Artificial Neural Networks and Deep Architectures-DD2437

Radial basis functions, competitive learning and self-organization

Rithika Harish Kumar, Suhas Sheshadri

Aim

The aim is to learn how to build the structure and perform training of an RBF network and analyze different methods for initializing the structure and learning the weights in an RBF network. Know the concept of vector quantization and learn how to use it in NN context. To recognize and implement different components in the SOM algorithm. Discuss the role of the neighborhood and analyze its effect on SOMs. Learn how SOM-networks can be used to fold high-dimensional spaces and cluster data.

Tools used

Jupyter Notebook is used to carry out the simulations and learning.

Results

2.1.1

What is the lower bound for the number of training examples, N?

At least n, else the system will be incompatible.

What happens with the error if N = n? Why?

Since the given data and the weights match, it would be zero (system is full rank).

Under what conditions, if any, does (4) have a solution in this case?

The equations are independent when N=n. Every training pattern has a different neuron.

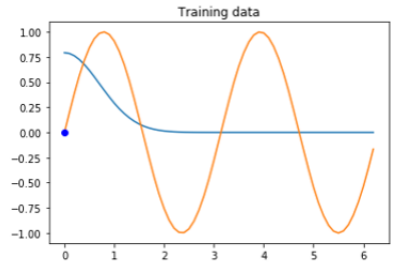
During training we use an error measure defined over the training examples. Is it good to use this measure when evaluating the performance of the network? Explain!

No, training error does not encounter generalization. Test error should be used for evaluation because trained data will perform better than unseen data.

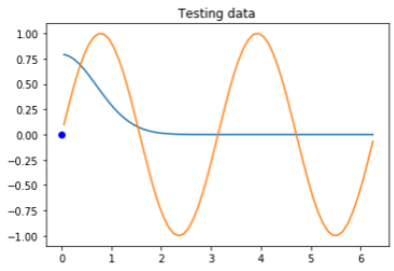
Part I: 3.1

Try to vary the number of units to get the absolute residual error below 0.1, 0.01 and 0.001 in the residual value. Please discuss the results, how many units are needed for the aforementioned error thresholds?

RBF nodes evenly distributed, for **sin 2x Nodes**: 1,6,10,15,30



Residual error: 0.583956639739

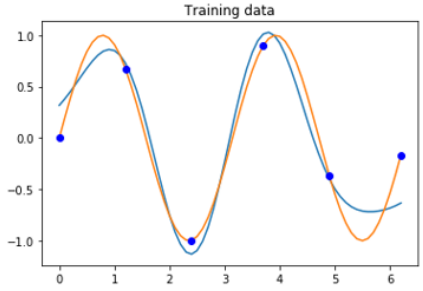


Residual error: 0.577148105464

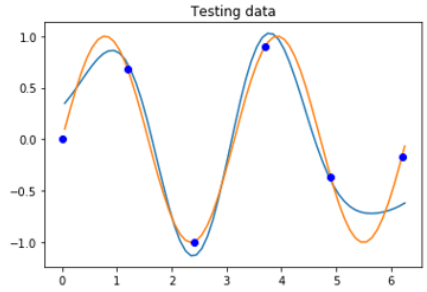
Artificial Neural Networks and Deep Architectures-DD2437

