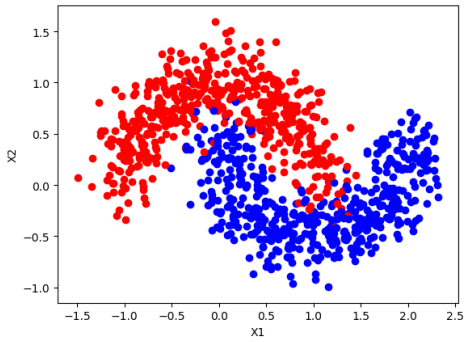
Confusion matrix

	Positive	Negative
True	95	98
False	4	3

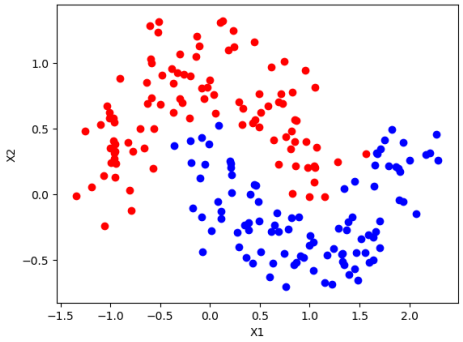
2. **Performance**

- Accuracy: 96.50%
- Precision: 95.96%
- Recall: 96.94%
- F1score: 96.45%

3. Visualization



Source image

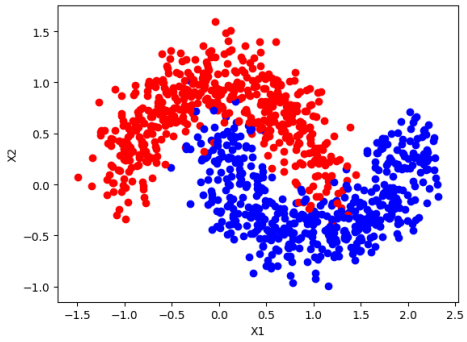


Result

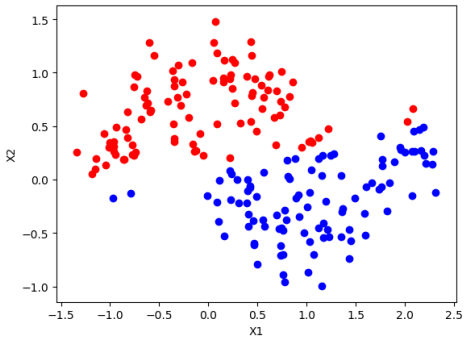
Overall performance

- Accuracy: 95.80%
- Precision: 95.65%
- Recall: 96.04%
- F1score: 95.82%

NNodes: 5



Source image

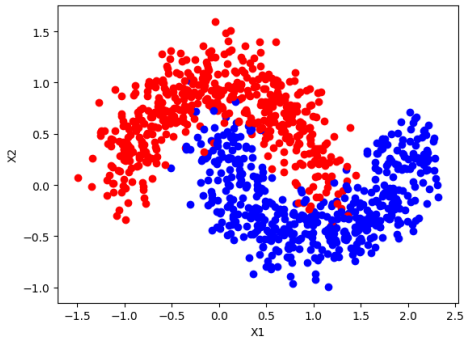


Result

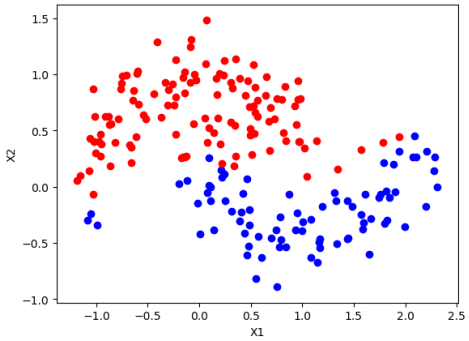
Overall performance:

- Accuracy: 89.00%
- Precision: 88.93%
- Recall: 89.20%
- F1score: 88.97%

NNodes: 10



Source image

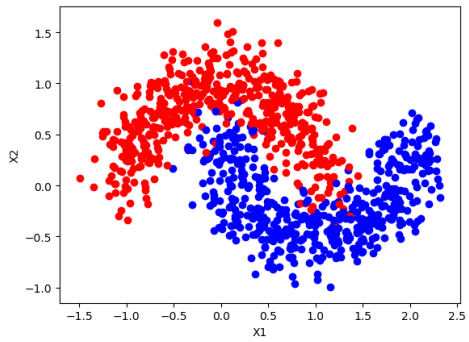


Result

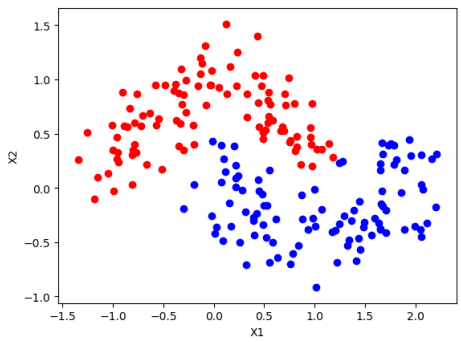
Overall performance

- Accuracy: 89.60%
- Precision: 90.75%
- Recall: 88.36%2
- F1score: 89.38%

NNodes: 20



Source image

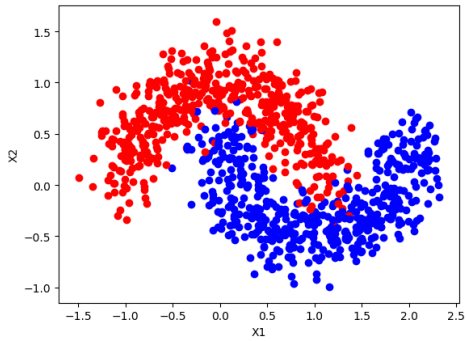


Result

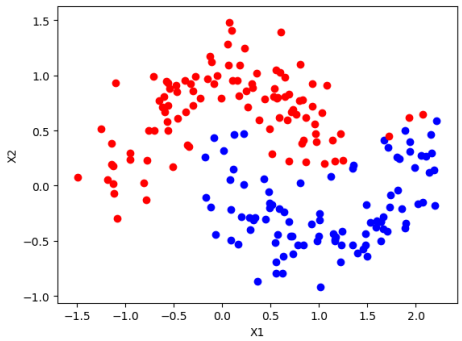
Overall performance:

- Accuracy: 92.20%
- Precision: 90.80%
- Recall: 94.34%
- F1score: 92.49%

NNodes: 30



Source image



Result

Overall performance:

- Accuracy: 95.30%
- Precision: 94.96%
- Recall: 95.66%
- F1score: 95.29%

Different Lambda for regularization

Lambda for regularization: 0.0000001

Parameters

- Number of neurons in hidden layer: 75
- Learning rate: 0.01
- Number of epochs: 300
- Cross validation k: 5
- Weight initialization: normal distribution N(0,3)

Result:

Lambda	0.001	0.0001	0.00001	0.000001	0.0000001
Accuracy	91.80%	93.10%	95.40%	95.30%	95.80%

Fold 1:

Training data: data1, data2, data3 and data4.

