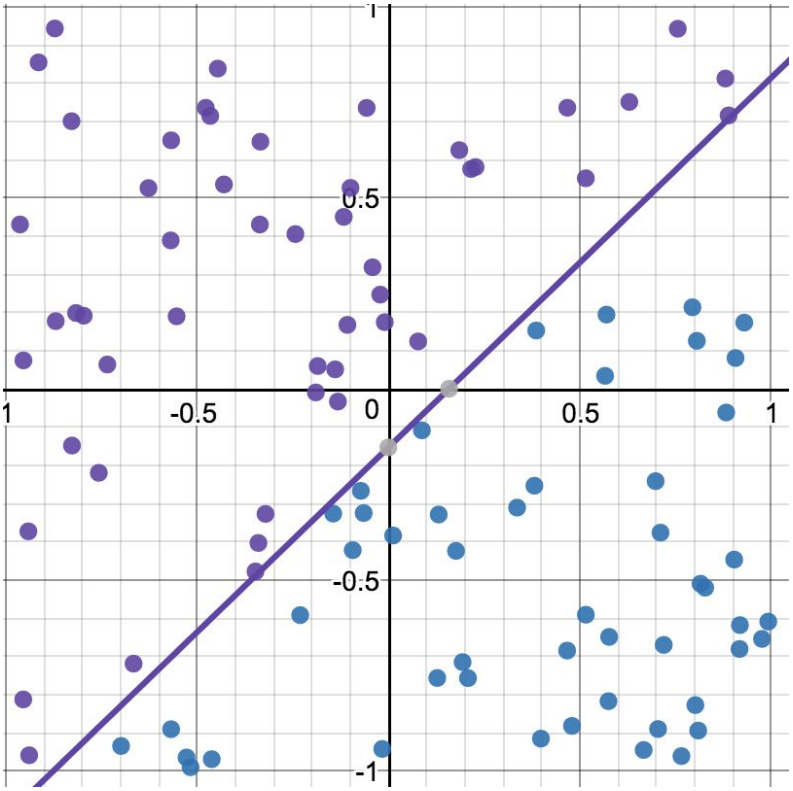
Martín Alejandro Molinero Morales
A01204818

Ann Report

Lab part 1



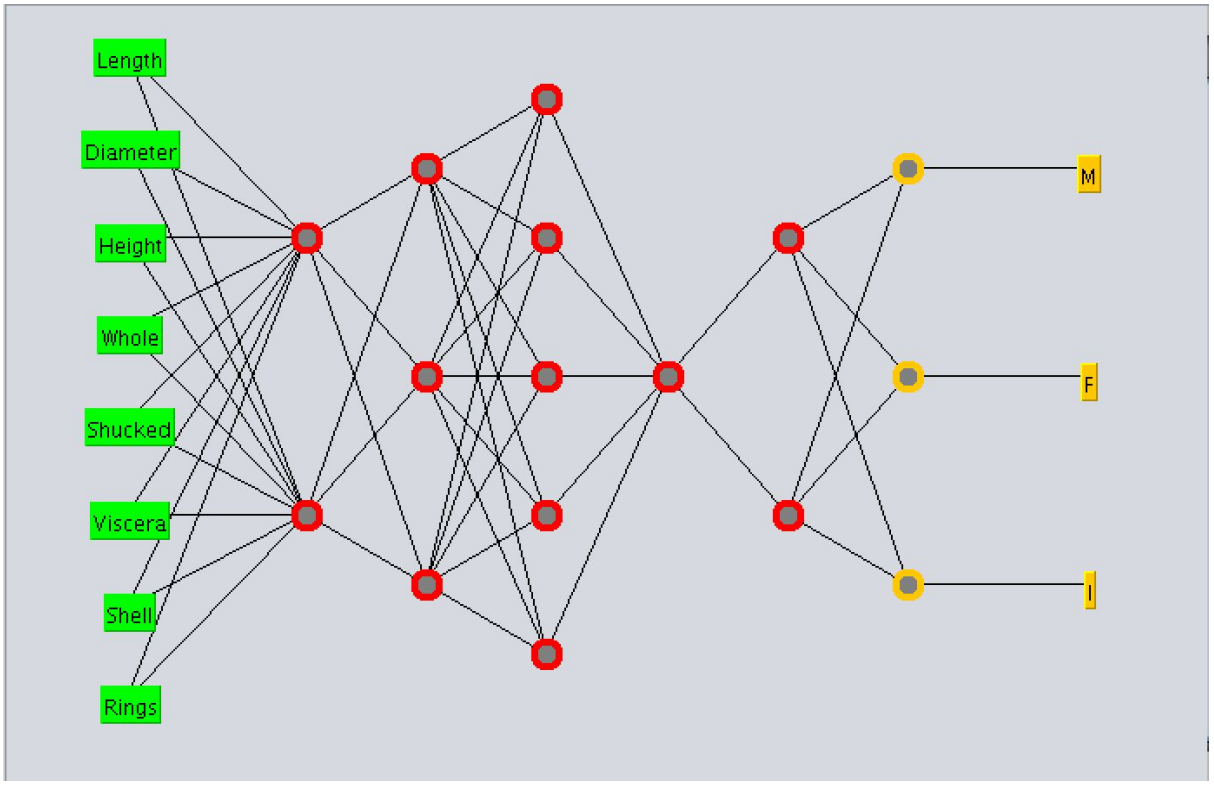
Line equation is 0.4472027037534373y =0.4324073516072876 x - 0.06864091860791924 where -y and x are given by the original input weights on the perceptron and b is given by the bias.
Blue ones are 1 output
Purple ones are 0 output

