EPL442

Assignment 3

Loukas Papalazarou 1013129

1. Design choices

- For this project I used Python since RBF nets are extremely fast to train and do not require a high performance language.
- I chose to represent the hidden layer nodes as an object. Each node object has its own center (taken from the centerVectors.txt file), it's own sigma value and functions to update said values.
- The centers are randomly generated.
- The RBF net is also represented as an object with a list of hidden nodes, weights, learning rates and the corresponding functions for updating.
- The tools.py module provides some helper unrelated to the RBF modeling or training/testing.
- To run, just execute: **python main.py**

2. Control run

The following is a run of the algorithm with some parameters that I found produce some good results.

numHiddenLayerNeurons 5

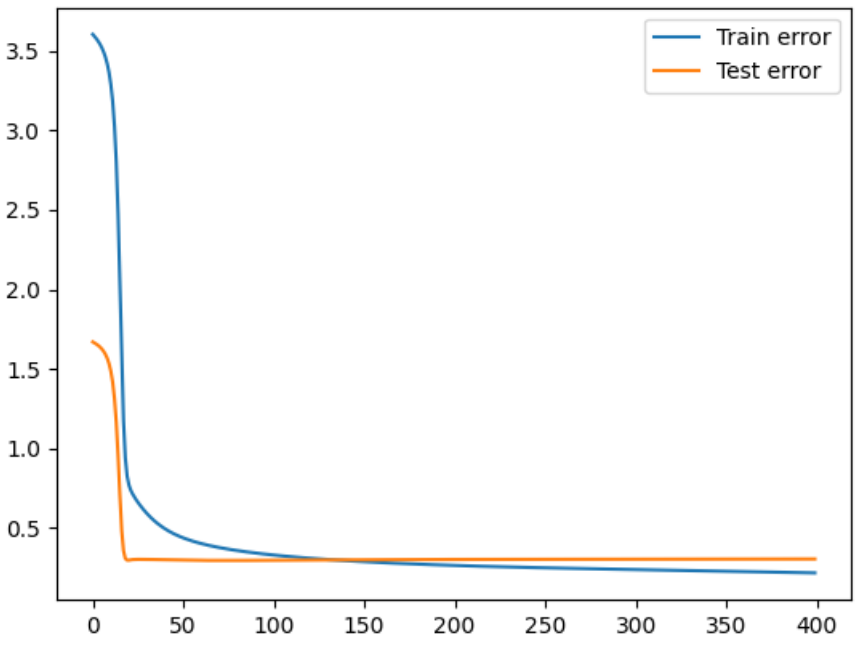
numInputNeurons 53

numOutputNeurons 1

learningRates 0.1 0.1 0.1

sigmas 1

maxIterations 400



Observations:

What we can see from this run is that the error starts at around 3.5 and stabilizes around 0.2 while going down exponentially. Same goes for the testing error but on a smaller scale just because of the volume of data. Expected results because of the relatively small learning rates and initial sigma.

3. Small learning rates, many epochs

We will now see a longer training cycle, with small learning rates.

numHiddenLayerNeurons 5

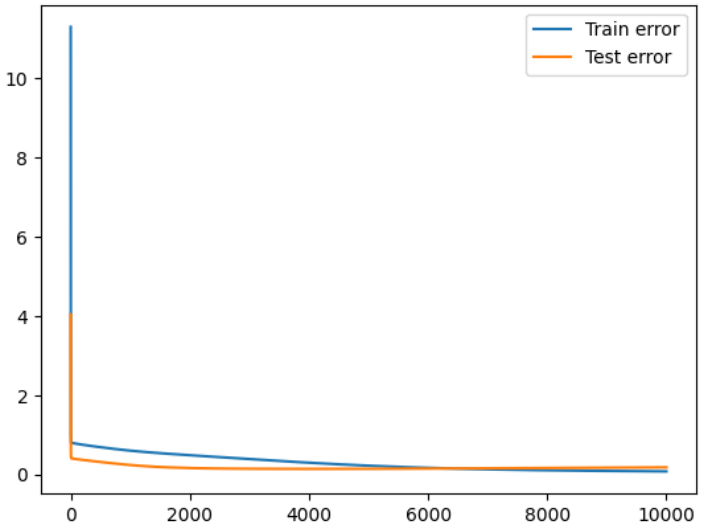
numInputNeurons 53

numOutputNeurons 1

learningRates 0.01 0.01 0.01

sigmas 1

maxIterations 10,000



Observations:

What we can see from this run is that the net dramatically drops its error in the first 1000 epochs and then very slowly the training and testing errors go down. Again, this is an expected result since the longer we are training for, the less the impact of the updates.

4. Many hidden nodes

In this run we increase the amount of hidden nodes to see what effect this has

numHiddenLayerNeurons 30

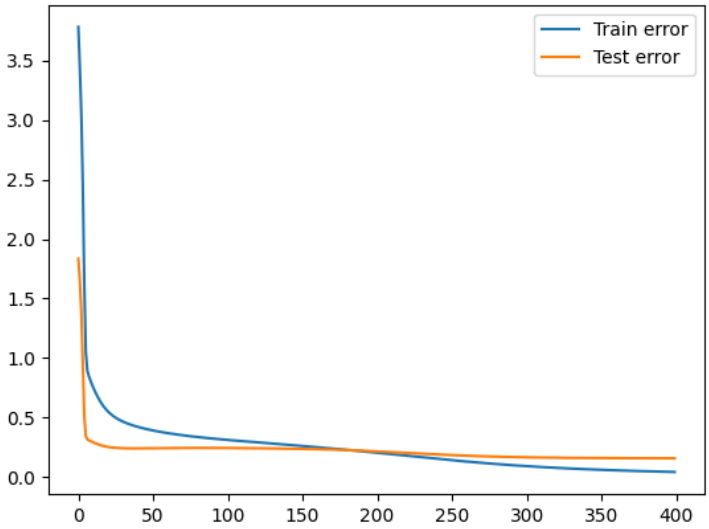
numInputNeurons 53

numOutputNeurons 1

learningRates 0.2 0.2 0.2

sigmas 1

maxIterations 400



Observations:

What we can see from this run is that nothing different from previous runs. Both the errors go down exponentially and settle around 0.3. This shows us that there is a cap to the number of hidden nodes that have an actual impact.

5. Large initial sigma

We will now increase the initial sigma value.

numHiddenLayerNeurons 5

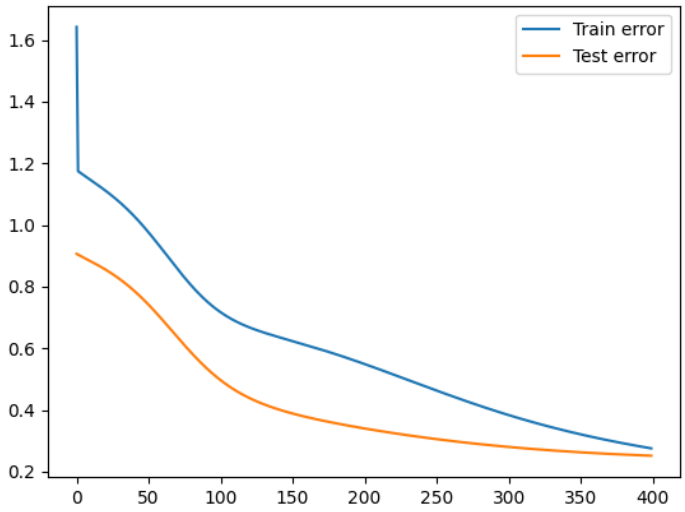
numInputNeurons 53

numOutputNeurons 1

learningRates 0.2 0.2 0.2

sigmas 5

maxIterations 400



Observations:

What we can see from this run is that the errors do go down but unlike the previous cases, this time the way the errors go down are way more chaotic and converge very late. We can explain this by the meaning of the sigma value, which is essentially the radius of the kernels, meaning a large sigma has a global influence meaning it causes changes that are not supposed to happen under normal conditions.