Testing data: data0.

1. **Confusion matrix**

Confusion matrix

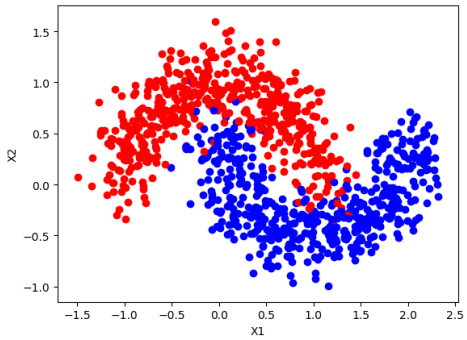
	Positive	Negative
True	95	98
False	4	3

2. **Performance**

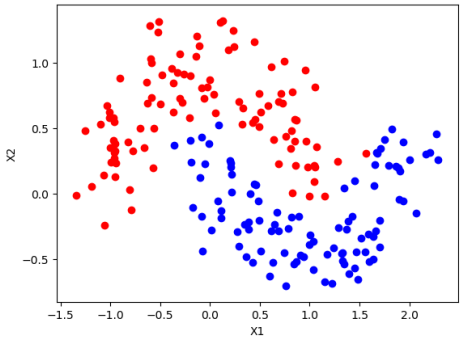
- Accuracy: 96.50%
- Precision: 95.96%

- Recall: 96.94%
- F1score: 96.45%

3. Visualization



Source image

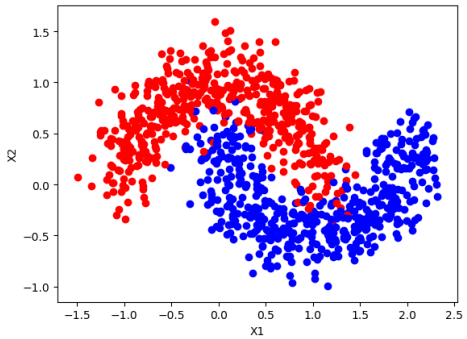


Result

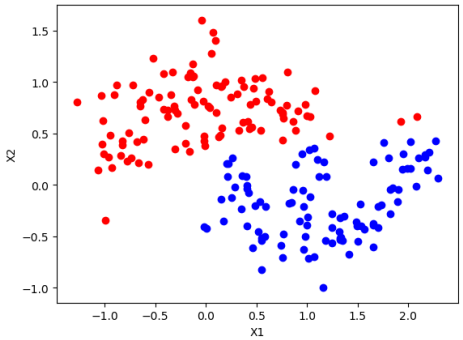
Overall performance

- Accuracy: 95.80%
- Precision: 95.65%
- Recall: 96.04%
- F1score: 95.82%

Lambda: 0.001



Source image

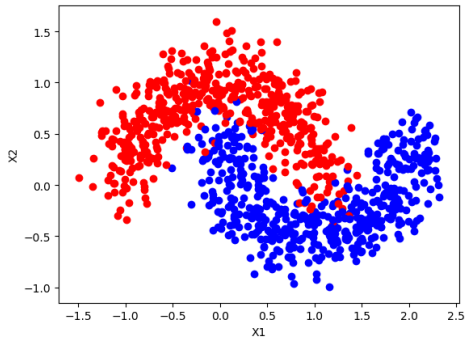


Result

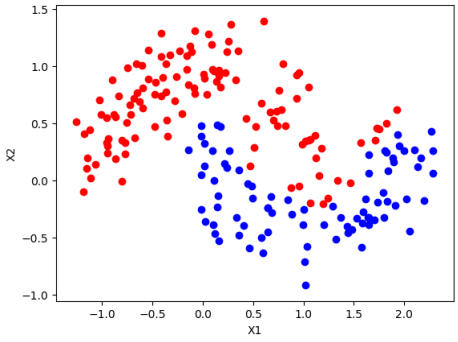
Overall performance:

- Accuracy: 91.80%
- Precision: 91.03%
- Recall: 92.50%
- F1score: 91.72%

Lambda: 0.0001



Source image

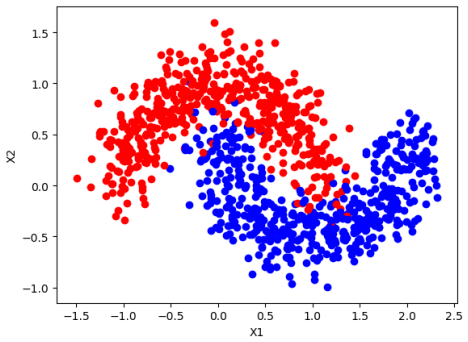


Result

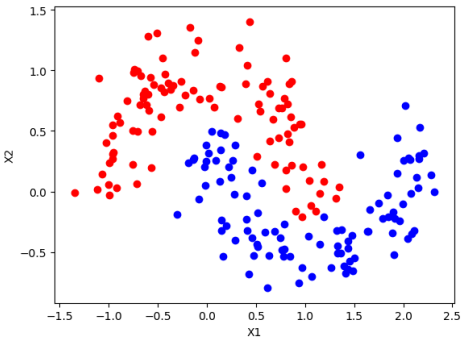
Overall performance:

- Accuracy: 93.10%
- Precision: 97.31%
- Recall: 88.57%
- F1score: 92.63%

Lambda: 0.00001



Source image

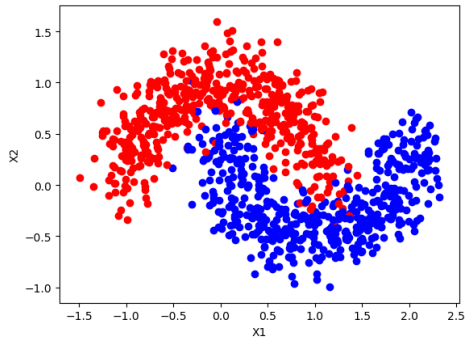


Result

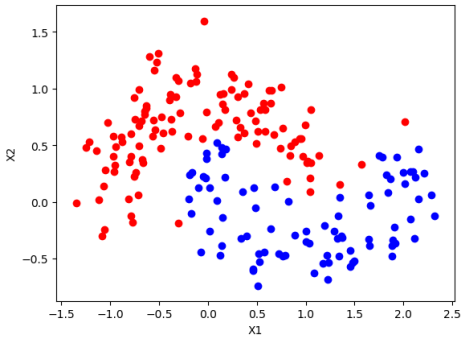
Overall performance:

- Accuracy: 95.40%
- Precision: 97.35%
- Recall: 93.49%
- F1score: 95.29%

Lambda: 0.000001



Source image



Result

Overall performance:

- Accuracy: 95.30%
- Precision: 95.91%
- Recall: 94.68%
- F1score: 95.25%

Different learning rate

Learning rate: 0.01

Parameters

- Number of neurons in hidden layer: 75
- Lambda for regularization: 0.0000001
- Number of epochs: 300
- Cross validation k: 5
- Weight initialization: normal distribution N(0,3)

Result:

Learning Rate	0.01	0.05	0.1	0.25	0.35	0.5
Accuracy	95.80%	95.60%	95.30%	95.50%	90.9%	84.60%

Fold 1:

Training data: data1, data2, data3 and data4.

Testing data: data0.

4. Confusion matrix

Confusion matrix

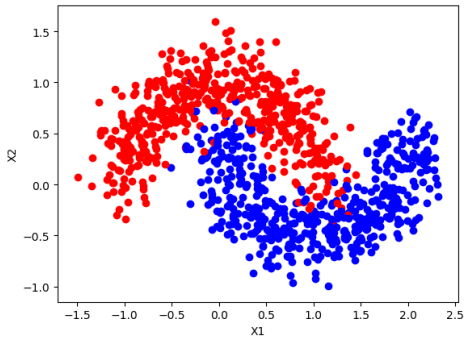
	Positive	Negative
True	95	98
False	4	3

5. Performance

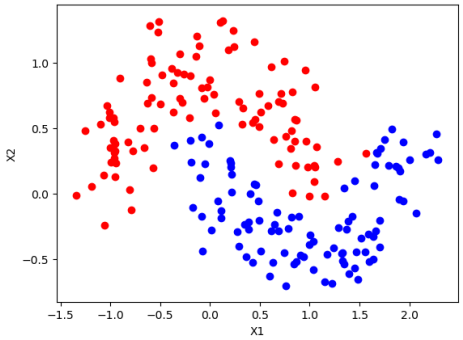
- Accuracy: 96.50%
- Precision: 95.96%

- Recall: 96.94%
- F1score: 96.45%

6. Visualization



Source image

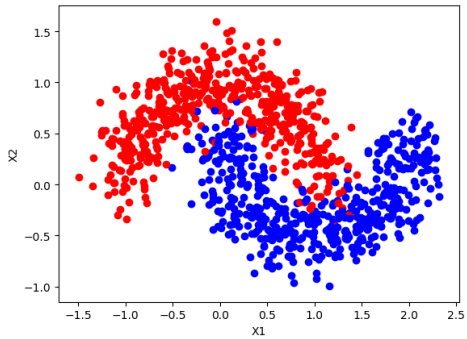


Result

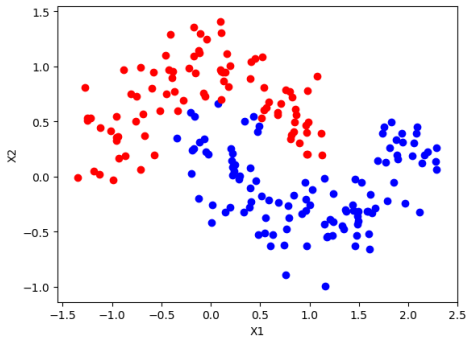
Overall performance

- Accuracy: 95.80%
- Precision: 95.65%
- Recall: 96.04%
- F1score: 95.82%

Learning Rate: 0.05



Source image

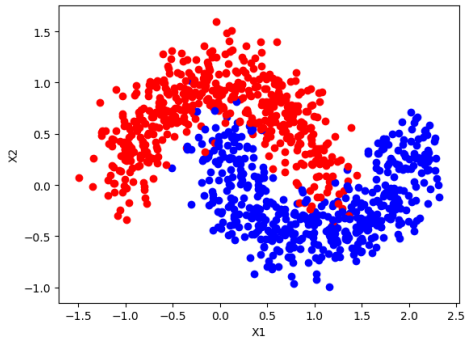


Result

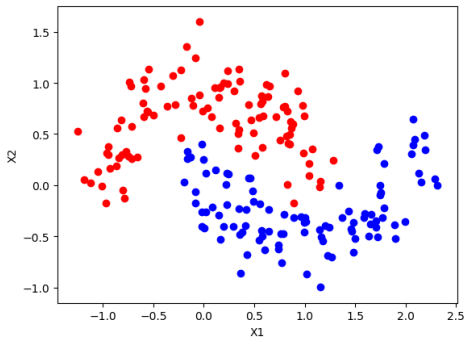
Overall performance:

- Accuracy: 95.60%
- Precision: 93.09%
- Recall: 98.59%
- F1score: 95.75%

Learning Rate: 0.1



Source image

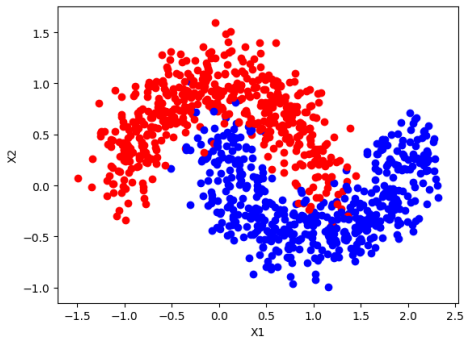


Result

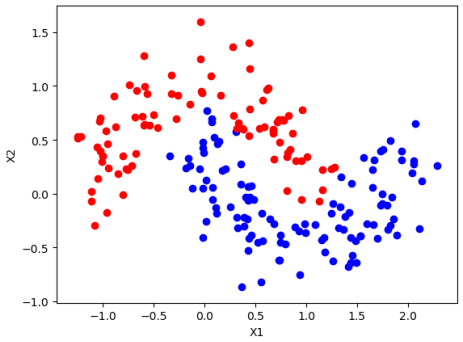
Overall performance:

- Accuracy: 95.30%
- Precision: 96.82%
- Recall: 93.50%
- F1score: 95.10%

Learning Rate: 0.25



Source image

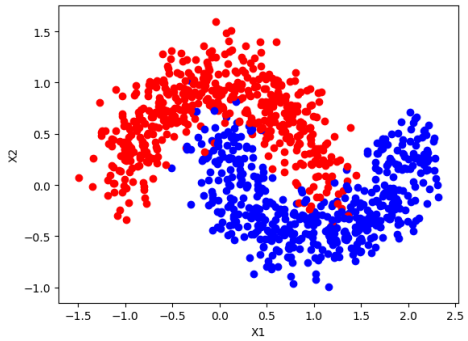


Result

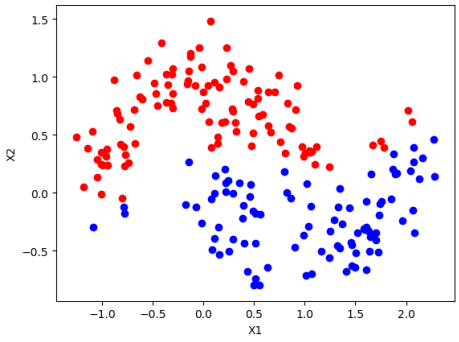
Overall performance:

- Accuracy: 95.50%
- Precision: 94.78%
- Recall: 96.24%
- F1score: 95.41%

Learning Rate: 0.35



Source image

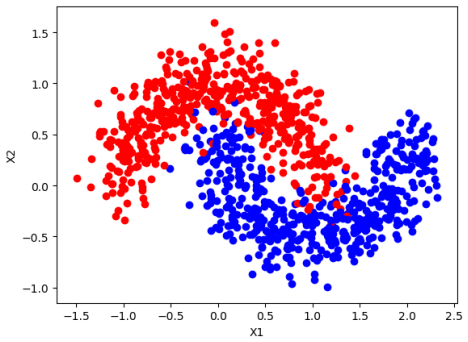


Result

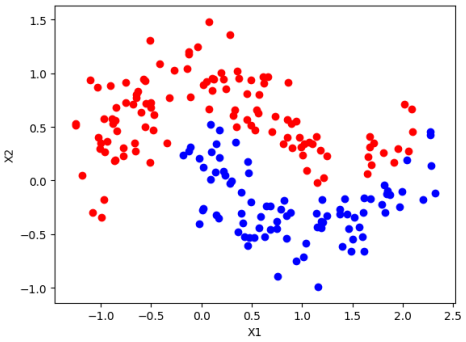
Overall performance:

- Accuracy: 90.90%
- Precision: 89.46%
- Recall: 92.98%
- F1score: 91.01%

Learning Rate: 0.5



Source image



Result

Overall performance:

- Accuracy: 84.60%
- Precision: 90.82%
- Recall: 77.75%
- F1score: 83.13%

Different epochs

Number of epochs: 300

Parameters

- Number of neurons in hidden layer: 75
- Lambda for regularization: 0.0000001
- Learning rate: 0.01
- Cross validation k: 5
- Weight initialization: normal distribution N(0,3)

Result:

Epochs	50	100	300	400	500
Accuracy	92.70%	93.40%	95.80%	96.60%	96.10%

Fold 1:

Training data: data1, data2, data3 and data4.

Testing data: data0.

1. Confusion matrix

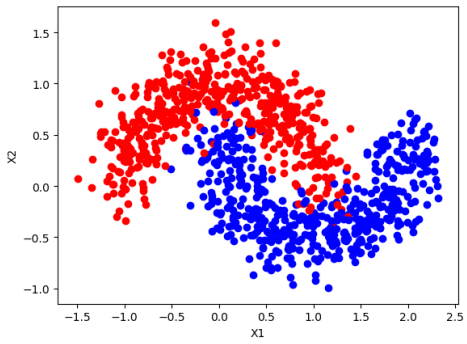
Confusion matrix

	Positive	Negative
True	95	98
False	4	3

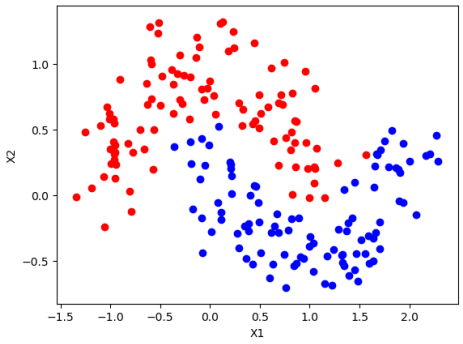
2. Performance

- Accuracy: 96.50%
- Precision: 95.96%
- Recall: 96.94%
- F1score: 96.45%

3. Visualization



Source image

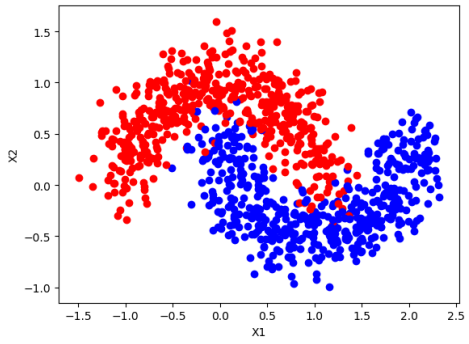


Result

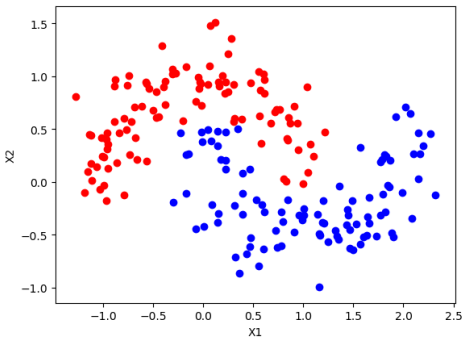
Overall performance

- Accuracy: 95.80%
- Precision: 95.65%
- Recall: 96.04%
- F1score: 95.82%

Epochs: 400



Source image

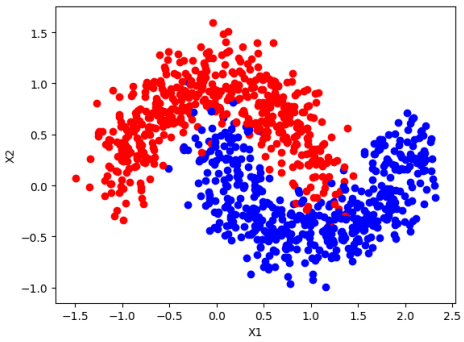


Result

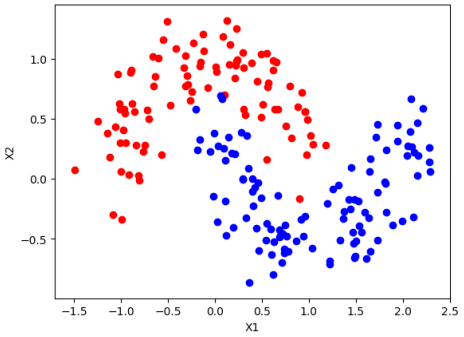
Overall performance:

- Accuracy: 96.60%
- Precision: 95.89%
- Recall: 97.41%
- F1score: 96.63%

Epochs: 500



Source image

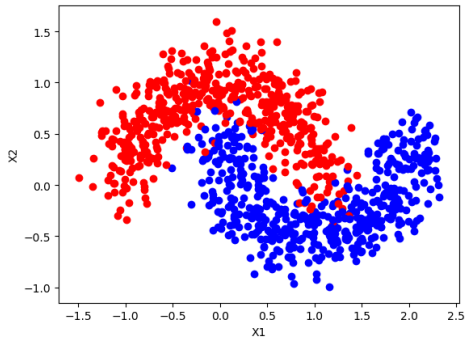


Result

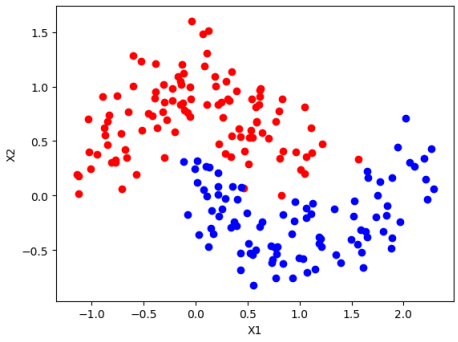
Overall performance:

- Accuracy: 96.10%
- Precision: 95.99%
- Recall: 96.19%
- F1score: 96.09%

Epochs: 100



Source image

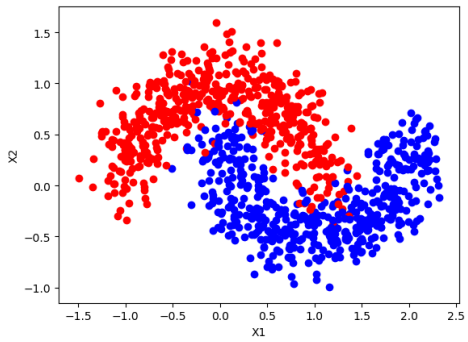


Result

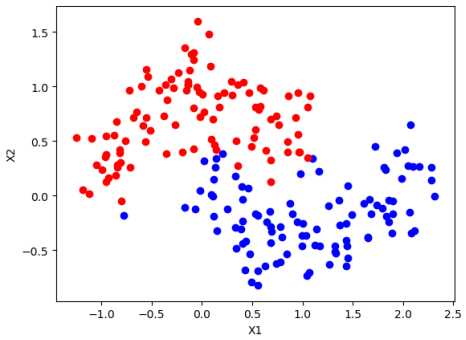
Overall performance:

- Accuracy: 93.40%
- Precision: 94.83%
- Recall: 91.82%
- F1score: 93.29%

Epochs: 50



Source image



Result

Overall performance:

- Accuracy: 92.70%
- Precision: 91.31%
- Recall: 94.60%
- F1score: 92.87%

Dataset2

Different number of neurons

Parameters

- Epochs: 300
- Learning rate: 0.01
- Regularization lambda: 1e-05

Result:

NNodes	10	25	50	100	150
Accuracy	80.98%	87.10%	90.43%	90.21%	90.32%

NNodes: 150

Confusion matrix:

```
[[84.  0.  0.  0.  1.  1.  1.  0.  1.  0.]
 [ 5. 80.  0.  0.  0.  0.  2.  0.  2.  2.]
 [ 2.  0. 84.  0.  0.  0.  0.  0.  0.  0.]
 [ 1.  1.  0. 76.  0.  7.  0.  1.  5.  0.]
 [ 2.  0.  0.  0. 83.  0.  3.  0.  0.  4.]
 [ 0.  0.  1.  0.  0. 84.  0.  0.  1.  5.]
 [ 0.  1.  0.  0.  0.  0. 90.  0.  0.  0.]
 [ 0.  1.  1.  0.  2.  0.  0. 79.  1.  5.]
 [ 0.  3.  2.  4.  0.  2.  2.  0. 75.  0.]
 [ 3.  0.  0.  3.  0.  4.  0.  1.  4. 77.]]
```

- Accuracy: 90.32%
- Precision: 90.53%
- Recall: 90.37%
- F1: 90.45%

NNodes: 100

Confusion matrix:

```
[[84.  0.  0.  0.  1.  2.  0.  1.  0.  0.]
 [ 0. 79.  0.  2.  0.  1.  1.  0.  3.  5.]
 [ 0.  0. 82.  3.  0.  0.  0.  0.  0.  1.]
 [ 0.  1.  1. 78.  0.  3.  0.  3.  5.  0.]
 [ 3.  1.  0.  0. 77.  0.  5.  0.  2.  4.]
 [ 0.  2.  0.  0.  0. 80.  2.  0.  1.  6.]
 [ 0.  1.  0.  0.  0.  0. 90.  0.  0.  0.]
 [ 0.  0.  1.  0.  2.  1.  0. 84.  1.  0.]
 [ 0.  7.  0.  3.  1.  6.  1.  0. 70.  0.]
 [ 0.  0.  0.  2.  0.  2.  0.  1.  0. 87.]]
```

- Accuracy: 90.21%
- Precision: 90.40%
- Recall: 90.23%
- F1: 90.32%

NNodes: 50

Confusion matrix:

```
[[83.  0.  1.  0.  1.  3.  0.  0.  0.  0.]
 [ 0. 82.  1.  2.  1.  0.  1.  0.  0.  4.]
 [ 0.  0. 84.  2.  0.  0.  0.  0.  0.  0.]
 [ 0.  3.  2. 75.  0.  2.  0.  2.  5.  2.]
```

[0. 0. 0. 0. 83. 0. 5. 0. 1. 3.]
[0. 0. 1. 0. 0. 83. 4. 0. 0. 3.]
[0. 2. 0. 0. 0. 0. 89. 0. 0. 0.]
[0. 1. 3. 0. 0. 2. 0. 79. 1. 3.]
[0. 4. 2. 0. 0. 6. 1. 0. 72. 3.]
[0. 1. 0. 2. 0. 3. 0. 1. 2. 83.]]

