Artificial Neural Networks and Deep Architectures, DD2437

Short report on lab assignment 2

Radial basis functions, competitive learning and self-organisation

Jordan Wu, Daniel Janischowsky

Oktober 29, 2019

1 Main objectives and scope of the assignment

- Implement and train RBF network using least squares and delta learning approach. Investigate hyperparameters effects on performance.
- Implement competitive learning to use with RBF network (Hybrid Learning).
- Implement and visualize data using self-organizing maps. Use SOMs to solve "hard" problems efficiently.

2 Methods

The entire lab was done in Python and Numpy

3 RBF

Reflections: The lower bound for the number of training samples (N) is the number of rbf nodes (n). When N=n, error will be 0. This is because each rbf node can sit on top of a sample, and we can have an unique solution to the system.

3.1 Batch training with least squares - supervised learning of network weights

For this section and the next, weights are initialized at equal intervals between 0 and 2π, unless specified to be randomly initialized.

Function	Threshold	Error Achieved (MAE)	Hidden
Sin	0.1	0.080182	9
Sin	0.01	0.009497	13
Sin	0.001	0.000845	33
Square ($\phi = sign$)	0.1	0.063492	6
Square ($\phi = sign$)	0.01	0.0	13
Square ($\phi = sign$)	0.001	0.0	17

Table 1: Hidden Nodes vs MAE ($\sigma = 0.3$)

We were unable to get $MAE < 0.1$ for square wave with linear activation. With sign activation, we the network is able to achieve 0 error with 13 hiddens. This activation can be used in 2-class classifiers where samples are given labels 1 and -1.

3.2 Regression With Noise

We chose to use mean absolute error. It is a better measure than mean squared error when it comes to regression analysis, since errors are much more interpretable, since the relationship between the mistakes of a model is linear to the error measure.

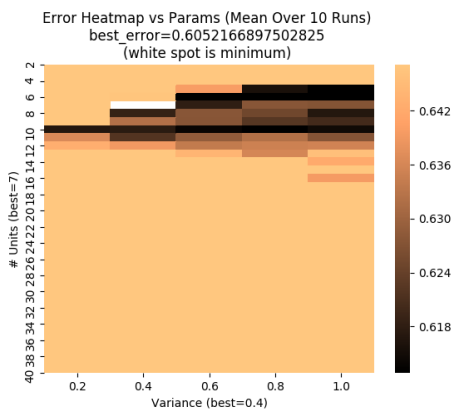
Minimizing squared loss can be done in closed form, so learning rate has no effect. With delta learning, we can see faster convergence with higher learning rate. However, when the learning rate is too high, we don't converge to the best solution as shown by the validation error.

Looking that the heatmaps for learning the sin function, we can see by increasing variance, we need to decrease number of RBF nodes. This is since each RBF node is able to account for more samples.

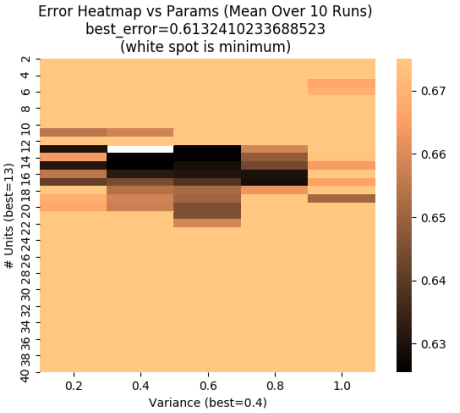
Our original RBF positioning method, at equal intervals, perform slightly better than randomly initializing (0.6 vs 0.68). Its also interesting that with sin, the random initialized network needed more hidden nodes. An intuition is that a some RBF nodes are thrown out after weighting so we get a more uniform distribution of nodes.