Lab part 2
Using the abalone training set, I wanted to classify the abalone toknow the sex of them based on the other parameters. The attributes are:
Sex {M,F,I}
Length NUMERIC
Diameter NUMERIC
Height NUMERIC
Whole NUMERIC
Shucked NUMERIC
Viscera NUMERIC
Shell NUMERIC
Rings NUMERIC

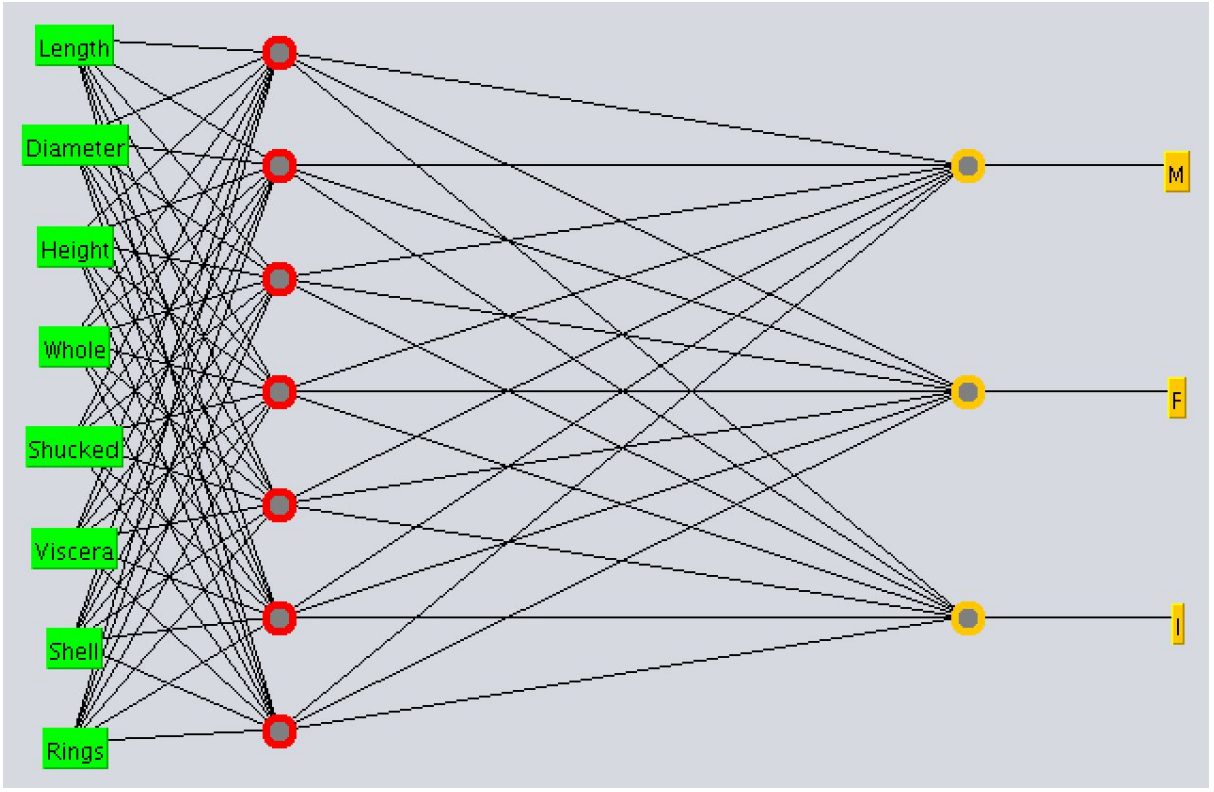
Martín Alejandro Molinero Morales
A01204818

