Machine Learning Project 2

jjjj222

1 Overview

This Artificial Neural Network (ANN) program is implemented in Ruby. It uses back-propagation algorithm discribed in textbook, but with stochastic update, to calculate the weights of edges in the feedforward network. The datasets used for testing are Credit Approval [1], Iris [2], Liver Disorders [3], Chronic Kidney Disease [4], and Teaching Assistant Evaluation [5].

2 Setup

2.1 Installation

There is no installation requirement since this program is developed in Ruby, However, please make sure the Ruby executable is at the path:

```
/usr/bin/ruby
```

2.2 Execution

Type the command below under the top directory:

```
./run_ann.rb <data_file> <attribute_file> <setup_file>
```

The format of "`<data_file>`", "`<attribute_file>`" and "`<setup_file>`" will be explained below.

2.2.1 Data File

A data file is a comma-separated file with one example per line, and the target attribute should be placed in the last column, with missing data specified as "`?`". Note that all the white space (ex: space, tab) will be removed before parsing.

```
Sunny,      85,     High,   Weak,   No
Rain,       57,     High,   Weak,   Yes
Rain,       64,     Normal, ?,      Yes
Rain,       ?,      Normal, Weak,   No
Sunny,      65,     Normal, Strong, Yes
Overcast,   33,     ?,      Strong, Yes
...
```

2.2.2 Attribute File

In attribute file, every line indicates an attirbute, and there should be exact 2 columns in a line: the first column contains the name of the attribute, and the second column specifies either this attribute is continuous (c) or discrete (d). Note that the total number of lines in the attribute file should be the same as the total number of columns in data file.

```
Outlook,        d
Temperature,    c
Humidity,       d
Wind,           d
PlayTennis,     d
```

2.2.3 Setup File

The setup file is a JSON file with key-value pairs, specifiying internal parameters of this ANN program. Parameters not specified or set as "default" will be set to default values as follows:

```
{
    "min_iteration"         : 100,
    "max_iteration"         : 10000,
    "update_ratio"          : 0.1,
    "hidden_layer"          : 1,
    "hidden_layer_width"    : <number of attributes>
    "momentum"              : 0.0
}
```

2.3 Project Structure and Evnironment

The development environment is as follows:

- `OS: OSX El Capitan 10.11.3`
- `Ruby: ruby 2.3.0p0 (2015-12-25 revision 53290)[x86_64-darwin15]`

The project structure is as follows:

- `run_ann.rb` : excutable
- `report.pdf` : report
- `makefile`: for test only
- `src/` : source codes
- `data/` : datasets for testing
- `test/` : results

3 Implementation

3.1 Pre-processing

The data, attribute, and setup file are parsed and pre-processed in "`class Tester`". All possible values for each discrete attribute will be extracted, and the mean and standard deviation of each continuous attribute will be calculated. Examples will be shuffled and then be partitioned into 10 parts for 10-fold cross-validation. In each iteration, 1 part of the sub-examples will be kept for testing, and the rest will be passed down to "`class ANN`" for ANN construction.

3.2 Learning

The ANN is built in "`class ANN`". All the examples used for learning will be shuffled again, and then 1/3 of them are put aside for validation; the rest 2/3 examples will be used for training.

Before training, all the examples are transformed to the corresponding inputs and outputs for ANN. The feedforward network will be set up based on the number of inputs and outputs, as well as the depth and width of hidden layered; every perceptrons in it use the standard sigmoid function as threshold, and all the weights of edges are initialized 0.0. This ANN is a fully connected network, which means that every output of the perceptrons of a given layer is connected to all the perceptrions of the next layer.

During training, this ANN uses backpropagation algorithm discribed in textbook with stochastic method to update the weights. After each iteration, it calculates the MSE of validation set, and keeps track of the weights and the iteration number with the minimal MSE. It will keep going until meeting the stop criteria, and then the weights will be rolled back to the one with the minimal MSE and used for future classification.

In classification, All the examples will be transform to the input of ANN as well, and the maximal value among the outputs of the ANN will be seen as the result.

3.2.1 Example Transformation

All the attributes will be mapped to some real numbers before training. For continuous attributes, the values will be normalized: being subtracted by the mean and then being divided by the standard deviation.

For discrete attributes, every possible value of a given attribute will be mapped to an input with value either 1.0 (if it's the same as current value in example) or 0.0 (otherwise). For example, if an attibute A with current value x has 3 possible values x, y, z, then it will be mapped to 3 attributes (A_x, A_y, A_z), with value (1.0, 0.0, 0.0). The mapping is in function `ANN#map_to_ann`

3.2.2 Missing Value

Examples with missing values are handled during the mapping. For continuous attributes, the missing value will be set to the mean of the attribute. For discrete attributes, the

value 1.0 will be evenly distributed through all inputs. For instance, if an attribute has 4 possible values, then each of them will be 0.25.