Radial basis functions, competitive learning and self-organization

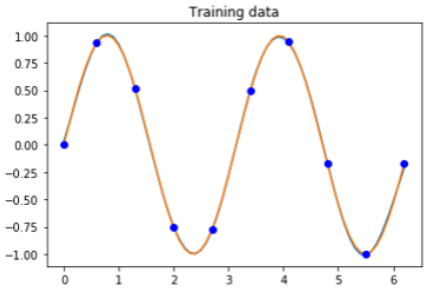
Rithika Harish Kumar, Suhas Sheshadri



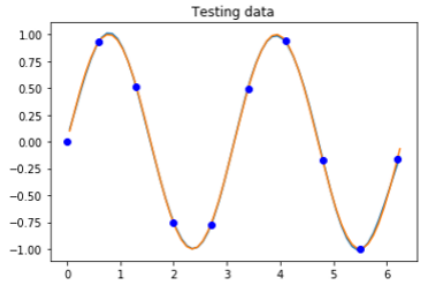
Residual error: 0.136974736335



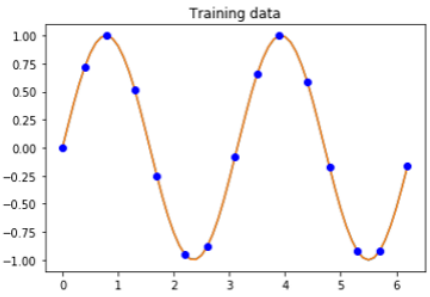
Residual error: 0.139566055483



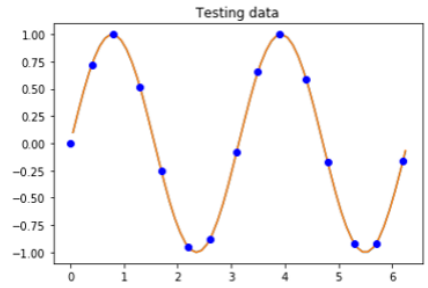
Residual error: 0.0109433642875



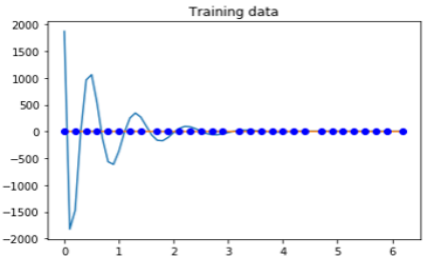
Residual error: 0.0114592418721



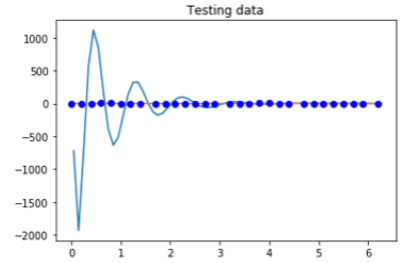
Residual error: 0.00122941489733



Residual error: 0.00125449347139



Residual error: 184.440072526



Residual error: 159.780825317

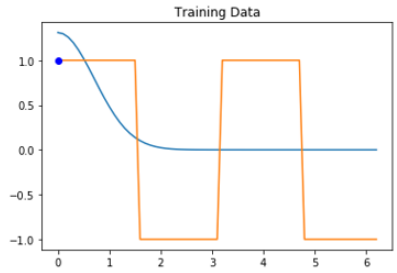
So, at least 6 units are required for 0.1, 10 units for 0.01 and 15 units for 0.001 residual error.

Artificial Neural Networks and Deep Architectures-DD2437

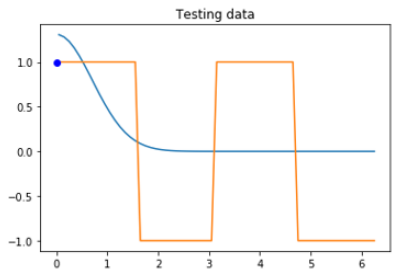
Radial basis functions, competitive learning and self-organization