Images

#1

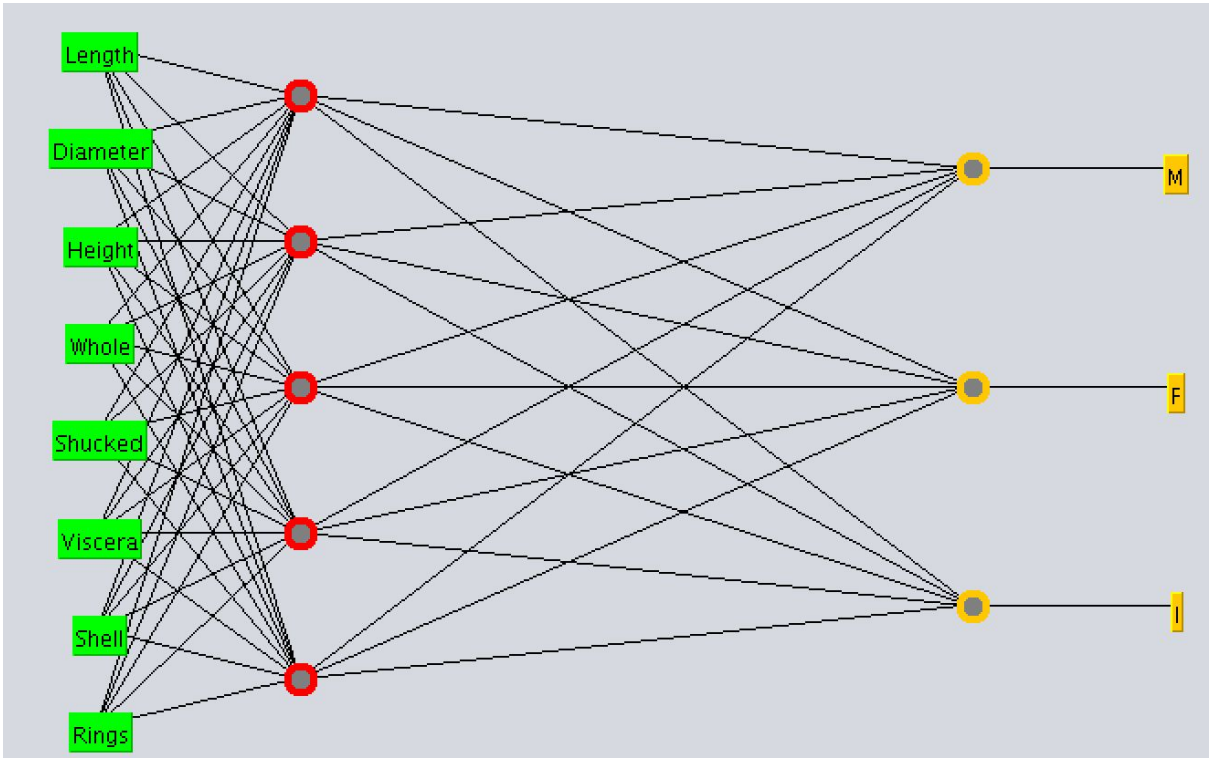


#2

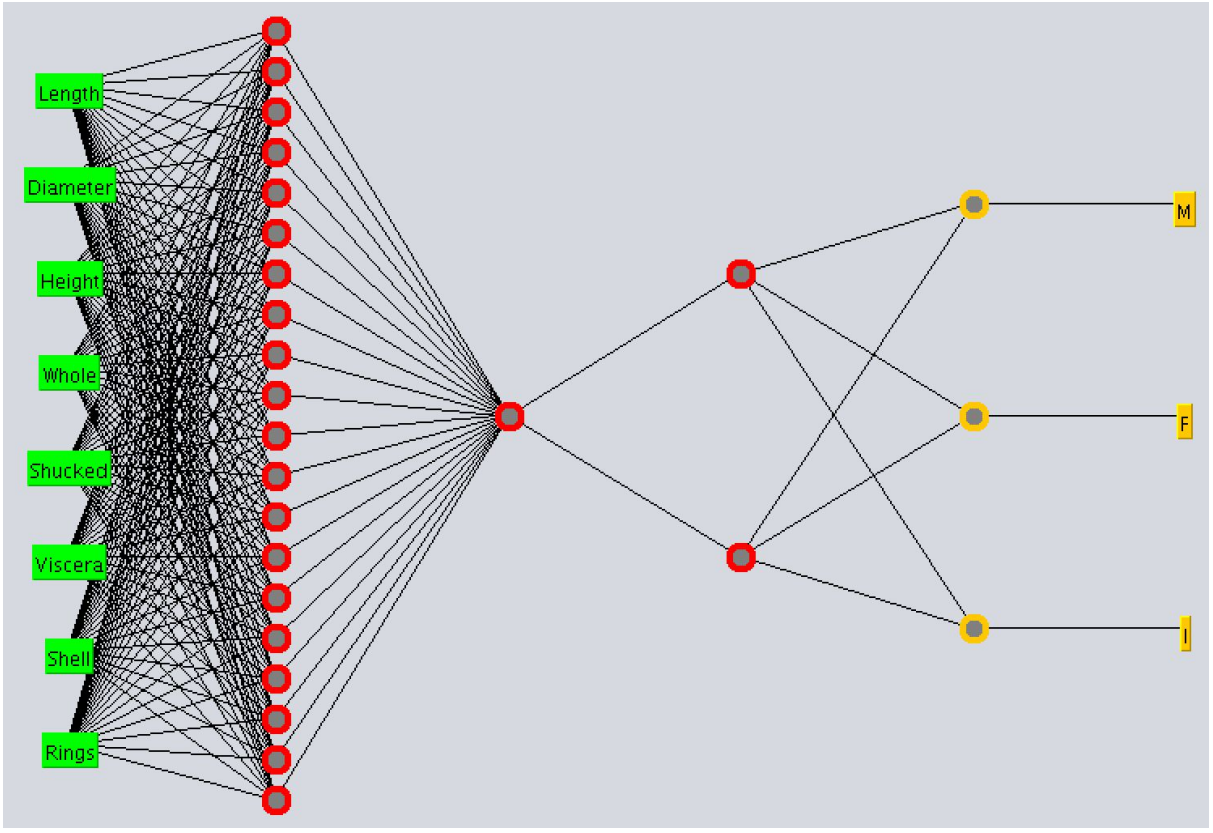


#3

Martín Alejandro Molinero Morales
A01204818

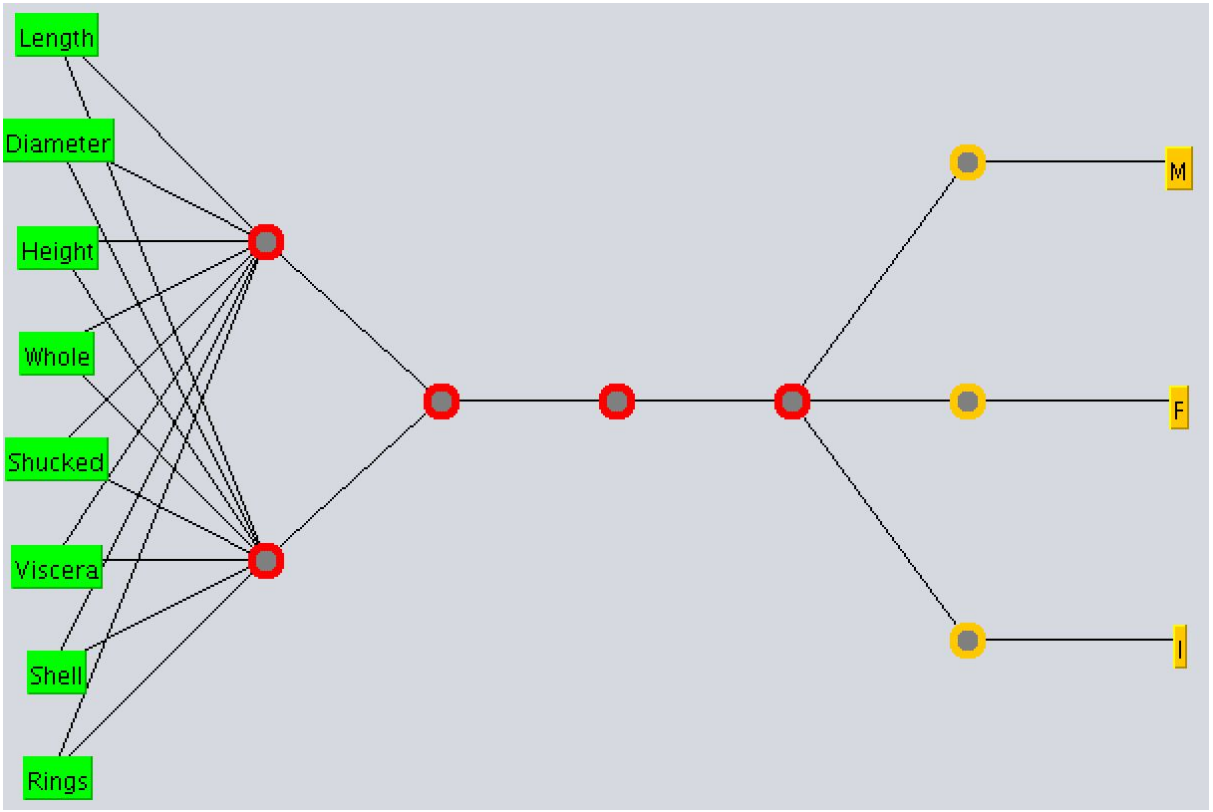


#4



#5

Martín Alejandro Molinero Morales
A01204818



case #	learning rate	epochs	time to build (s)	correctly classified	incorrectly classiied	hidden layers
1	0.2	500	12.79	36.45%	63.54%	2,3,5,1,2
2	0.3	500	8.3	57.87%	42.12%	7
3	0.1	500	17.57	56.69%	43.3033%	5
4	0.3	1000	37.82	55.7%	44.2%	20,1,2
5	0.1	500	46.77	36.4%	63.5%	2,1,1,1

I think, mainly the learning rate and the structure of the network is what determines how it behaves with the input data. For example, if we have a very complex structure like the #4, it takes a lot of time to balance the weights for the 20 node hidden layer. I realized that, mainly what made the difference between having a good correctly classified percentage and not having it, is, instead of having a lot of hidden layers, just having one hidden layer with a little bit of more nodes.
