

Clustering Animals and Congressmen through a Competitive Learning Network

Abstract

This paper explains the results and process of running an unsupervised learning technique called a **competitive learning network** (*see next section: What is a Competitive Learning Network*) first on a data set of 10 animals and their various features. We wish to see how well the competitive learning network segments the animals into their respective biological classes (e.g. birds, mammals, and reptiles). This is a reproduction of the experiment done in Knight [1]. Next I look at how well the competitive learning network performs on a data set composed of congressmen and how they voted on 16 bills in 1984 [2]. I am especially curious here to see if the competitive learning network can cluster the congress men into their respective political parties (e.g. Democrat, and Republican).

What is a Competitive Learning Network?

A competitive learning network (CLN) is an unsupervised learning technique that is used to find similarities in examples in a dataset and therefore suggest groupings of the examples. This is known as a clustering algorithm since it “clusters” the data examples together into similar groups. It is called unsupervised because we do not tell the algorithm how to group the examples together. It is left to its own devices to figure out any similarities in the data.

A CLN is made up of two layers of nodes; an input layer and an output layer. Each node in the input layer connects to every node in the output layer. Each connection has a weight assigned to it. The values assigned to these weights are where knowledge is stored in this network. The each node in the input layer is associated with a feature in the dataset. Each node in the output layer corresponds to a potential grouping of the data. To provide a clearer picture of how this all works, below are a table showing our animal dataset, and a diagram of how the CLN is set up.

Animal	Has Hair	Has Scales	Has Feathers	Flies	Lives in Water	Lays Eggs
Dog	1	0	0	0	0	0
Cat	1	0	0	0	0	0
Bat	1	0	0	1	0	0
Whale	1	0	0	0	1	0
Canary	0	0	1	1	0	1
Robin	0	0	1	1	0	1
Ostrich	0	0	1	0	0	1
Snake	0	1	0	0	0	1
Lizard	0	1	0	0	0	1
Alligator	0	1	0	0	1	1

Table 1. A data set containing animals and some animal attributes.

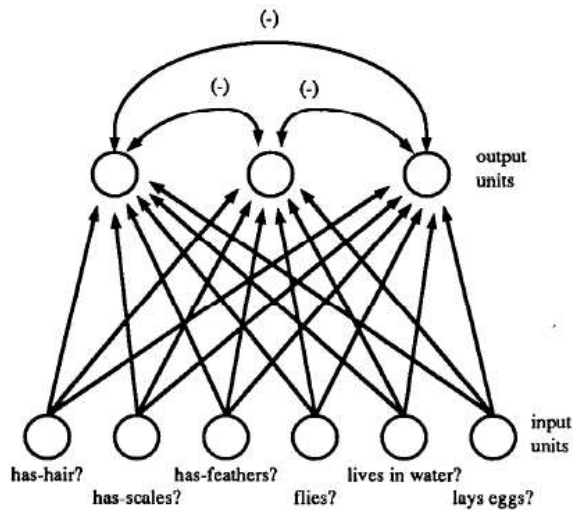


Figure 1. The CLN architecture for clustering these animals into three groups. This image comes from Knight [1].

The CLN learns by being having each record (stimuli) fed into the input layer and having the output nodes compete for control of that input. It may sound complicated but it's not. In this sense “competing” just means that the output node with the highest weighted sum of its inputs “wins”.

The winner gets all of his incoming weights from active nodes increased while the losers do not get their weights changed. The effect of updating the weights like this is that the winner then becomes more likely to win a future input with similar attributes.

There are a few different methods for increasing the weights and distributing the weights. The most popular method is to increase each of the winner's weights by following this formula¹:

$$weight_value = (learning_rate) * (input_value / number_non_zero_input_nodes) - (learning_rate * previous_weight_value)$$

¹ This paper breaks from the tradition of writing formulas as symbols and then explaining what the symbols mean elsewhere in the paper (if you're lucky!). Instead I believe it is easier for the reader to follow if formulas are written in a pseudo-code style with meaningful variable names.