3.2.3 Stop Criteria

There are 2 stop criteria: the first one is a hard limit, which is specified as "`max_iteration`" in the setup file. The program will stop training whenever reachs this limit; the second one is soft limit, when the iteration number between "`min_iteration`" and "`max_iteration`", if the current iteration number is twice as large as the iteration number with the minimal MSE. the program will stop.

4 Experimental Result

The datasets used for testing are Credit Approval [1], Iris [2], Liver Disorders [3], Chronic Kidney Disease [4], and Teaching Assistant Evaluation [5], and the parameters are set as follows:

```
{
    "min_iteration"         : 100,
    "max_iteration"         : 10000,
    "update_ratio"          : 0.1,
    "hidden_layer"          : 1,
    "hidden_layer_width"    : <number of attributes>
    "momentum"              : 0.0
}
```

All the detail information, including the MSE for training set and validation set in each round, can be found in files "`./test/test_ann/<CASE>/<CASE>.out`". (The details of ID3 and majority classifier are in"`./test/test_id3/<CASE>/<CASE>.out`")

4.1 Comparison between ANN and ID3

Based on experimental results, there seems to be no significant difference between ID3 and ANN in terms of accuracy in these 5 datasets. The accuracy in the first 4 datasets are nearly identical between these 2 learning algorithm, and I think the problem of ANN in the 5th dataset is that there are too many edges and the network hasn't stablized enough in 100 iteration and was stuck in the local minimum.

name	classifier	leaf/it	accuracy	SE	95% CI
crx	ANN	31.0	85.7	3.5	(77.9, 93.4)
	ID3	2.0	85.5	4.0	(76.5, 94.5)
	majority	N/A	55.5	5.8	(42.5, 68.5)
iris	ANN	5446.2	95.3	3.1	(88.5, 100.0)
	ID3	3.0	96.0	4.4	(86.1, 100.0)
	majority	N/A	21.3	4.0	(12.4, 30.3)
liver	ANN	3381.7	67.6	6.6	(52.8, 82.3)
	ID3	38.8	67.0	7.4	(50.5, 83.5)
	majority	N/A	58.0	7.2	(41.9, 74.1)
kidney	ANN	2134.7	94.3	4.3	(84.6, 100.0)
	ID3	11.1	96.0	3.4	(88.4, 100.0)
	majority	N/A	62.5	6.8	(47.3, 77.7)
tae	ANN	52.6	34.5	11.9	(7.9, 61.0)
	ID3	241.8	52.4	10.4	(29.2, 75.7)
	majority	N/A	25.2	10.7	(1.4, 48.9)

Table 1: Results for ANN, ID3, and majority classifier

4.2 Hidden layer

There seems to be no difference in terms of width of hidden layer. I think it's because these 2 cases are relatively easy and not much concepts for addtional nodes to learn.

On the other hand, The accuracy decrease when the number of hidden layer increase in the iris dataset, I think it's because the setup of iteration is not enough for network with 2 hidden layer to converge.

layer	width	iteration	accuracy	SE	95% CI
0	0	729.2	93.3	7.3	(77.0, 100.0)
1	2	7341.7	96.7	3.3	(89.2, 100.0)
	4	5446.2	95.3	3.1	(88.5, 100.0)
	6	6169.1	96.0	3.3	(88.7, 100.0)
	8	5215.0	96.0	3.3	(88.7, 100.0)
2	2	4001.6	52.7	33.9	(0.0, 100.0)
	4	3101.1	52.7	33.9	(0.0, 100.0)
	6	4001.0	53.3	34.8	(0.0, 100.0)
	8	6074.7	74.7	31.7	(4.1, 100.0)

Table 2: Results for iris [2]

layer	width	iteration	accuracy	SE	95% CI
0	0	748.6	67.8	8.1	(49.8, 85.8)
1	2	3414.8	67.3	7.7	(50.1, 84.5)
	4	2657.8	67.6	7.4	(51.0, 84.1)
	6	3381.7	67.6	6.6	(52.8, 82.3)
	8	3514.9	67.8	6.2	(54.0, 81.7)
2	2	1338.2	66.1	6.7	(51.2, 80.9)
	4	1527.8	67.3	7.0	(51.6, 82.9)
	6	2105.2	67.0	7.2	(51.0, 82.9)
	8	2738.9	66.7	6.8	(51.5, 81.9)

Table 3: Results for liver [3]

References

[1] Credit Approval Data Set
http://archive.ics.uci.edu/ml/datasets/Credit+Approval

[2] Iris Data Set
http://archive.ics.uci.edu/ml/datasets/Iris

[3] Liver Disorders Data Set
http://archive.ics.uci.edu/ml/datasets/Liver+Disorders

[4] Chronic Kidney Disease Data Set
http://archive.ics.uci.edu/ml/datasets/Chronic_Kidney_Disease

[5] Teaching Assistant Evaluation Data Set
http://archive.ics.uci.edu/ml/datasets/Teaching+Assistant+Evaluation