Also I think that on example #4 I was forcing the overfitting, because even when i had a larger number of nodes in comparison to #2 and #3, i did not reduced the learning rate and also I increased the number of epochs. In some structures, like this last one, I realized that the error per epoch was decreasing more when I put a larger iterations number, something that didn't happen for example with case #4, in which the error per epochs arrives to a point where its not reduced and therefore we end up having the same error with 500 epochs than with more than 2000.

Martín Alejandro Molinero Morales
A01204818

Aspects for reflection:

- Explanations as to what are ANNs good for.

They are useful to classify data with a lot of parameters that are important for calculating the output category where the data row fits based on its similarity. In order to work, it should be linearly separable in order for the algorithm to work properly.

- Where would you use them?

I'd use them with labeled datasets that have more attributes to take in account than the tree algorithms we saw earlier in this course. Mainly, I will use them to classify things based on their characteristics (nominal or numerical).

- Are they worth the effort implementing or not?

The power of the neural networks is big, even though we now have some libraries that can make our life a little bit easier by not having to code every single type of neural networks we for different type of problems. They are more useful for more real applications than the initial algorithms on this course and its interesting to study how they behave and the numerical approach that is behind them. I think the benefits of using them totally worth the effort that takes to implement them.

- What kinds of problems do they not solve?

Datasets with unlabeled data or that are not linearly separable.