- Accuracy: 90.43%
- Precision: 90.80%
- Recall: 90.45%
- F1: 90.63%

NNodes: 25

Confusion matrix:

[[76. 0. 0. 2. 1. 3. 6. 0. 0. 0.]
[0. 79. 0. 0. 5. 1. 0. 0. 1. 5.]
[0. 0. 75. 9. 0. 0. 0. 0. 0. 2.]
[0. 2. 1. 74. 0. 2. 1. 0. 10. 1.]
[2. 0. 0. 0. 81. 0. 4. 4. 0. 1.]
[0. 0. 0. 1. 0. 87. 1. 0. 0. 2.]
[0. 6. 0. 0. 0. 0. 85. 0. 0. 0.]
[0. 0. 1. 0. 3. 4. 0. 73. 6. 2.]
[0. 5. 0. 4. 0. 6. 0. 0. 72. 1.]
[0. 0. 0. 4. 0. 5. 0. 0. 2. 81.]]

- Accuracy: 87.10%
- Precision: 87.68%
- Recall: 87.06%
- F1: 87.37%

NNodes: 10

Confusion matrix:

[[85. 0. 0. 1. 1. 1. 0. 0. 0. 0.]
[1. 60. 0. 0. 4. 6. 2. 0. 5. 13.]
[0. 0. 69. 12. 0. 4. 0. 0. 1. 0.]
[0. 1. 9. 69. 0. 5. 0. 5. 2. 0.]
[0. 1. 0. 0. 80. 0. 7. 1. 0. 3.]
[0. 0. 0. 0. 1. 84. 2. 1. 0. 3.]
[0. 0. 0. 0. 0. 0. 90. 0. 0. 1.]
[0. 0. 1. 1. 0. 10. 0. 71. 0. 6.]
[2. 6. 9. 5. 1. 11. 0. 4. 48. 2.]
[9. 2. 0. 4. 0. 4. 0. 1. 0. 72.]]

- Accuracy: 80.98%
- Precision: 81.83%
- Recall: 80.93%
- F1: 81.38%

Different Lambda for regularization

Parameters

- NNodes: 100
- Epochs: 300
- Learning rate: 0.01

Result:

Lambda	1e-06	1e-05	1e-04	1e-03	1e-02
Accuracy	90.43%	90.32%	92.21%	94.10%	90.32%

Regularization lambda: 1e-06

Confusion matrix:

```
[[82.  0.  1.  1.  0.  0.  4.  0.  0.  0.]
 [ 0. 79.  2.  0.  0.  1.  1.  0.  0.  8.]
 [ 0.  0. 86.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  2.  0. 74.  0.  4.  0.  3.  6.  2.]
 [ 0.  1.  0.  0. 78.  0.  6.  0.  0.  7.]
 [ 0.  0.  0.  0.  0. 85.  3.  0.  0.  3.]
 [ 0.  1.  0.  0.  0.  0. 90.  0.  0.  0.]
 [ 0.  1.  1.  2.  0.  1.  0. 83.  1.  0.]
 [ 0.  6.  1.  0.  0.  4.  0.  0. 71.  6.]
 [ 0.  0.  0.  2.  0.  2.  0.  1.  2. 85.]]
```

- Accuracy: 90.43%
- Precision: 91.09%
- Recall: 90.47%
- F1: 90.78%

Regularization lambda: 1e-05

Confusion matrix:

```
[[81.  1.  2.  0.  1.  1.  2.  0.  0.  0.]
 [ 0. 81.  1.  0.  1.  0.  0.  0.  1.  7.]
 [ 0.  0. 83.  2.  0.  0.  0.  0.  0.  1.]
 [ 0.  4.  0. 74.  0.  3.  1.  1.  7.  1.]
 [ 2.  0.  0.  0. 81.  0.  2.  0.  0.  7.]
 [ 0.  2.  1.  0.  0. 84.  3.  0.  0.  1.]
 [ 0.  1.  0.  0.  0.  0. 90.  0.  0.  0.]
 [ 0.  1.  0.  0.  0.  3.  0. 81.  2.  2.]
 [ 0.  8.  0.  0.  0.  5.  0.  1. 73.  1.]
 [ 0.  1.  0.  1.  0.  3.  0.  2.  1. 84.]]
```

- Accuracy: 90.32%
- Precision: 90.82%
- Recall: 90.34%

- F1: 90.58%

Regularization lambda: 0.0001
Confusion matrix:

```
[[86.  0.  0.  0.  1.  0.  1.  0.  0.  0.]
 [ 4. 81.  0.  0.  0.  1.  0.  0.  1.  4.]
 [ 2.  0. 80.  4.  0.  0.  0.  0.  0.  0.]
 [ 0.  4.  1. 77.  0.  5.  0.  0.  2.  2.]
 [ 2.  0.  0.  0. 81.  0.  4.  1.  0.  4.]
 [ 0.  1.  0.  0.  0. 84.  0.  0.  1.  5.]
 [ 0.  0.  1.  0.  0.  0. 90.  0.  0.  0.]
 [ 0.  0.  0.  0.  3.  1.  0. 83.  2.  0.]
 [ 0.  3.  0.  1.  0.  3.  0.  1. 79.  1.]
 [ 1.  0.  0.  0.  0.  1.  0.  0.  2. 88.]]
```

- Accuracy: 92.21%
- Precision: 92.45%
- Recall: 92.23%
- F1: 92.34%

Regularization lambda: 0.0001
Confusion matrix:

```
[[85.  0.  0.  0.  0.  0.  3.  0.  0.  0.]
 [ 1. 82.  0.  1.  0.  1.  1.  0.  1.  4.]
 [ 3.  0. 83.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  1.  0. 74.  0.  7.  0.  6.  2.  1.]
 [ 1.  0.  0.  0. 85.  0.  2.  0.  0.  4.]
 [ 0.  0.  1.  0.  0. 88.  0.  0.  0.  2.]
 [ 0.  1.  0.  0.  0.  0. 90.  0.  0.  0.]
 [ 0.  1.  1.  1.  1.  4.  0. 79.  1.  1.]
 [ 0.  4.  0.  1.  0.  7.  2.  1. 69.  4.]
 [ 1.  0.  0.  1.  0.  3.  0.  0.  1. 86.]]
```

- Accuracy: 91.32%
- Precision: 91.82%
- Recall: 91.32%
- F1: 91.57%

Regularization lambda: 0.001
Confusion matrix:

```
[[86.  0.  0.  0.  1.  0.  1.  0.  0.  0.]
 [ 0. 82.  0.  1.  0.  0.  1.  0.  1.  6.]
 [ 1.  0. 84.  1.  0.  0.  0.  0.  0.  0.]
 [ 0.  5.  0. 76.  0.  3.  0.  2.  5.  0.]
 [ 1.  1.  0.  0. 86.  0.  1.  0.  0.  3.]
 [ 0.  0.  0.  0.  0. 87.  1.  0.  0.  3.]
```

[0. 1. 0. 0. 0. 0. 90. 0. 0. 0.]
[0. 0. 0. 0. 1. 1. 0. 85. 2. 0.]
[0. 4. 0. 0. 0. 2. 0. 0. 82. 0.]
[0. 0. 0. 1. 0. 2. 0. 0. 1. 88.]]

