

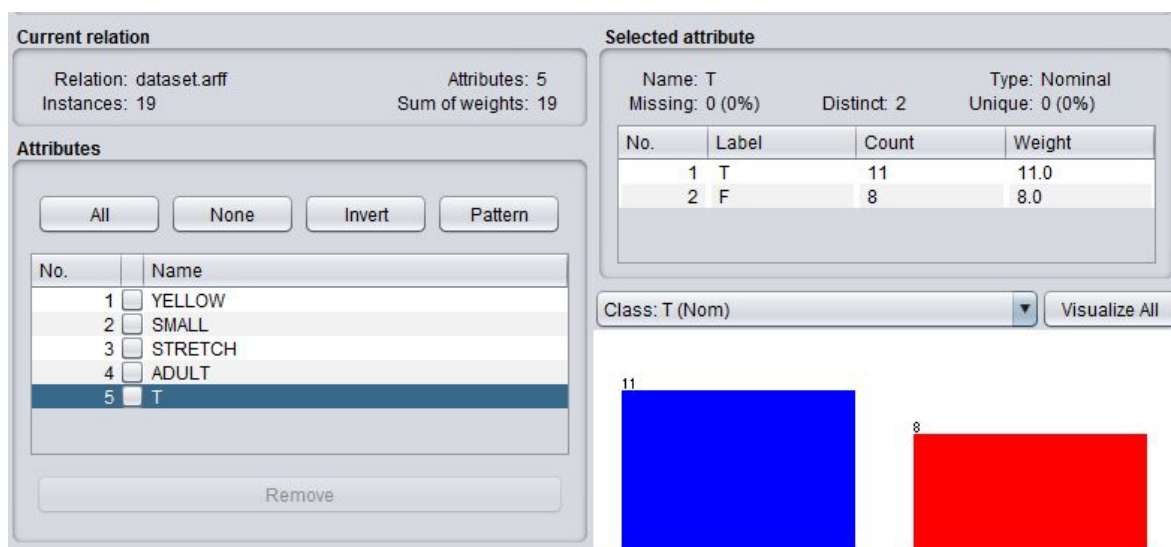
Perceptron / Neural Networks

For this lab we decided to use the [Balloons Data Set](#) to implement a multilayer perceptron using Weka.

Dataset:

YELLOW, SMALL, STRETCH, ADULT, T
YELLOW, SMALL, STRETCH, CHILD, T
YELLOW, SMALL, DIP, ADULT, T
YELLOW, SMALL, DIP, CHILD, F
YELLOW, SMALL, DIP, CHILD, F
YELLOW, LARGE, STRETCH, ADULT, T
YELLOW, LARGE, STRETCH, CHILD, T
YELLOW, LARGE, DIP, ADULT, T
YELLOW, LARGE, DIP, CHILD, F
YELLOW, LARGE, DIP, CHILD, F
PURPLE, SMALL, STRETCH, ADULT, T
PURPLE, SMALL, STRETCH, CHILD, T
PURPLE, SMALL, DIP, ADULT, T
PURPLE, SMALL, DIP, CHILD, F
PURPLE, SMALL, DIP, CHILD, F
PURPLE, LARGE, STRETCH, ADULT, T
PURPLE, LARGE, STRETCH, CHILD, T
PURPLE, LARGE, DIP, ADULT, T
PURPLE, LARGE, DIP, CHILD, F
PURPLE, LARGE, DIP, CHILD, F

We loaded the dataset in the preprocess module, and could see the instances, attributes and sum of all the weights.



Current relation

Relation: dataset.arff
Instances: 19

Attributes: 5
Sum of weights: 19

Attributes

All None Invert Pattern

No.	Name
1	<input type="checkbox"/> YELLOW
2	<input type="checkbox"/> SMALL
3	<input type="checkbox"/> STRETCH
4	<input type="checkbox"/> ADULT
5	<input checked="" type="checkbox"/> T

Remove

Selected attribute

Name: T
Missing: 0 (0%)
Distinct: 2
Type: Nominal
Unique: 0 (0%)

No.	Label	Count	Weight
1	T	11	11.0
2	F	8	8.0

Class: T (Nom) Visualize All

11 8

Using the Multilayer Perceptron function in Weka, we can load and run the multilayer perceptron for the dataset mentioned above. This is the result:

Classifier

Choose **MultilayerPerceptron** -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H 5 -G -R

Test options

☒ Use training set
☐ Supplied test set Set...
☐ Cross-validation Folds 10
☐ Percentage split % 66
More options...