Rithika Harish Kumar, Suhas Sheshadri

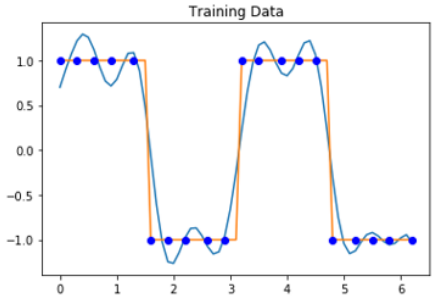
Square(2x) nodes: 1 and 20



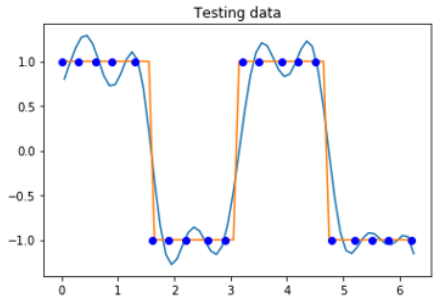
Residual error: 0.853794931649



Residual error: 0.857625849543



Residual error: 0.194119395393

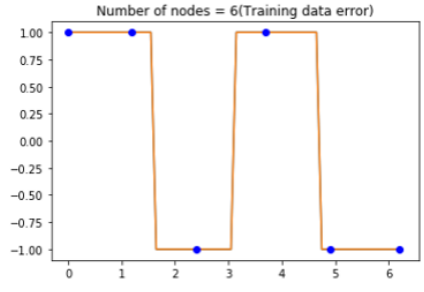


Residual error: 0.195964977088

20 units are required for 0.1 residual error.

The residual error doesn't go less than 0.1. But for some number of hidden nodes, it gives zero training error.

How can you simply transform the output of your RBF network to reduce the residual error to 0 for the square(2x) problem? Still, how many units do you need? In what type of applications could this transform be particularly useful?



Residual error: 0.0

Nodes 6, 9, 16, 17 and 19 are the list of hidden units which give 0 testing data error.

Rithika Harish Kumar, Suhas Sheshadri

We can add one more unit in the hidden layer which works as a residual compensator. It subtracts the residual component from the output and makes training error zero. When test data is input into network, the last unit will lookup the table of residual to find corresponding value.

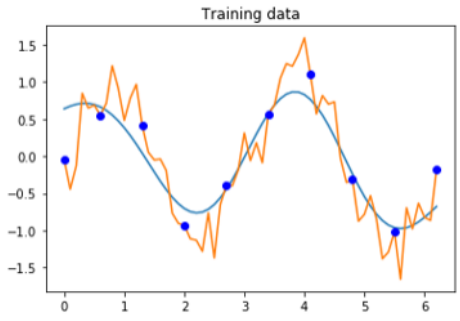
3.2

Compare the two learning approaches in terms of the number of epochs and the number of nodes needed to obtain the requested performance levels. How does it compare, particularly with respect to the number of RBF nodes, to the batch mode training of sin(2x) without noise, as in the previous assignment task? Compare the rate of convergence for different learning rates

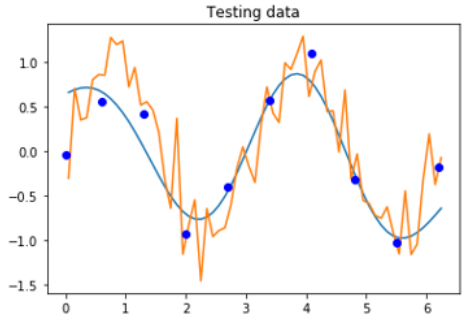
Less epochs give more residual error compared to high epochs

Sin 2x with noise: For 100 epochs and number of nodes = 10

Delta rule

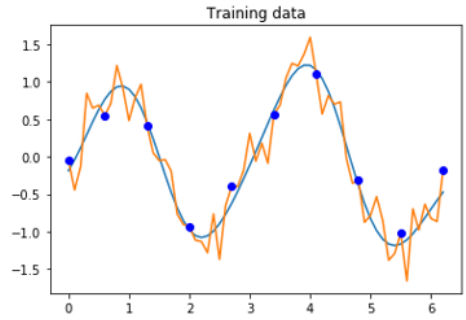


Residual error: 0.299827085756

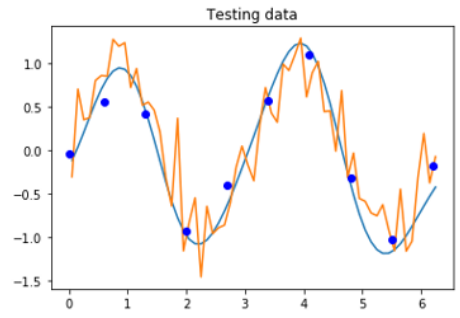


Residual error: 0.323771545426

Online learning



Residual error: 0.196654021144



Residual error: 0.270538650437

How important is the positioning of the RBF nodes in the input space? What strategy did you choose? Is it better than random positioning of the RBF nodes?