The learning rate should be a small constant between 0 and 1. I used 0.25 in all experiments in this paper.

Running the CLN on the Animal Dataset

I did a total of four runs with different parameters on this dataset. Overall the CLN did an impressive job of clustering the animals into their respective groups of Mammals, Birds, and Reptiles. Below are the parameters and discussion of results for each run. The detailed results and weights can be found in appendix A.

Run 1 with 3 output nodes and using the 1/N constraint²:

Here we see that all animals get clustered into only two groups. We correctly group all Mammals together in output node 1, but Birds and Reptiles get grouped together in Output node 2 while output node 3 gets nothing. This does seem intuitive in a way, both Birds and Reptiles do lay eggs and none from either group have hair.

Run 2 with 3 output nodes but without using the 1/N constraint:

This time all of the animals are grouped correctly into their respective biological classes. Diving into the final weight values (see appendix A), we see that output node 1 gets strong weighting for has hair, output node 2 for has scales, and output 3 for has feathers. This certainly is a fool-proof way to tell these classes of animals apart.

Run 3 with 5 output nodes and using the 1/N constraint:

This gives us essentially the same result as run 1. Output nodes 3-5 have all of their weights end up at the same value of .1167 and never win any input stimuli. The pattern I detect is that the 1/N constraint makes us detect only two groupings for these animals.

Run 4 with 5 output nodes but without using the 1/N constraint:

Here we see the same correct results as run 2. Output nodes 4-5 have all of their weights decrease to an identical value and never win any input stimuli. This suggests that perhaps the CLN is good at finding the true natural groupings of input even when you implement it with extra input nodes. Perhaps it even suggests that if you are not sure how many clusters you want to find that you should err on the side of having too many output nodes.

Overall runs 2 and 4 gave us the best results. The most important lessons to draw from this are that not using the 1/N constraint works best for this data and that extra output nodes are not necessarily harmful.

Running the CLN on Congressional Voting Records

Here we look at whether a CLN can look at the voting records of US representatives and group the representatives by their political party.

This data comes from the machine learning repository [2] and it is comprised specifically of 1984 United States Congressional Voting Records for each of the U.S. House of Representatives Congressmen on the 16 important votes [2].

There were originally 435 records in this dataset however I removed any records where an attribute was missing (presumably because the congressman did not vote.) I did this

² This constraint means that the winner's incoming weights are updated such that the sum of all weights are allowed constrained to equal 1. When we run without this constraint, the weights are just changed by the aforementioned formula.

because I don't know how to express missing values to the CLN. I was left with 232 data records (The most devoted congressmen perhaps?)

I designed the CLN to have two output nodes because I knew that I wanted it to discover 2 political parties. I did use the 1/N constraint for this experiment because I did not have time to try both constraint settings. Turning off the constraint would be interesting future work however.

The results were very good as you can see below. Appendix B also shows the final weight values which show us some surprising and not so surprising generalizations about Republicans and Democrats. As you can see from table 2 below, the CLN made output node 1 the Democrat node, and output node 2 the Republican Node. It correctly classified 106 out of 124 Democrats and 101 out of 108 Republicans. I took a spot check of a few of the misclassified congressmen and they certainly did have a mixed voting record. Perhaps you could call them the moderates?

CLN Winner (Final Iteration)	Party	Count of Party
Winner 1		119
	democrat	18
	republican	101
Winner 2		113
	democrat	106
	republican	7

Table 2. How the CLN classified the congressmen.

Conclusions

The CLN performed very well on both datasets and when it did fail, it happened on ambiguous, middle of the group type cases such as congressional moderates. In the animal dataset we did a much better when not using the 1/N constraint and using either 3 or 5 output nodes did not make a difference in our final groupings. Overall, the experiments run in this paper suggest that the CLN is a useful clustering algorithm.