(Nom) T

Start Stop

Result list (right-click for options)

- 20:20:11 - functions.MultilayerPer
- 20:20:28 - functions.MultilayerPer
- 20:23:19 - functions.MultilayerPer
- 20:25:14 - functions.MultilayerPer

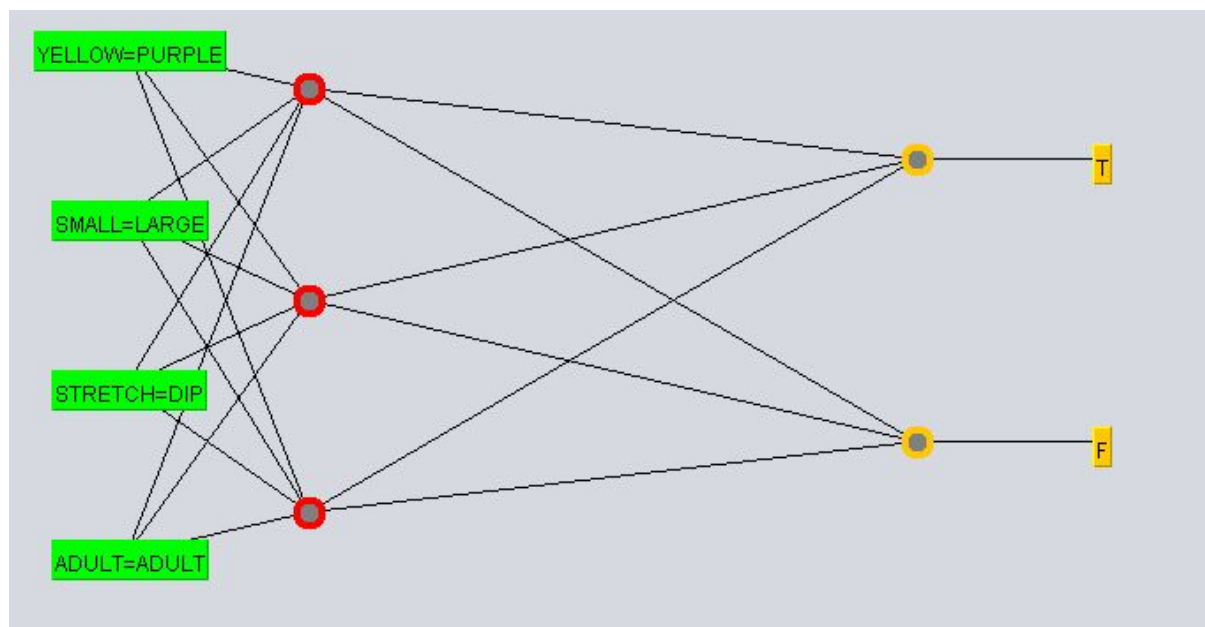
Classifier output

```
Root mean squared error          0.0154
Relative absolute error          2.9058 %
Root relative squared error      3.1102 %
Total Number of Instances       19

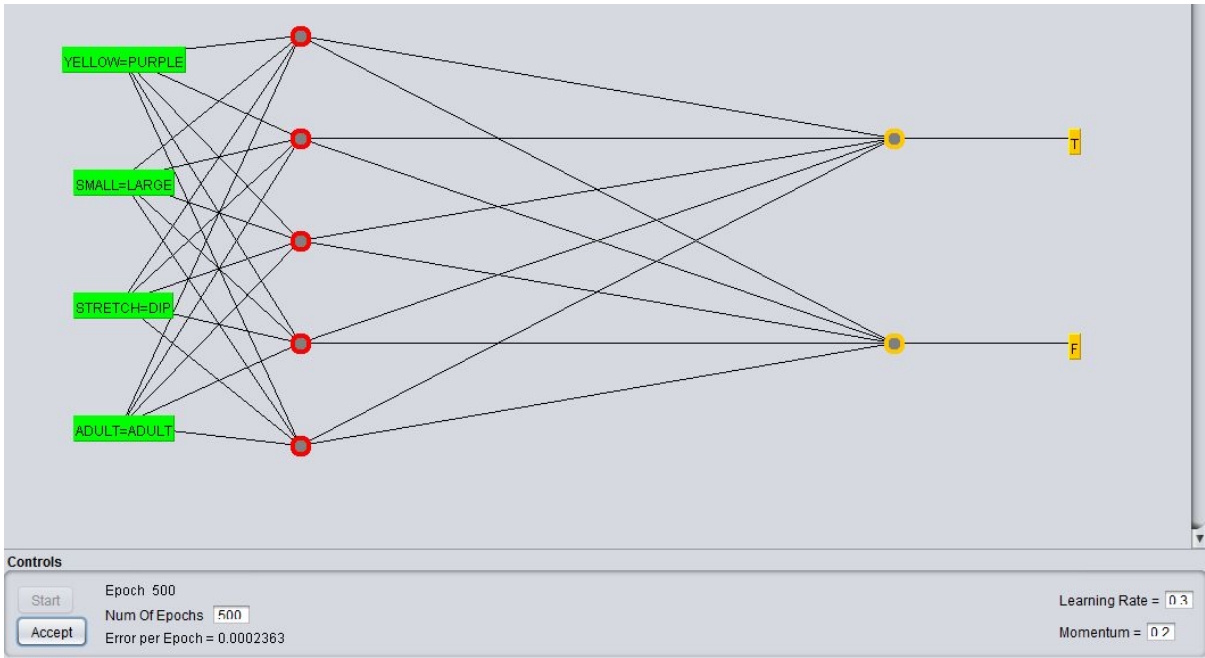
=== Detailed Accuracy By Class ===
               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC
               1.000    0.000    1.000    1.000    1.000    1.000
               1.000    0.000    1.000    1.000    1.000    1.000
Weighted Avg.   1.000    0.000    1.000    1.000    1.000    1.000

=== Confusion Matrix ===
  a  b  <-- classified as
11  0  |  a = T
 0  8  |  b = F
```