Function	Learning	Num Hidden	σ	Validation Error (MAE)
Sin	batch	7	0.4	0.108579
Square	batch	13	0.4	0.190366
Sin	delta	12	0.2	0.090481
Square	delta	5	0.6	0.372743

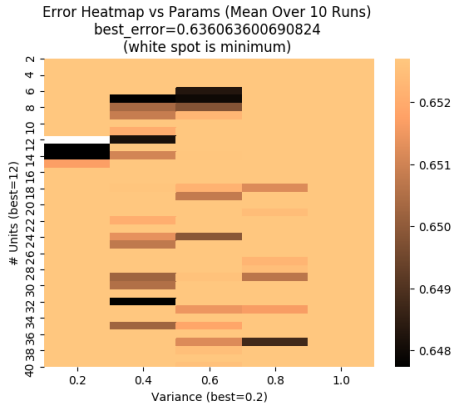
Table 2: Best Models on Clean Data



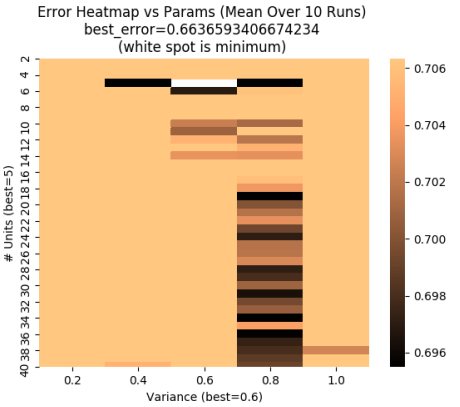
Batch Learning Sin



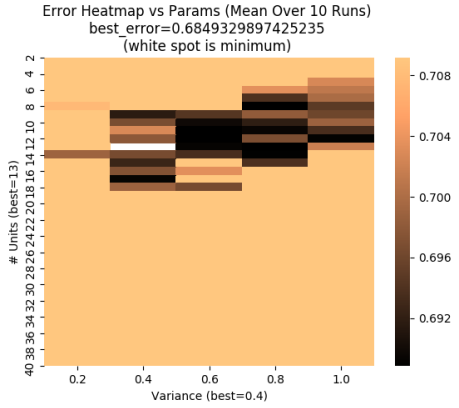
Batch Learning Square



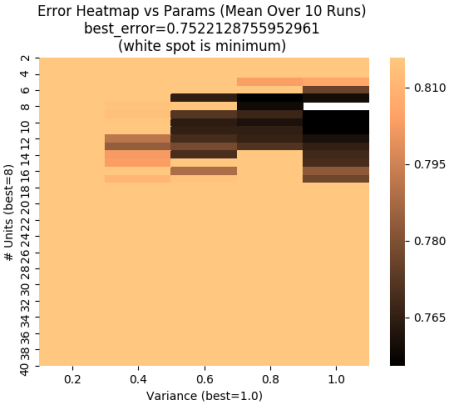
Delta Learning Sin



Delta Learning Square



Batch Learning Sin (Random Weights)



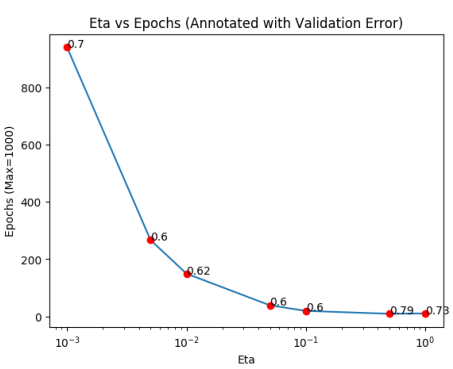
Batch Learning Square (Random Weights)

Figure 1: Params vs Validation Error Heatmap

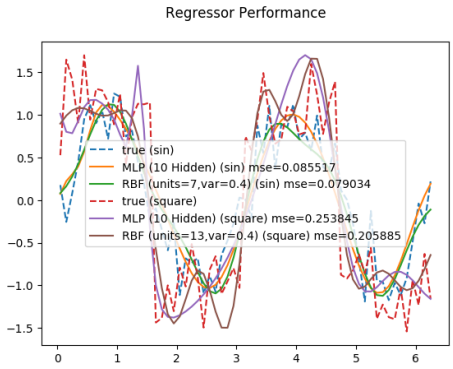
*x-axis should be σ (std) not variance.

We were unable to get the MLP to converge using SGD, so we use LBFGS. We

see that validation performance using RBF net is better than MLP. The RBF net can also be solved in closed form polynomial time using least squares, so its training is much faster.



Eta vs Epochs to Converge



RBF vs MLP

Figure 2: Performance Investigations

3.3 Competitive Learning

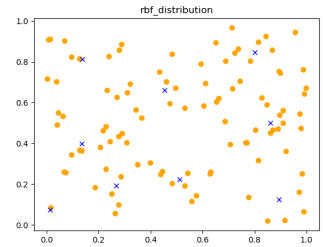
Winner takes all strategy:
We saw convergence due to decreasing weight updates each epoch. The RBFs are getting more and more equally distributed over the given area. Some of the RBFs are more often declared as winner then others. However we did not found any RBF that could be considered as dead and did not get any update.

shared winning strategy:
We implemented a simple shared winning algorithm in which every RBF got updated based on the distance to the sample. With this method we can see an even more equally distributed field.