Positioning the nodes by distributing it and evenly spacing yields less error than clustering.

Artificial Neural Networks and Deep Architectures-DD2437

Radial basis functions, competitive learning and self-organization

Rithika Harish Kumar, Suhas Sheshadri

How does the rate of convergence change with different values of eta?

If eta is too high, noise on the input data might lead to an increased error in the weight. However, if eta is too small, an optimum (least squares) solution might never be achieved.

Please compare your optimal RBF network trained in batch mode with a two-layer perceptron trained with backprop (also in batch mode).

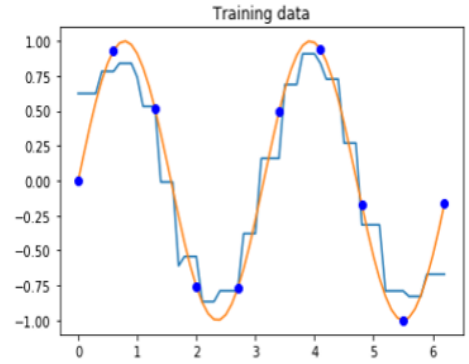
Two-layer perceptron has slightly lower error in comparison to RBF network. Although, RBF's will produce a robust system compared to MLP.

3.3

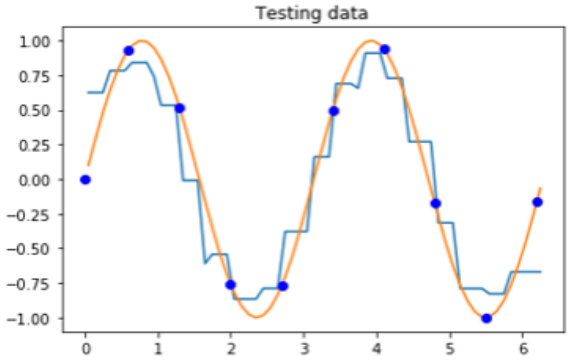
Compare the CL-based approach with your earlier RBF network where you manually positioned RBF nodes in the input space. Use the same number of units (it could be the number of units that allowed you to lower the absolute residual error below 0.01). Make this comparison for both noise- free and noisy approximation of sin(2x). Pay attention to convergence, generalization performance and the resulting position of nodes.

Delta

Sin2x without noise no of nodes= 10 no of epochs =100 positioning using CL

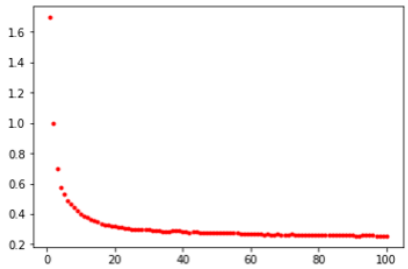


Residual error: 0.156307838459



Residual error: 0.1666418282

Graph with no of epochs and error rate. Error reduces as epoch increases. Error also reduces as number of nodes increases.



Artificial Neural Networks and Deep Architectures-DD2437

Radial basis functions, competitive learning and self-organization

Rithika Harish Kumar, Suhas Sheshadri

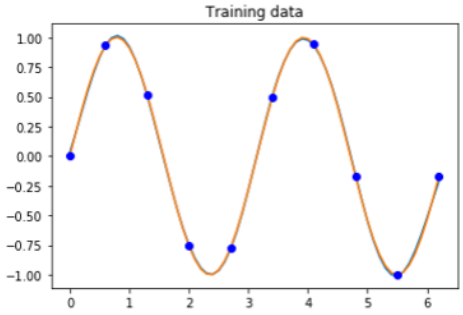
Sin2x with noise

Residual error: 0.254847512059

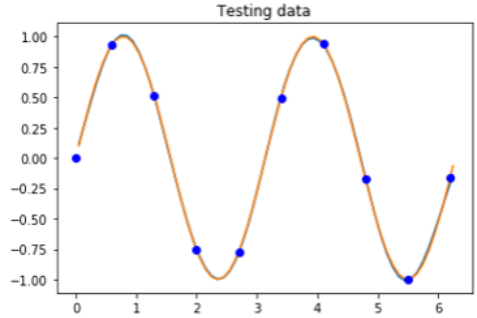
Residual error: 0.299904270296

Least squares:

Sin2x without noise no of nodes= 10, no of epochs =100, positioning using CL

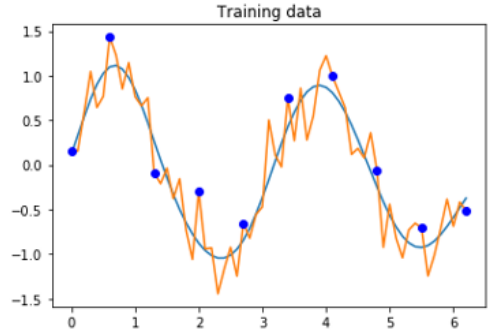


Residual error: 0.0109433642875



Residual error: 0.0114592418721

Sin2x with noise



Residual error: 0.209189552574

Residual error: 0.253556668598

Artificial Neural Networks and Deep Architectures-DD2437

Radial basis functions, competitive learning and self-organization

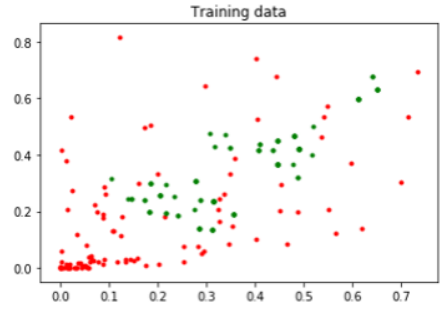
Rithika Harish Kumar, Suhas Sheshadri

Introduce a strategy to avoid dead units, e.g. by having more than a single winner. Choose an example to demonstrate this effect in comparison with the vanilla version of our simple CL algorithm.

Initialize with random input vectors.
Another way is to provide updates to even losing nodes, albeit with very small weights.
Introduce noise to data.
Balanced updates, allowing losers to win.

Configure an RBF network with the use of CL for positioning the RBF units to approximate a two-dimensional function, i.e. from R2 to R2. First thing to do is to load the data and then train the RBF network to find a mapping between the input and output values. Please be careful with the selection of a suitable number of nodes and their initialization to avoid dead-unit and overfitting problems. Report your results and observations, ideally with the support of illustrations, and document your analyses (e.g., inspect the position of units in the input space).

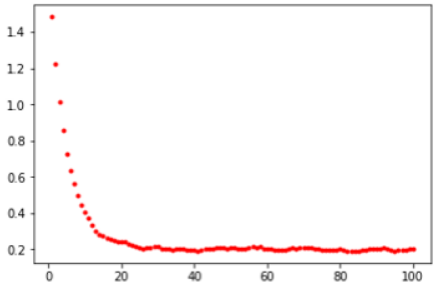
Mapped input and output values, Number of epochs=100 and no of nodes=10