This is the result of the perceptron for the dataset, where there are 3 hidden nodes.



We added 2 more hidden nodes and this was the result:



All the evaluation was done with a learning rate of 0.3 and with a training time (epoch) of 500 and the value result was 100%.

=== Evaluation on training set ===

Time taken to test model on training data: 0 seconds

=== Summary ===

Correctly Classified Instances	19	100	%
Incorrectly Classified Instances	0	0	%
Kappa statistic	1		
Mean absolute error	0.0202		
Root mean squared error	0.0217		
Relative absolute error	4.1423	%	
Root relative squared error	4.4023	%	
Total Number of Instances	19		

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000	T
	1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000	F
Weighted Avg.	1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000	

=== Confusion Matrix ===

```
a b <-- classified as
11 0 | a = T
 0 8 | b = F
```

Consult the complete output [HERE](#)

An Artificial neural network are used in computer science and other disciplines in order to perform tasks that are difficult for a machine to do it. Using ANN has many advantages such as record-breaking accuracy, image and sound recognition. It also as a nonlinear model, it is easy to use and understand compared to statistical methods. Also if we used ANN with Backpropagation learning algorithm we can solve various classification and forecasting problems.

It also requires less formal statistical training and ability to implicitly detect complex nonlinear relationships between dependent and independent variables.

ANN has so many applications some examples are the following:

- Aerospace – Autopilot aircrafts, aircraft fault detection.
- Electronics – Code sequence prediction, IC chip layout, chip failure analysis, machine vision, voice synthesis.
- Medical – Cancer cell analysis, EEG and ECG analysis, prosthetic design, transplant time optimizer.
- Software – Pattern Recognition in facial recognition, optical character recognition.
- Speech – Speech recognition, speech classification, text to speech conversion
- Financial – Real estate appraisal, loan advisor, mortgage screening, corporate bond rating, portfolio trading program, corporate financial analysis.

ANN are worth the effort only if the problem is too complex for other alternatives such as linear regression or decision trees. Otherwise, an ANN would be overkill and would actually make the program harder to maintain because of ANN's usual illegibility compared to other forms of machine learning.

In their pure form, ANNs do not solve problems that require unsupervised learning. In practice, they can also be prohibitively inefficient with solving computationally complex algorithms, such as problems that deal with precise probabilities where a Bayes network would be a far more appropriate solution.

REFERENCES:

<https://www.youtube.com/watch?v=mo2dqHbLpQo> → Weka tutorial

https://www.tutorialspoint.com/artificial_intelligence/artificial_intelligence_neural_networks.htm