**Extensions, Improvements, Additional Work**

**Rajdeep Pinge**

**201401103**

1. **RBF classification using pseudo inverse**

**Modification: Centres are initialized using k-Means clustering.**

**Description:** For RBF classification, we need to set up the centres and the spread of the Gaussian function. The spread is dependent on the centres. So performance depends on how we initialize the centres.

There are 2 ways of initializing the centres, one is to randomly select the input data samples and take them as centres. Other way is to use the k-means clustering to adjust the centres.

**Observations:**

For Set 5

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Training | | | Testing | | |
| Dataset | Number of hidden neurons | Overall Accuracy (ηo) | Average Accuracy (ηa) | Geometric Mean Accuracy (ηg) | Overall Accuracy (ηo) | Average Accuracy (ηa) | Geometric Mean Accuracy (ηg) |
| AE | 12 | 96.7741 | 96.875 | 96.8245 | 99.2700 | 99.5689 | 99.5661 |
| Iris | 12 | 97.7777 | 97.7777 | 97.7264 | 98.0952 | 98.0952 | 98.0860 |
| ION | 40 | 87.5 | 87.5 | 87.2239 | 69.7211 | 63.9026 | 60.5016 |
| Liver | 20 | 66.2650 | 66.2650 | 65.9906 | 63.4482 | 63.3793 | 63.3775 |
| PIMA | 12 | 73.5483 | 73.5483 | 73.3188 | 71.4673 | 73.7445 | 73.5080 |
| VC | 153 | 85.0891 | 85.0342 | 83.9725 | 70.6161 | 70.5325 | 66.3911 |
| Wine | 20 | 98.3333 | 98.3333 | 98.3047 | 92.3728 | 92.8427 | 92.7667 |

The reference to number of neurons is taken from the original algorithm of RBF pseudo inverse.

**Confusion Matrices:**

* **Ae:**

Over training data:

14.0 0.0 0.0 0.0

0.0 16.0 0.0 0.0

0.0 1.0 15.0 0.0

0.0 0.0 1.0 15.0

Over testing data:

26.0 0.0 0.0 0.0

0.0 57.0 0.0 1.0

0.0 0.0 30.0 0.0

0.0 0.0 0.0 23.0

* **Iris:**

Over training data:

15.0 0.0 0.0

0.0 14.0 1.0

0.0 0.0 15.0

Over testing data:

35.0 0.0 0.0

0.0 34.0 1.0

0.0 1.0 34.0

* **ION:**

Over training data:

34.0 2.0

7.0 29.0

Over testing data:

136.0 25.0

51.0 39.0

* **Liver:**

Over training data:

50.0 33.0

23.0 60.0

Over testing data:

39.0 23.0

30.0 53.0

* **PIMA:**

Over training data:

105.0 50.0

32.0 123.0

Over testing data:

173.0 82.0

23.0 90.0

* **VC:**

Over training data:

149.0 0.0 0.0 1.0

3.0 115.0 32.0 10.0

7.0 30.0 105.0 5.0

2.0 0.0 2.0 156.0

Over testing data:

107.0 4.0 2.0 1.0

6.0 67.0 39.0 9.0

4.0 43.0 38.0 9.0

1.0 2.0 4.0 86.0

* **Wine:**

Over training data:

20.0 0.0 0.0

1.0 19.0 0.0

0.0 0.0 20.0

Over testing data:

38.0 1.0 0.0

5.0 45.0 1.0

0.0 2.0 26.0

**Inference/Conclusion:** The results are very close to the random initialization algorithm. The results change in the range of ~5-10% because the k-means algorithms is dependent on the initial centroid positions for clustering.

In some cases, the results are very poor. This may be attributed to the fact that k-means clustering always gives a local optimum and is vulnerable to the presence of outliers.