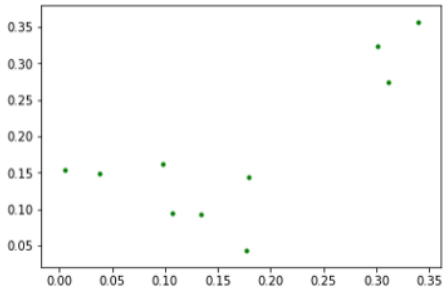


Residual error: 0.203503295676

Residual error: 0.207161305294

Number of epochs vs error and the weights



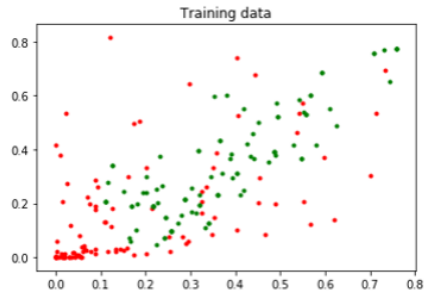


No of nodes= 20

Artificial Neural Networks and Deep Architectures-DD2437

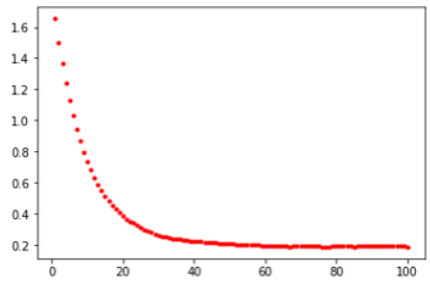
Radial basis functions, competitive learning and self-organization

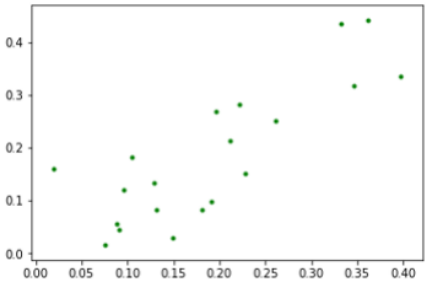
Rithika Harish Kumar, Suhas Sheshadri



Residual error: 0.190400388541

Residual error: 0.211254305289





Part II: 4.1

Finally, you have to print out the result, i.e. the animals in a natural order. Do this by looping through all animals once more, again calculating the index of the winning output node. Save these indices in a 32-element vector pos. By sorting this vector, we will get the animals in the desired order. Check the resulting order. Does it make sense? If everything works, animals next to each other in the listing should always have some similarity between them. Insects should typically be grouped together, separate from the different cats, for example.

No of epochs = 20 with neighbor size = 25 and no of epoch = 100

Rithika Harish Kumar, Suhas Sheshadri

```
[[1 "'penguin'"]
 [4 "'ostrich'"]
 [11 "'frog'"]
 [15 "'crocodile'"]
 [18 "'seaturtle'"]
 [23 "'horse'"]
 [27 "'giraffe'"]
 [29 "'camel'"]
 [31 "'pig'"]
 [34 "'antelop'"]
 [38 "'kangaroo'"]
 [40 "'rabbit'"]
 [43 "'elephant'"]
 [45 "'bat'"]
 [47 "'rat'"]
 [52 "'lion'"]
 [54 "'cat'"]
 [57 "'ape'"]
 [60 "'bear'"]
 [62 "'walrus'"]
 [64 "'dog'"]
 [66 "'hyena'"]
 [69 "'skunk'"]
 [73 "'spider'"]
 [78 "'moskito'"]
 [82 "'housefly'"]
 [84 "'butterfly'"]
 [86 "'dragonfly'"]
 [88 "'grasshopper'"]
 [90 "'beetle'"]
 [95 "'pelican'"]
 [97 "'duck'"]]
```

```
[[0 "'antelop'"]
 [0 "'giraffe'"]
 [0 "'elephant'"]
 [0 "'pig'"]
 [0 "'camel'"]
 [0 "'rabbit'"]
 [0 "'horse'"]
 [1 "'bat'"]
 [1 "'kangaroo'"]
 [12 "'skunk'"]
 [12 "'rat'"]
 [13 "'hyena'"]
 [14 "'cat'"]
 [14 "'dog'"]
 [14 "'bear'"]
 [14 "'lion'"]
 [14 "'ape'"]
 [24 "'walrus'"]
 [27 "'frog'"]
 [27 "'crocodile'"]
 [27 "'seaturtle'"]
 [40 "'ostrich'"]
 [42 "'duck'"]
 [42 "'pelican'"]
 [42 "'penguin'"]
 [55 "'dragonfly'"]
 [55 "'beetle'"]
 [68 "'grasshopper'"]
 [70 "'butterfly'"]
 [82 "'housefly'"]
 [83 "'moskito'"]
 [96 "'spider'"]]
```

The insects have relatively high points and are clustered together, meaning that those species are similar in various characteristics. Cat and dog are clustered together. Large animals have relatively low points. The order can be inversed, since it is a topology preservation algorithm.