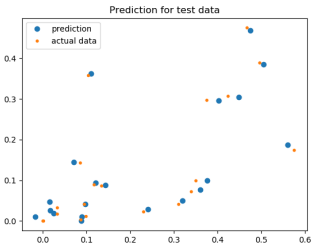
	one winner	shared winning
max distance	0.773	0.606
min distance	0.424	0.486
mean distance	0.559	0.528

Table 3: distribution of RBFs for different algorithms

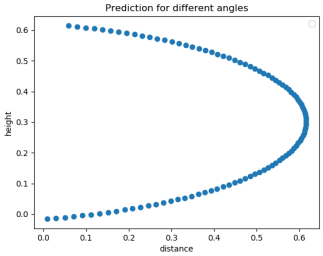
RBF network for a ballistic example:
We used 9 RBFs to classify the given data. Since we saw a shared winning approach slightly better performing, we took this also for the new RBF network. After training we achieved an error of 0.018.



RBF Distribution



Test vs Prediction



Output vs Angles (Fixed Velocity)

Figure 3: Ballistic Figures

4 SOM

4.1 Topological Ordering Animal Species

topologically_sorted = ['spider' 'housefly' 'beetle' 'grasshopper' 'butterfly' 'dragonfly' 'moskito' 'pelican' 'duck' 'penguin' 'ostrich' 'frog' 'seaturtle' 'crocodile' 'walrus' 'rat' 'rabbit' 'hyena' 'bear' 'dog' 'ape' 'lion' 'cat' 'skunk' 'bat' 'elephant' 'kangaroo' 'horse' 'camel' 'giraffe' 'pig' 'antelop']

The groups are insects, flying animals, winged animals, water animals, small mammals, medium mammals, then "long" animals.

4.2 Cyclic Tour

The SOM tries to find a ordering that puts "close" samples next to each other, this results in an approximate solution to the TSP problem. Since the weights are randomly initialized and result it slightly different solutions, we ran the SOM 10 times, then chose the one that gave us the minimum path.

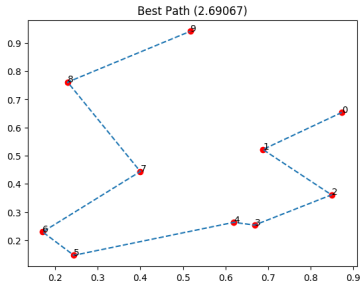
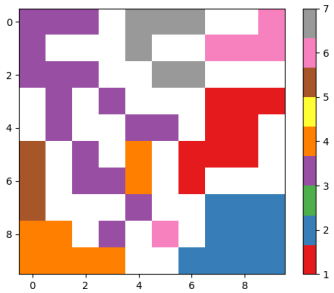
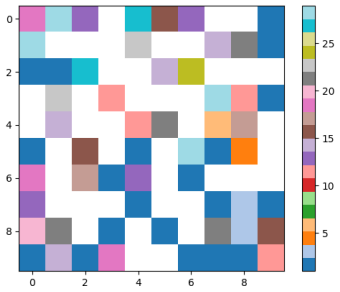


Figure 4: Cyclic Tour Best Path

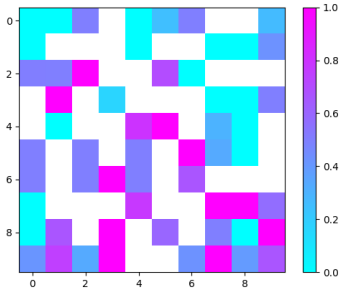
4.3 Data Clustering MP's Votes



Party Mapping (Mode)



District Mapping (Mode)



Sex Mapping (Mean)

Figure 5: 2D Vote Mappings

We see the mapping with respect to party gives the most obvious clusters. Which is intuitive since members in the same party will vote the same. Mapping with respect to district is not really organized, which is expected, since members from the same district should represent different peoples of a district. There is some semblence of clustering for sex. We can see males (pink) cluster around the bottom half, with females (blue) on the top-right.