1. **MLP classification using Fourth Power Error**

**Description:** Instead of squared error, here we use the fourth power error. Fourth power error highly penalizes the algorithm for the wrong prediction. But at the same time care must be taken that the initial weights and not close to zero. The learning rate, hidden neurons and the epoch must also be chosen carefully, because if the error goes beyond certain bound, the sum of error blows up and the algorithm cannot converge

**Observations:**

For Set 5

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | Training | | | Testing | | |
| Dataset | Number of hidden neurons | Learning rate | Epoch | Overall Accuracy (ηo) | Average Accuracy (ηa) | Geometric Mean Accuracy (ηg) | Overall Accuracy (ηo) | Average Accuracy (ηa) | Geometric Mean Accuracy (ηg) |
| AE | 12 | 1e-3 | 1000 | 96.7741935483871 | 96.875 | 96.82458365518542 | 99.27007299270073 | 99.03846153846155 | 99.02427357425654 |
| Iris | 12 | 1e-3 | 8000 | 91.11111111111111 | 91.11111111111111 | 90.90087467832687 | 92.38095238095238 | 92.38095238095238 | 91.93910814472488 |
| ION | 40 | 1e-3 | 1000 | 91.66666666666667 | 91.66666666666667 | 91.49816162222558 | 86.05577689243027 | 83.00552104899931 | 82.30210805944338 |
| Liver | 20 | 1e-4 | 25000 | 66.26506024096386 | 66.26506024096386 | 66.26506024096386 | 63.44827586206897 | 63.78740769529732 | 66.26506024096386 |
| PIMA | 12 | 1e-4 | 10000 | 74.51612903225806 | 74.51612903225806 | 74.49867132463717 | 75.81521739130434 | 73.67863959743188 | 73.47028683096399 |
| VC | 30 | 1e-4 | 10000 | 77.63371150729336 | 77.75680272108842 | 74.41447512637694 | 73.22274881516587 | 74.42886202101737 | 69.47165889948228 |
| Wine | 20 | 1e-3 | 5000 | 100.0 | 100.0 | 100.0 | 97.45762711864407 | 97.30122818358113 | 97.29895670692979 |

Learning rate is tuned according to the output. If the output blows up, the learning rate is reduced. Number of hidden neurons are taken from the original algorithm.

**Confusion Matrices:**

* **Ae:**

Over training data:

14.0 0.0 0.0 0.0

0.0 16.0 0.0 0.0

0.0 1.0 15.0 0.0

0.0 0.0 1.0 15.0

Over testing data:

25.0 1.0 0.0 0.0

0.0 58.0 0.0 0.0

0.0 0.0 30.0 0.0

0.0 0.0 0.0 23.0

* **Iris:**

Over training data:

15.0 0.0 0.0

0.0 13.0 2.0

0.0 2.0 13.0

Over testing data:

35.0 0.0 0.0

0.0 28.0 7.0

0.0 1.0 34.0

* **ION:**

Over training data:

35.0 1.0

5.0 31.0

Over testing data:

151.0 10.0

25.0 65.0

* **Liver:**

Over training data:

55.0 28.0

28.0 55.0

Over testing data:

41.0 21.0

32.0 51.0

* **PIMA:**

Over training data:

118.0 37.0

42.0 113.0

Over testing data:

202.0 53.0

36.0 77.0

* **VC:**

Over training data:

147.0 0.0 1.0 2.0

9.0 75.0 72.0 4.0

6.0 38.0 100.0 3.0

1.0 0.0 2.0 157.0

Over testing data:

112.0 1.0 0.0 1.0

4.0 47.0 66.0 4.0

5.0 29.0 58.0 2.0

1.0 0.0 0.0 92.0

* **Wine:**

Over training data:

20.0 0.0 0.0

0.0 20.0 0.0

0.0 0.0 20.0

Over testing data:

38.0 1.0 0.0

0.0 50.0 1.0

0.0 1.0 27.0

**Inference/Conclusion:** There is dependency between the number of neurons and the learning rate. This may be because, every neuron contributes a forth power error and is the sum of all the errors is significant, the output blows up. Therefore for large number of neurons, we have to reduce the learning rate and increase the epoch.

The algorithm behaves poorly for certain small data sets like Iris where accuracy over testing data has dropped from ~96% to 91%. This may be because of the adverse effect of high penalty over the data where algorithm is not able to learn properly because of the small size of data.

1. **MLP classification using Least Square Error**

**Modification: Using momentum to control weight updates.**

**Description:** If the data set is large, the algorithm takes huge amount of time and iterations to converge. To reduce the amount of time and epoch, the concept of momentum has been introduced.