- Accuracy: 94.10%
- Precision: 94.30%
- Recall: 94.14%
- F1: 94.22%

Regularization lambda: 0.01

Confusion matrix:

[[86. 0. 0. 0. 1. 0. 1. 0. 0. 0.]
[0. 76. 0. 0. 2. 1. 0. 0. 0. 12.]
[0. 0. 80. 5. 0. 0. 0. 0. 0. 1.]
[0. 5. 1. 77. 0. 2. 0. 5. 1. 0.]
[0. 2. 0. 0. 87. 0. 1. 1. 1. 0.]
[0. 0. 0. 0. 0. 85. 1. 0. 0. 5.]
[0. 2. 0. 0. 0. 0. 89. 0. 0. 0.]
[0. 0. 1. 0. 1. 0. 0. 87. 0. 0.]
[0. 11. 1. 1. 0. 9. 1. 1. 59. 5.]
[0. 0. 0. 2. 0. 4. 0. 0. 0. 86.]]

- Accuracy: 90.32%
- Precision: 90.98%
- Recall: 90.29%3
- F1: 90.63%

Different learning rate

Parameters

- NNodes: 100
- Epochs: 300
- Regularization lambda: 0.001

Result:

Learning Rate	0.01	0.1	0.5	0.75	1
Accuracy	93.21%	93.33%	91.32%	86.65%	82.20%

Learning rate: 0.01

Confusion matrix:

[[86. 0. 0. 0. 1. 0. 1. 0. 0. 0.]
[0. 80. 0. 1. 0. 0. 1. 0. 1. 8.]
[0. 0. 85. 1. 0. 0. 0. 0. 0. 0.]
[0. 6. 0. 76. 0. 3. 0. 2. 4. 0.]
[0. 1. 0. 0. 86. 0. 2. 0. 0. 3.]

[0. 0. 0. 0. 0. 87. 1. 0. 0. 3.]
[0. 1. 0. 0. 0. 0. 90. 0. 0. 0.]
[0. 0. 0. 0. 1. 0. 0. 84. 1. 3.]
[0. 7. 0. 0. 0. 2. 1. 0. 77. 1.]
[1. 0. 0. 1. 0. 2. 0. 0. 1. 87.]]

- Accuracy: 0.932146829810901
- Precision: 0.9354923250190981
- Recall: 0.9324240486964438
- F1: 0.9339556668521314

Learning rate: 0.1

Confusion matrix:

[[86. 0. 0. 0. 1. 0. 1. 0. 0. 0.]
[0. 73. 0. 2. 1. 0. 0. 0. 5. 10.]
[0. 0. 86. 0. 0. 0. 0. 0. 0. 0.]
[0. 3. 0. 76. 0. 4. 0. 2. 6. 0.]
[1. 0. 0. 0. 86. 0. 1. 0. 3. 1.]
[0. 0. 0. 0. 0. 87. 1. 0. 0. 3.]
[0. 1. 0. 0. 0. 0. 90. 0. 0. 0.]
[0. 0. 0. 0. 2. 0. 0. 85. 2. 0.]
[0. 3. 0. 0. 0. 2. 0. 0. 83. 0.]
[0. 0. 0. 1. 0. 2. 0. 0. 2. 87.]]

- Accuracy: 0.9332591768631813
- Precision: 0.9362297839390326
- Recall: 0.9338363090256104
- F1: 0.9350315147925639

Learning rate: 0.5

Confusion matrix:

[[82. 0. 1. 0. 1. 1. 3. 0. 0. 0.]
[0. 71. 0. 2. 0. 1. 0. 1. 1. 15.]
[0. 0. 86. 0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 78. 0. 4. 0. 3. 6. 0.]
[0. 0. 0. 0. 83. 0. 2. 0. 3. 4.]
[0. 0. 0. 2. 0. 81. 0. 0. 1. 7.]
[0. 2. 0. 0. 0. 0. 87. 0. 2. 0.]
[0. 0. 0. 0. 0. 0. 0. 88. 1. 0.]
[0. 4. 1. 0. 0. 3. 0. 1. 79. 0.]
[0. 0. 0. 1. 0. 2. 0. 0. 3. 86.]]

- Accuracy: 0.9132369299221357
- Precision: 0.9190302248775799
- Recall: 0.9138782504744889
- F1: 0.9164469970416527

Learning rate: 0.75

Confusion matrix:

```
[[77.  0.  3.  0.  0.  2.  3.  0.  3.  0.]
 [ 0. 66.  1.  2.  1.  0.  0.  0.  5. 16.]
 [ 0.  0. 85.  0.  0.  0.  0.  0.  0.  1.]
 [ 0.  0.  0. 73.  0.  1.  0.  6.  7.  4.]
 [ 0.  0.  0.  0. 83.  0.  4.  0.  3.  2.]
 [ 0.  0.  1. 10.  0. 54.  3.  0.  2. 21.]
 [ 0.  2.  0.  0.  0.  0. 88.  0.  1.  0.]
 [ 0.  0.  0.  0.  1.  0.  0. 87.  1.  0.]
 [ 0.  2.  0.  0.  0.  2.  0.  0. 78.  6.]
 [ 0.  0.  0.  0.  0.  2.  0.  0.  2. 88.]]
```

- Accuracy: 0.8665183537263627
- Precision: 0.8843671703002431
- Recall: 0.8673871559360535
- F1: 0.875794868219974

Learning rate: 1.0

Confusion matrix:

```
[[84.  0.  0.  0.  1.  0.  2.  0.  0.  1.]
 [ 0. 71.  0.  1.  2.  1.  0.  0.  0. 16.]
 [ 1.  2. 65.  0.  0.  0.  0.  1.  8.  9.]
 [ 0.  0.  0. 41.  0.  5.  0.  4.  6. 35.]
 [ 0.  2.  0.  0. 83.  0.  0.  2.  0.  5.]
 [ 0.  0.  0.  0.  0. 59.  8.  0.  0. 24.]
 [ 0.  2.  0.  0.  0.  0. 89.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0. 89.  0.  0.]
 [ 0.  5.  0.  0.  0.  8.  0.  1. 73.  1.]
 [ 0.  0.  0.  0.  0.  4.  0.  1.  2. 85.]]
```