References

1. K. Knight, "Connectionist ideas and algorithms," Communications of the ACM, vol. 33, no. 11, pp. 59--74, 1990.
2. Newman, D.J. & Hettich, S. & Blake, C.L. & Merz, C.J. (1998). UCI Repository of machine learning databases [<http://www.ics.uci.edu/~mlearn/MLRepository.html>]. Irvine, CA: University of California, Department of Information and Computer Science. (*Specific dataset URI: <ftp://ftp.ics.uci.edu/pub/machine-learning-databases/voting-records>*)

Appendix

A. Detailed Results from the Animal Dataset

Run 1: 3 outputs and 1/N constraint On

Final Weights

<u>Attribute Name</u>	<u>Output 1 Value</u>	<u>Output 2 Value</u>	<u>Output 3 Value</u>
Has Hair	0.6800	0.0000	0.1667
Has Scales	0.0000	0.3010	0.1667
Has Feathers	0.0000	0.1203	0.1667
Flies	0.1371	0.0561	0.1667
Lives in Water	0.1829	0.1014	0.1667
Lays Eggs	0.0000	0.4212	0.1667

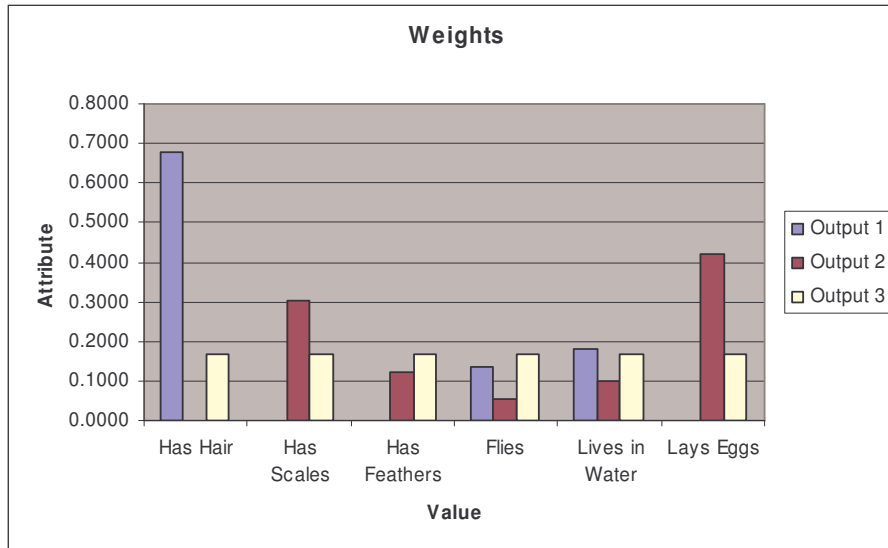


Figure 2. Graph of final weight values for each output.

