UNIVERSITY OF NEW SOUTH WALES

COMP9417

MACHINE LEARNING AND DATA MINING

Machine Learning Project Report

Authors:	*Student Numbers:*
Gervasio P. S. Neto	5050769
Ruan de Menezes Costa	5050761

March 24, 2015

1 Parsing input

To initialize either the Perceptron or the Linear Unit it is necessary to have a **.arff* file that contains the attributes' names and respective types and the actual data.

Real attributes are identified by the keyword "real" in the attribute line. Discrete attributes have their names followed by the possible values that can be assumed. Boolean attributes are viewed as attributes that can only have values 0 and 1.

The expected format is:

```
%Comment lines begin with a percentage mark.
@relation data-set-name
    @attribute real_attribute real
    @attribute discrete_attribute {v1, v2, v3}
    @attribute boolean_attribute {0,1}
    @attribute target
    @data
    real_value, discrete_value, boolean_value, target_value
```

When initializing a Linear Unit, it is expected that the target value attribute will be a *real* attribute (as to make numerical prediction possible). Likewise, when initializing the Perceptron, it's expected the target value (and all the other attributes) are boolean.

If, while parsing a data point, we realized the data was incomplete (e.g there are less values in the data line than there are attributes or one of the values is a '?') we discard the data point.

After opening the input file and going through all the lines, a list of data points and target values is compiled. These lists are then passed to the functions responsible for prediction.

2 Perceptron

Due to the visualization, the perceptron implemented is a step by step version. At each time the caller wants the perceptron to run one step, that is, go through one example. The perceptron then updates it's state (current example, current weights, current sum and current output), so the visualizer can grab these results and display them. The learning rate is set at 0.1. The learning rule was implemented as in the slides, $\Delta w = sign * \alpha * x_i$, where sign will be negative if the output is larger than the target or negative otherwise.

2.1 Results

The perceptron was applied to two 3 8-input boolean functions. The 8-input OR, the 8-input AND (which are linearly separable) and some random 8-input non linearly separable boolean function. The functions are at the inputs folder. NOTE: There are $2^{2^8} \approx 1.15 \text{ x } 10^{77}$ 8-input boolean functions. Of these, only 17561539552946 are linearly independent. To see that OR and AND are linearly separable, you just need to think of the boolean hypercube.

Results for the AND function:
$w_0 : -3.4, w_1 : 0.1, w_2 : 0.1\, w_3 : 0.3, w_4 : 0.4\, w_5 : 0.5\, w_6 : 0.6, w_7 : 0.7, w_8 : 0.7999999999999999$
Number of steps necessary: 56578
Time for training: 32 ms

Results for the OR function:
$w_0 : 0.0, w_1 : 0.1, w_2 : 0.1, w_3 : 0.1, w_4 : 0.1, w_5 : 0.1, w_6 : 0.1, w_7 : 0.1, w_8 : 0.1$
Number of steps necessary: 512
Time for training: 0 ms

As expected, the perceptron never finished training on the non-linearly separable function.

3 Linear Unit

The Linear Unit implements Gradient Descent using Batch Mode, meaning it will update the increments Δw for each data point before updating the actual weights. It will repeat this process for a pre-defined amount of time - *epochs*. This number is entered as input to the program.

We deal with discrete values in the following manner: If there are n possible values the attribute can have, then we have n boolean inputs, each corresponding to a particular value. The boolean input corresponding to the actual discrete attribute value in the observed data point is set to 1 and the other $n-1$ are set to 0.

We also deal with very different values in the real attributes. For instance, in the *autoMpg.arff* data set, the attribute *acceleration* is always much smaller (in absolute values) then the *weight* attribute. To stop this from skewing results, we normalize the real attributes before passing them to the Linear Unit. We put them in the 0 to 1 range using the formula:

$$normalized\_value = \frac{actual\_value - smallest\_value}{biggest\_value - smallest\_value}$$

The update formula is implemented as in the slides. The learning rate used is 10^{-3} and 80% of the data is used for training while the remaining 20% are used for validation.

3.1 Results

Since we randomly select which of the data points will be in the training or validation set, there are small variances in the observed errors each time the unit is run. Running it 10 times, with 2000 epochs each time, we observed the following error values:

	Average Value	Standard Deviation
Mean Absolute Error	2.206	0.145
Average Absolute Value	0.329	0.028
Mean Squared Error	9.093	1.488
Root Mean Squared Error	3.063	0.374

4 Visualizer

The visualization has the same layout both for the perceptron and the linear unit. It displays the current example being examined, the current weights, current weighted sum and current output (for the linear unit the former two will be the same). The default delay between the displays of the states is 2s.

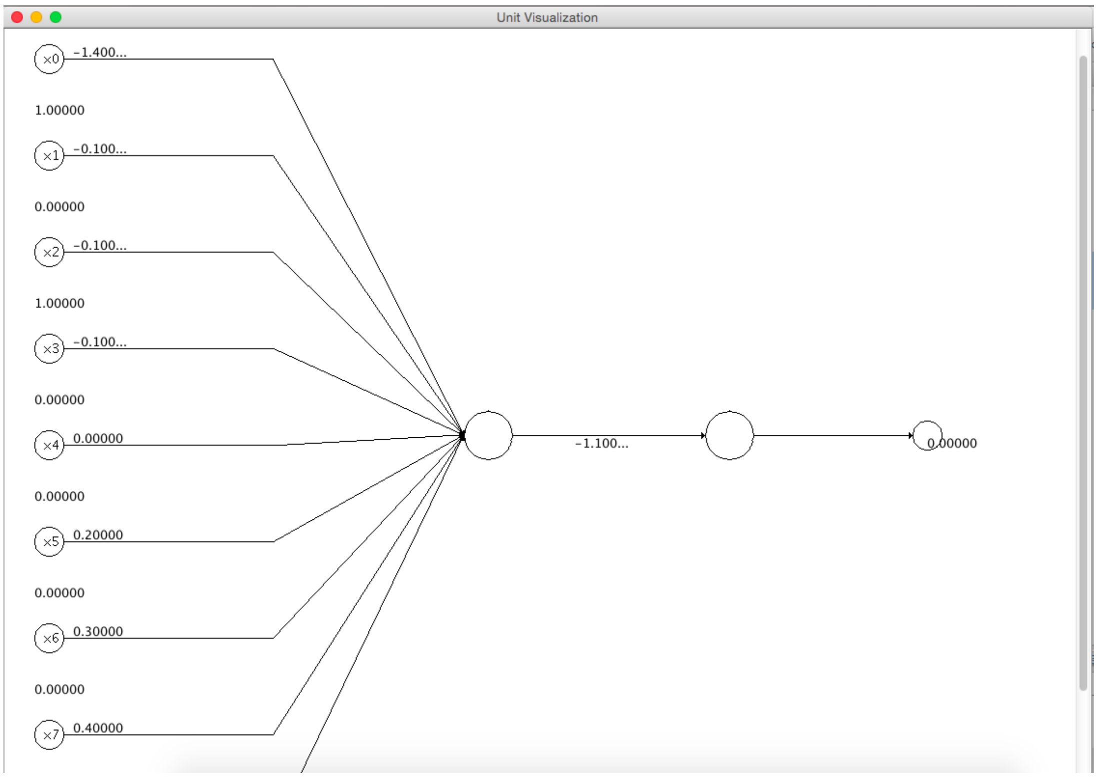

Figure 1: Visualizer