Aprenentatge Automàtic 1

GCED

Lluís A. Belanche

belanche@cs.upc.edu

Soft Computing Research Group
Dept. de Ciències de la Computació (Computer Science)
Universitat Politècnica de Catalunya

2019-2020

LECTURE 8d: Artificial neural networks (I)

Artificial neural networks (I): the MLP

The MLP in R

```
nnet(x, y, weights, size, Wts, mask,
     linout = FALSE, entropy = FALSE, softmax = FALSE,
     censored = FALSE, skip = FALSE, rang = 0.7, decay = 0,
     maxit = 100, Hess = FALSE, trace = TRUE, MaxNWts = 1000,
     abstol = 1.0e-4, reltol = 1.0e-8, ...)
```

Suppose we are dealing with a three-class problem with 4 inputs:

```
nnet(x, y, size=2, softmax = TRUE)

a 4-2-3 network with 19 weights
```

Artificial neural networks (I): the MLP

Using exactly the *same* call to `nnet` three times, we have got these results:

```
# weights:  19
initial value 143.78535
iter  10 value 82.67353
iter  20 value 7.343014
iter  30 value 3.130778
iter  40 value 2.817692
iter  50 value 2.803824
iter  60 value 2.781383
iter  70 value 2.743949
iter  80 value 2.310192
iter  90 value 0.148647
iter 100 value 0.031570
iter 110 value 0.018951
...... <removed>
iter 860 value 0.000156
iter 870 value 0.000155
iter 880 value 0.000151
iter 890 value 0.000145
iter 900 value 0.000087
final  value 0.000087
converged
```

```
# weights:  19
initial value 147.09410
iter  10 value 70.01521
iter  20 value 34.64103
iter  30 value 21.04111
iter  40 value 6.781165
iter  50 value 1.729377
iter  60 value 1.051125
iter  70 value 0.725090
iter  80 value 0.318526
iter  90 value 0.136883
iter 100 value 0.116817
iter 110 value 0.111052
...... <removed>
iter1940 value 0.001235
iter1950 value 0.001235
iter1960 value 0.001230
iter1970 value 0.001225
iter1980 value 0.001212
iter1990 value 0.001206
iter2000 value 0.001198
final  value 0.001198
stopped after 2000 iterations
```

```
# weights:  19
initial  value 140.510964
iter  10 value 87.626321
iter  20 value 33.779907
iter  30 value 7.576811
iter  40 value 7.027161
...... <removed>
iter 100 value 6.448934
iter 110 value 6.448800
iter 120 value 6.448505
final  value 6.446125
converged
```

Artificial neural networks (I): the MLP

Tricks of the trade

Activation function An MLP usually trains faster if the activation function is anti-symmetric $g(-z) = -g(z)$ (e.g., the tanh)

Target values should not be the asymptotic values of the activation function, but leave a margin

Input values should be preprocessed so that:

1. their mean value is zero and their stdev is 1 (use standardization)
2. they are uncorrelated (use PCA)

Initial weights should be small and zero-centered, to avoid initially driving the neurons into saturation

Artificial neural networks (I): the MLP

Tricks of the trade

On-line vs. Batch

- **On-line training** is much faster –particularly in large redundant datasets– and often results in better solutions. It can also track changes in the data

- **Batch training** is better understood (convergence) and can be accelerated using second-order nformation; it is not sensitive to the ordering of examples

Today we call **mini-batches** (different) subsets of training observations used in consecutive epochs.

Artificial neural networks (I): the MLP

Number of hidden neurons

- While the number of inputs and outputs are dictated by the problem, the number of hidden neurons is not directly related to the application domain (it largely remains an unsolved problem)

  1. A small number may not be sufficient (**underfitted** model)

  2. A large number may be too much (**overfitted** model)

- Many methods abound: constructive (e.g., cascade correlation), pruning (e.g., optimal brain damage), golden search, ...

- Arguably the best method (it is more principled and arguably faster) is to use a large number and **regularize** the network

Artificial neural networks (I): the MLP

Number of hidden neurons

In order to find the best one-hidden-layer network architecture, there are two simple and common methods:

- Explore different numbers of hidden, with no regularization
- Fix a large number of hidden units, and explore different regularization values
- Both at the same time is not a sensible thing to do and usually a waste of computing resources