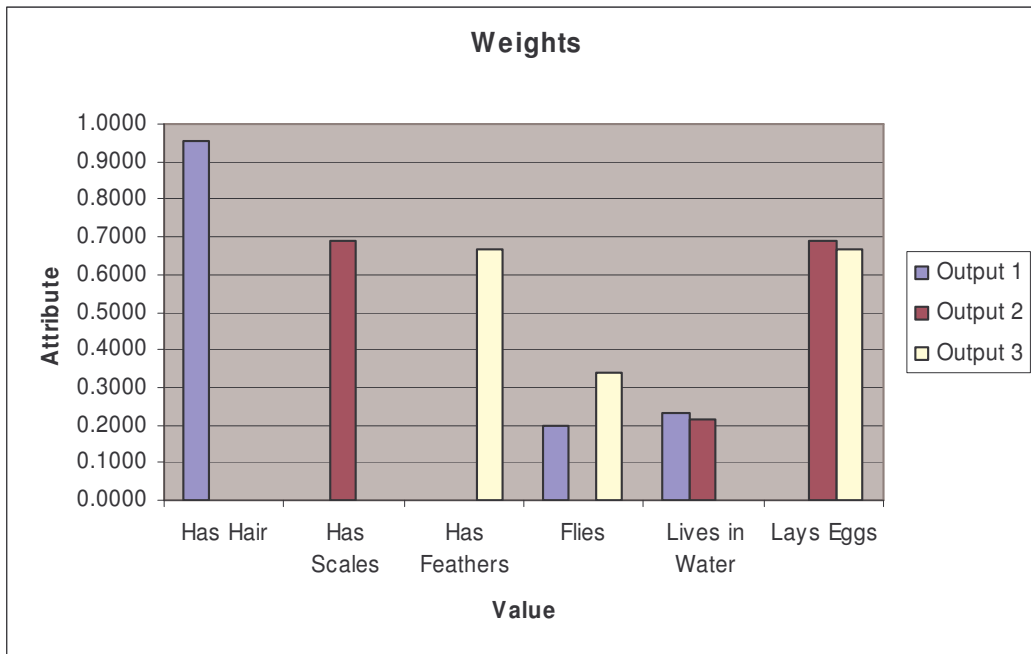
Final Winner by Stimulus

CLN Winner (Final Iteration)	Animal
Winner 1	Dog
Winner 1	Cat
Winner 1	Bat
Winner 1	Whale
Winner 2	Canary
Winner 2	Robin
Winner 2	Ostrich
Winner 2	Snake
Winner 2	Lizard
Winner 2	Alligator

Run 2: 3 outputs and 1/N constraint Off

Final Weights

<u>Attribute Name</u>	<u>Output 1 Value</u>	<u>Output 2 Value</u>	<u>Output 3 Value</u>
Has Hair	0.9520	0.0000	0.0000
Has Scales	0.0000	0.6905	0.0000
Has Feathers	0.0000	0.0000	0.6654
Flies	0.1967	0.0000	0.3382
Lives in Water	0.2344	0.2154	0.0000
Lays Eggs	0.0000	0.6905	0.6654



Final Winner by Stimulus

CLN Winner (Final Iteration)	Animal
Winner 1	Dog
Winner 1	Cat
Winner 1	Bat
Winner 1	Whale
Winner 3	Canary
Winner 3	Robin
Winner 3	Ostrich
Winner 2	Snake
Winner 2	Lizard
Winner 2	Alligator

Run 3: 5 outputs and 1/N constraint On

Final Weights

<u>Attribute Name</u>	<u>Output 1 Value</u>	<u>Output 2 Value</u>	<u>Output 3 Value</u>	<u>Output 4 Value</u>	<u>Output 5 Value</u>
Has Hair	0.6800	0.0000	0.1667	0.1667	0.1667
Has Scales	0.0000	0.3010	0.1667	0.1667	0.1667
Has Feathers	0.0000	0.1203	0.1667	0.1667	0.1667
Flies	0.1371	0.0561	0.1667	0.1667	0.1667
Lives in Water	0.1829	0.1014	0.1667	0.1667	0.1667
Lays Eggs	0.0000	0.4212	0.1667	0.1667	0.1667

Final Winner by Stimulus

CLN Winner (Final Iteration)	Animal
Winner 1	Dog
Winner 1	Cat
Winner 1	Bat
Winner 1	Whale
Winner 2	Canary
Winner 2	Robin
Winner 2	Ostrich
Winner 2	Snake
Winner 2	Lizard
Winner 2	Alligator

Run 4: 5 outputs and 1/N constraint Off

Final Weights

<u>Attribute Name</u>	<u>Output 1 Value</u>	<u>Output 2 Value</u>	<u>Output 3 Value</u>	<u>Output 4 Value</u>	<u>Output 5 Value</u>
Has Hair	0.9520	0.0000	0.0000	0.4082	0.4082
Has Scales	0.0000	0.6905	0.0000	0.4082	0.4082
Has Feathers	0.0000	0.0000	0.6654	0.4082	0.4082
Flies	0.1967	0.0000	0.3382	0.4082	0.4082
Lives in Water	0.2344	0.2154	0.0000	0.4082	0.4082
Lays Eggs	0.0000	0.6905	0.6654	0.4082	0.4082