- Accuracy: 0.8220244716351501
- Precision: 0.8671961902999932
- Recall: 0.8223134676243878
- F1: 0.8441586622621878

Different epochs

Parameters

- NNodes: 100
- Learning rate: 0.01
- Regularization lambda: 0.001

Result:

Epochs	10	25	50	100	200
Accuracy	82.09%	86.65%	88.77%	92.99%	93.77%

Epochs: 200

Confusion matrix:

```
[[87.  0.  0.  0.  1.  0.  0.  0.  0.  0.]
 [ 0. 82.  0.  1.  0.  0.  0.  0.  2.  6.]
 [ 2.  0. 84.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  2.  0. 75.  0.  4.  0.  2.  7.  1.]
 [ 1.  1.  0.  0. 86.  0.  1.  0.  0.  3.]
 [ 0.  0.  0.  0.  0. 86.  1.  0.  0.  4.]
 [ 0.  1.  0.  0.  0.  0. 90.  0.  0.  0.]
 [ 0.  0.  0.  0.  1.  0.  0. 86.  2.  0.]
 [ 0.  3.  0.  0.  0.  4.  0.  0. 81.  0.]
 [ 1.  0.  0.  1.  0.  2.  0.  0.  2. 86.]]
```

- Accuracy: 93.77%
- Precision: 94.02%
- Recall: 93.81f%
- F1: 93.91%

Epochs: 100

Confusion matrix:

```
[[86.  0.  0.  0.  0.  0.  1.  0.  1.  0.]
 [ 2. 80.  0.  1.  0.  0.  1.  0.  3.  4.]
 [ 0.  0. 83.  3.  0.  0.  0.  0.  0.  0.]
 [ 0.  4.  2. 78.  0.  2.  0.  2.  3.  0.]
 [ 0.  0.  0.  0. 84.  0.  3.  0.  0.  5.]
 [ 0.  1.  0.  0.  0. 85.  2.  0.  0.  3.]
 [ 0.  2.  0.  0.  0.  0. 89.  0.  0.  0.]
 [ 0.  1.  0.  0.  0.  1.  0. 85.  1.  1.]
 [ 0.  3.  0.  1.  0.  2.  1.  0. 80.  1.]
 [ 0.  0.  0.  1.  0.  2.  0.  2.  1. 86.]]
```

- Accuracy: 92.99%
- Precision: 93.17%
- Recall: 93.03%
- F1: 93.10%

Epochs: 50

Confusion matrix:

```
[[85.  0.  0.  0.  1.  0.  1.  1.  0.  0.]
 [ 3. 81.  1.  3.  0.  1.  0.  0.  2.  0.]
 [ 4.  0. 78.  1.  0.  0.  0.  1.  0.  2.]
 [ 0.  0.  2. 75.  0.  7.  0.  0.  5.  2.]
```

[1. 0. 0. 0. 80. 0. 7. 0. 0. 4.]
[0. 0. 1. 1. 0. 80. 3. 0. 0. 6.]
[0. 2. 0. 0. 0. 0. 89. 0. 0. 0.]
[0. 2. 1. 1. 1. 2. 0. 74. 6. 2.]
[0. 7. 0. 0. 0. 6. 0. 0. 75. 0.]
[1. 0. 0. 2. 0. 5. 0. 1. 2. 81.]]

- Accuracy: 88.77%
- Precision: 89.15%
- Recall: 88.78%
- F1: 88.97%

Epochs: 25

Confusion matrix:

[[86. 0. 0. 0. 1. 1. 0. 0. 0. 0.]
[1. 72. 1. 2. 0. 0. 0. 0. 6. 9.]
[4. 0. 82. 0. 0. 0. 0. 0. 0. 0.]
[0. 9. 1. 75. 0. 1. 0. 2. 1. 2.]
[1. 3. 0. 0. 80. 0. 7. 0. 0. 1.]
[0. 0. 0. 1. 0. 83. 3. 0. 1. 3.]
[0. 3. 0. 0. 0. 0. 88. 0. 0. 0.]
[0. 3. 1. 0. 2. 9. 0. 68. 1. 5.]
[0. 10. 0. 1. 1. 7. 2. 1. 62. 4.]
[1. 0. 0. 1. 0. 4. 0. 1. 2. 83.]]

- Accuracy: 86.65%
- Precision: 87.39%
- Recall: 86.66%
- F1: 87.02%

Epochs: 10

Confusion matrix:

[[82. 0. 1. 0. 1. 3. 0. 0. 0. 1.]
[0. 78. 0. 6. 1. 1. 0. 0. 3. 2.]
[3. 0. 71. 5. 0. 0. 0. 4. 0. 3.]
[0. 4. 4. 66. 0. 5. 1. 1. 7. 3.]
[0. 4. 0. 0. 79. 1. 4. 0. 0. 4.]
[0. 1. 1. 0. 0. 77. 4. 0. 1. 7.]
[0. 3. 2. 0. 1. 0. 84. 0. 1. 0.]
[0. 0. 6. 1. 1. 6. 0. 65. 7. 3.]
[0. 9. 2. 3. 0. 6. 0. 1. 64. 3.]
[2. 1. 0. 5. 0. 8. 0. 1. 3. 72.]]

- Accuracy: 82.09%
- Precision: 82.62%
- Recall: 82.08%
- F1: 82.35%

Discussion

Discuss your method and results:

The neural network classifier works successfully and the accuracy of the model achieves at least 90%.

When initializing the weight matrices, data distributions matter. In my experience on this project, I find that normal distribution outperforms uniform distribution. Additionally, the variance of normal distribution also matters. The bigger the variance, the difference larger between the initial weights, which may have influence on the model that you are going to train.

For the dataset2, there is an interesting phenomenon. When the lambda for regularization becomes larger, the accuracy of the model gets higher. The reason may be that there are many dirty data sample (or noise) to distract the learning.

What are the strengths and weaknesses of your method?

The neural network classifier works successfully and the accuracy of the model achieves at least 90%.

Potential future work:

Current network only contains one hidden layer. A neural network with multiple hidden layers will be developed to solve more complicated classification problem. The extensibility and reusability of the code can be improved.

Conclusions

The neural network with only one hidden layer can successfully complete the tasking of classifying data. For the input of the model, it can be a one-dimensional input or a multi-dimensional input. This model can not only solve the binary classification problem, but also classify the data of multiple categories.