4.2 Cyclic tour

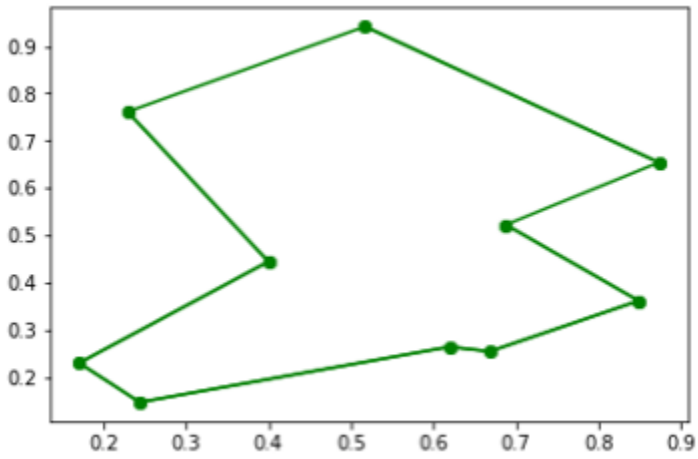
Please plot both the tour and the training points. Give your interpretation.

No of epochs = 20, neighbor size = 2. After 15 epochs, decrease the neighbor size to 1 and after 10 more epochs, neighbor size is set to 0. The result is shown below.

Artificial Neural Networks and Deep Architectures-DD2437

Radial basis functions, competitive learning and self-organization

Rithika Harish Kumar, Suhas Sheshadri



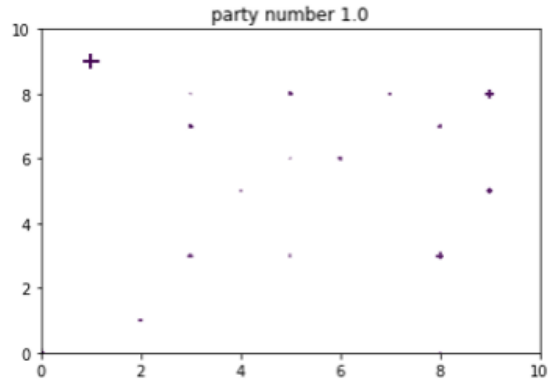
4.3 Data Clustering: Votes of MPs

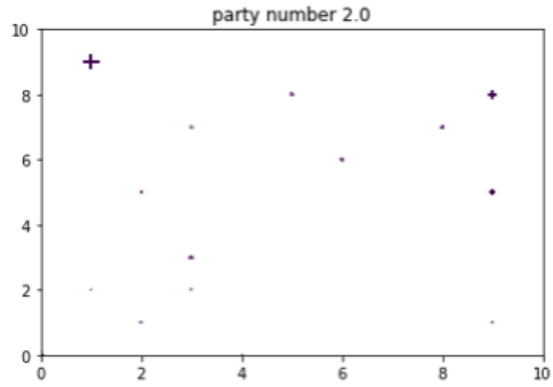
Please display the results with respect to different attributes (i.e. party, gender, district) and describe the results, provide your interpretation.

No of epochs = 100 (results were identical when we tried with 1000 epochs).

The clustering of parties is shown below.

Different parties (with different political views, hence different kinds of votes) are clustered significantly.