The momentum term is used for updating the DeltaW values i.e. to modify the weights of the model. It has been found out that taking into account the previous DeltaW value while updating the current value helps the algorithm converge faster. This is applied using the following update formulae in the code for the two weights:

DWo = DWo + lam \* (er \* Yh') + alpha \* DWoOld; % including the momentum term % update rule for output weight

DWi = DWi + lam \* ((Wo'\*er).\*Yh.\*(1-Yh))\*xx' + alpha \* DWiOld; %update for input weight

The term alpha \* DWoOld is called the momentum term. This term is used to control the weight update process. Alpha is the momentum constant which determines how much the previous DeltaW influences the current update.

**Observations:**

For Set 5

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | Training | | | Testing | | |
| Dataset | Number of hidden neurons | Learning rate | Momentum Factor | Epoch | Overall Accuracy (ηo) | Average Accuracy (ηa) | Geometric Mean Accuracy (ηg) | Overall Accuracy (ηo) | Average Accuracy (ηa) | Geometric Mean Accuracy (ηg) |
| AE | 12 | 1e-3 | 0.01 | 500 | 96.7741935483871 | 96.875 | 96.82458365518542 | 99.27007299270073 | 99.03846153846155 | 99.02427357425654 |
| Iris | 12 | 1e-3 | 0.02 | 7000 | 97.77777777777777 | 97.77777777777779 | 97.7264805918825 | 98.0952380952381 | 98.0952380952381 | 98.0577593308067 |
| ION | 40 | 1e-3 | 0.01 | 500 | 91.66666666666667 | 91.66666666666667 | 91.28709291752769 | 84.86055776892431 | 79.86887508626639 | 77.89501171603702 |
| Liver | 20 | 1e-4 | 0.005 | 15000 | 65.66265060240964 | 65.66265060240963 | 65.63777576807824 | 63.44827586206897 | 63.78740769529732 | 63.744412891586975 |
| PIMA | 12 | 1e-4 | 0.002 | 5000 | 73.87096774193549 | 73.87096774193549 | 73.86462854896367 | 75.54347826086956 | 73.728960610793 | 73.57884855521381 |
| VC | 30 | 1e-4 | 0.001 | 5000 | 78.93030794165315 | 79.08630952380953 | 75.95357291570136 | 75.59241706161137 | 76.56701027542339 | 72.72688509120057 |
| Wine | 20 | 1e-3 | 0.01 | 2000 | 100.0 | 100.0 | 100.0 | 95.76271186440678 | 95.45715722186311 | 95.43772319221303 |

The number of neurons and the learning rate were taken from the values found out in original algorithm. The alpha value and the number of epochs have been found out experimentally.

**Confusion Matrices:**

* **Ae:**

Over training data:

14.0 0.0 0.0 0.0

0.0 16.0 0.0 0.0

0.0 1.0 15.0 0.0

0.0 0.0 1.0 15.0

Over testing data:

25.0 1.0 0.0 0.0

0.0 58.0 0.0 0.0

0.0 0.0 30.0 0.0

0.0 0.0 0.0 23.0

* **Iris:**

Over training data:

35.0 0.0 0.0

0.0 33.0 2.0

0.0 0.0 35.0

Over testing data:

15.0 0.0 0.0

0.0 14.0 1.0

0.0 0.0 15.0

* **ION:**

Over training data:

36.0 0.0

6.0 30.0

Over testing data:

157.0 4.0

34.0 56.0

* **Liver:**

Over training data:

56.0 27.0

30.0 53.0

Over testing data:

41.0 21.0

32.0 51.0

* **PIMA:**

Over training data:

116.0 39.0

42.0 113.0

Over testing data:

200.0 55.0

35.0 78.0

* **VC:**

Over training data:

148.0 0.0 0.0 2.0

10.0 77.0 64.0 9.0

5.0 33.0 105.0 4.0

2.0 0.0 1.0 157.0

Over testing data:

113.0 0.0 0.0 1.0

5.0 55.0 54.0 7.0

5.0 28.0 59.0 2.0

1.0 0.0 0.0 92.0

* **Wine:**

Over training data:

20.0 0.0 0.0

0.0 20.0 0.0

0.0 0.0 20.0

Over testing data:

38.0 1.0 0.0

2.0 49.0 0.0

0.0 2.0 26.0

**Inference/Conclusion:** The value of alpha, i.e. the momentum constant must be proportional to the learning rate, if it is too high, the weights will be driven by the previous value and may not converge.

Another interesting observation is the number of epochs required to train the algorithms. As compared to the simple algorithm without momentum term, the epoch required is 50%. Hence the time consumed is less but the accuracy is almost the same. This shows the importance of the momentum update.