Final Winner by Stimulus

CLN Winner (Final Iteration)	Animal
Winner 1	Dog
Winner 1	Cat
Winner 1	Bat
Winner 1	Whale
Winner 3	Canary
Winner 3	Robin
Winner 3	Ostrich
Winner 2	Snake
Winner 2	Lizard
Winner 2	Alligator

B. Results from the Congressional Dataset

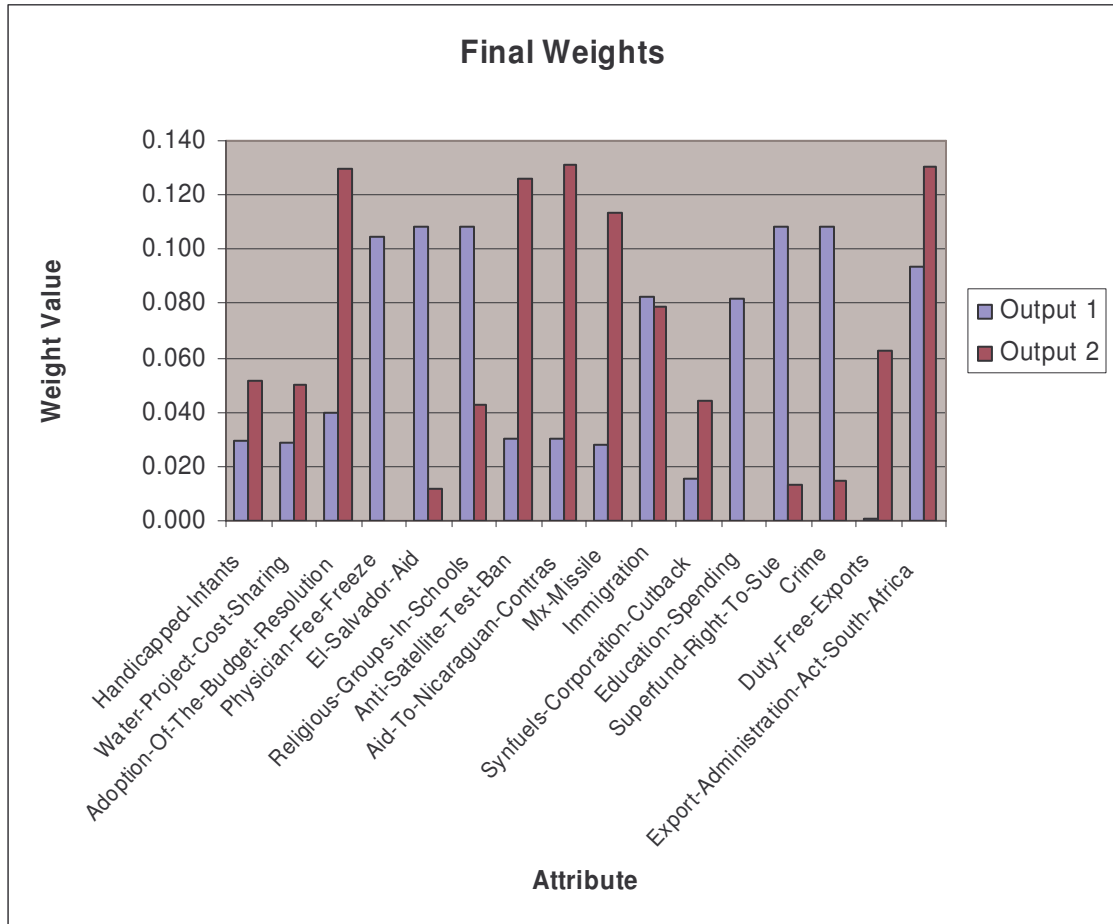


Figure 3. Here are the final CLN weights after iterating on the congressional dataset. Output node 1 mostly won the Republicans and output node 2 the Democrats. I leave it as an exercise to the reader to use these weights to make some fun generalizations about the two parties (at least for 1984).