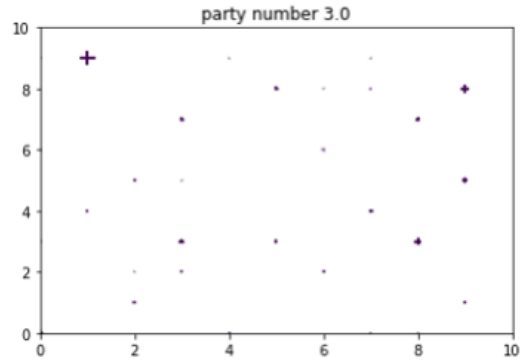


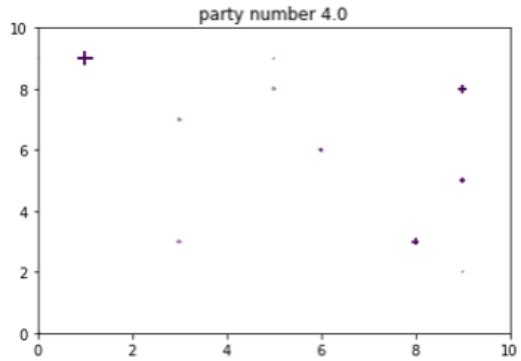


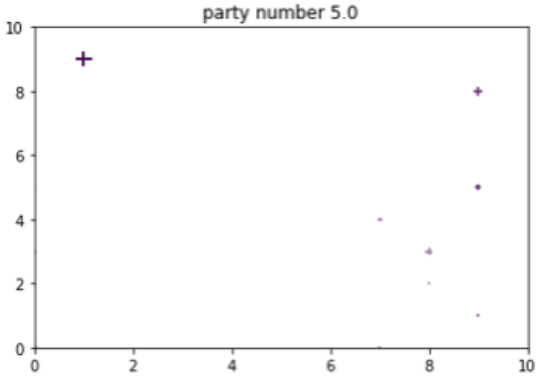
Artificial Neural Networks and Deep Architectures-DD2437

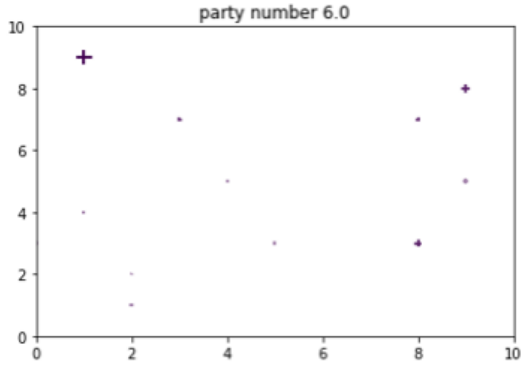
Radial basis functions, competitive learning and self-organization

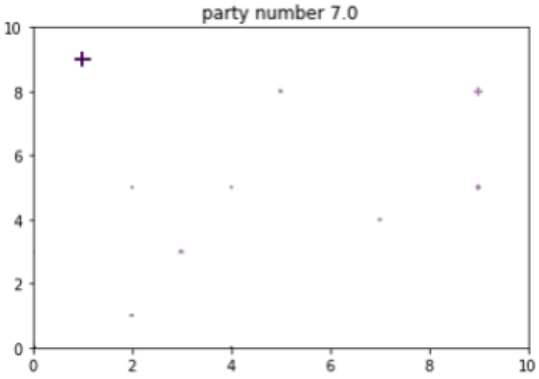
Rithika Harish Kumar, Suhas Sheshadri





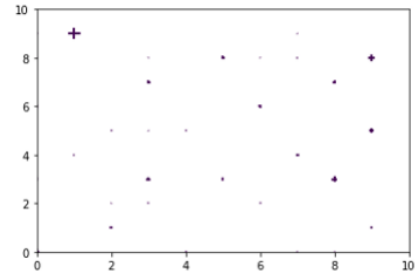






W.r.t. Gender

Male MPs are more than female MPs but their political view is not significantly based on their sex.



Artificial Neural Networks and Deep Architectures-DD2437

Radial basis functions, competitive learning and self-organization

Rithika Harish Kumar, Suhas Sheshadri

W.r.t. District

Voting pattern cannot be predicted based on the classification. Nothing much is visible.

2.0	39
1.0	29
16.0	18
5.0	17
13.0	14
6.0	13
17.0	13
3.0	12
22.0	11
28.0	11
4.0	11
25.0	11
15.0	11
24.0	11
21.0	11
14.0	11
29.0	11
12.0	10
26.0	10
18.0	10
23.0	10
8.0	9
11.0	9
20.0	9
19.0	7
7.0	7
10.0	6
27.0	6
9.0	2

Conclusions:

Lab assignment successfully completed, aims achieved.