HOMEWORK FOR NEURAL NETWORK EXAM

Student: MASSIMO NOCENTINI
Professor: GREGORIO LANDI

October 10, 2014

Abstract

This short document collects the work I did in order to support my Neural Network course final exam. The goal is to study neural networks using different training functions in order to test them against a dataset which holds data about abalone. Here I describe my experience, giving an outline of the problem and its application environment, the structure of dataset used in experiments and a comparison of classification's results obtained applying different training functions.

1 PROBLEM DESCRIPTION

We choose a classification problem in order to gain insights about neural networks. In particular, the dataset under study contains informations about abalone and the contained data comes from real measurements. The original article related to this dataset can be found in [1] and the dataset itself can be downloaded from [2]. We report the original problem description given in the article:

> Predicting the age of abalone from physical measurements. The age of abalone is determined by cutting the shell through the cone, staining it, and counting the number of rings through a microscope – a boring and time-consuming task. Other measurements, which are easier to obtain, are used to predict the age. Further information, such as weather patterns and location (hence food availability) may be required to solve the problem.
>
> From the original data examples with missing values were removed (the majority having the predicted value missing), and the ranges of the continuous values have been scaled for use with an ANN (by dividing by 200).

So our goal is to classify abalone's age, in order to achieve that we tackle the dataset structure in the next section.

Comprehensive distribution over 2924 samples		Secret distribution over 1253 samples	
One	39 (1.33 %)	One	35 (2.79 %)
Two	1463 (50.03 %)	Two	559 (44.61 %)
Three	1240 (42.41 %)	Three	477 (38.07 %)
Four	158 (5.40 %)	Four	144 (11.49 %)
Five	21 (0.72 %)	Five	36 (2.87 %)
Six	3 (0.10 %)	Six	2 (0.16 %)

Table 1: On the left: histogram of samples used for training after aggregation. On the right: histogram of samples kept secret after aggregation.

2 DATASET STRUCTURE

The chosen dataset holds 4177 samples and 8 features variables and it is structured according the following description:

Name	Data Type	Meas.	Description
Sex	nominal		M, F, and I (infant)
Length	continuous	mm	Longest shell measurement
Diameter	continuous	mm	perpendicular to length
Height	continuous	mm	with meat in shell
Whole weight	continuous	grams	whole abalone
Shucked weight	continuous	grams	weight of meat
Viscera weight	continuous	grams	gut weight (after bleeding)
Shell weight	continuous	grams	after being dried
Rings	integer		+1.5 gives the age in years

With the following statistic summary:

	Length	Diam	Height	Whole	Shucked	Viscera	Shell	Rings
Min	0.075	0.055	0.000	0.002	0.001	0.001	0.002	1
Max	0.815	0.650	1.130	2.826	1.488	0.760	1.005	29
Mean	0.524	0.408	0.140	0.829	0.359	0.181	0.239	9.934
SD	0.120	0.099	0.042	0.490	0.222	0.110	0.139	3.224
Correl	0.557	0.575	0.557	0.540	0.421	0.504	0.628	1.0

Response variable *Rings* is the target of classification. In our work we do not use each age value $v \in \{1, \ldots, 29\}$ as classification bucket since having 29 classes is quite dispersive. Thus we aggregate the available *Rings* values in six classes, called *One*, *Two*, *Three*, *Four*, *Five* and *Six* for simplicity, each of length five, starting from zero, inclusive on the left. Formally, $\forall v \in Rings$ holds:

$$v \in One \leftrightarrow v \in \{0, \ldots, 4\}$$
$$v \in Two \leftrightarrow v \in \{5, \ldots, 9\}$$
$$v \in Three \leftrightarrow v \in \{10, \ldots, 14\}$$
$$v \in Four \leftrightarrow v \in \{15, \ldots, 19\}$$
$$v \in Five \leftrightarrow v \in \{20, \ldots, 24\}$$
$$v \in Six \leftrightarrow v \in \{25, \ldots, 29\}$$

In Table 1 we report the distribution of samples over the six defined classes: we take apart the dataset in two partitions, the bigger one is used for training, on the other hand the smaller one contains samples kept secret in order to study how well a training function works when it sees a new sample not trained for.

Dataset partitioning has been performed using the uniform distribution, and we repeat partitioning until distributions shown in Table 1 appeared, such that distribution of samples used for training is quite similar to distribution of samples kept secret.

	Backpropagation		RProp		SCG		LM	
All	One	0 (0.00 %)	One	14 (0.48 %)	One	0 (0.00 %)	One	32 (1.09 %)
	Two	2091 (71.51 %)	Two	1548 (52.94 %)	Two	1678 (57.39 %)	Two	1434 (49.04 %)
	Three	824 (28.18 %)	Three	1344 (45.96 %)	Three	1246 (42.61 %)	Three	1384 (47.33 %)
	Four	1 (0.03 %)	Four	18 (0.62 %)	Four	0 (0.00 %)	Four	74 (2.53 %)
	Five	3 (0.10 %)	Five	0 (0.00 %)	Five	0 (0.00 %)	Five	0 (0.00 %)
	Six	5 (0.17 %)	Six	0 (0.00 %)	Six	0 (0.00 %)	Six	0 (0.00 %)
Secrets	One	0 (0.00 %)	One	14 (1.12 %)	One	0 (0.00 %)	One	23 (1.84 %)
	Two	998 (79.65 %)	Two	669 (53.39 %)	Two	842 (67.20 %)	Two	596 (47.57 %)
	Three	241 (19.23 %)	Three	553 (44.13 %)	Three	411 (32.80 %)	Three	572 (45.65 %)
	Four	2 (0.16 %)	Four	17 (1.36 %)	Four	0 (0.00 %)	Four	62 (4.95 %)
	Five	4 (0.32 %)	Five	0 (0.00 %)	Five	0 (0.00 %)	Five	0 (0.00 %)
	Six	8 (0.64 %)	Six	0 (0.00 %)	Six	0 (0.00 %)	Six	0 (0.00 %)
One	One	0 (0.00 %)	One	11 (31.43 %)	One	0 (0.00 %)	One	19 (54.29 %)
	Two	30 (85.71 %)	Two	24 (68.57 %)	Two	35 (100.00 %)	Two	16 (45.71 %)
	Three	0 (0.00 %)	Three	0 (0.00 %)	Three	0 (0.00 %)	Three	0 (0.00 %)
	Four	0 (0.00 %)	Four	0 (0.00 %)	Four	0 (0.00 %)	Four	0 (0.00 %)
	Five	0 (0.00 %)	Five	0 (0.00 %)	Five	0 (0.00 %)	Five	0 (0.00 %)
	Six	5 (14.29 %)	Six	0 (0.00 %)	Six	0 (0.00 %)	Six	0 (0.00 %)
Two	One	0 (0.00 %)	One	3 (0.54 %)	One	0 (0.00 %)	One	4 (0.72 %)
	Two	509 (91.06 %)	Two	463 (82.83 %)	Two	481 (86.05 %)	Two	448 (80.14 %)
	Three	47 (8.41 %)	Three	93 (16.64 %)	Three	78 (13.95 %)	Three	107 (19.14 %)
	Four	0 (0.00 %)	Four	0 (0.00 %)	Four	0 (0.00 %)	Four	0 (0.00 %)
	Five	0 (0.00 %)	Five	0 (0.00 %)	Five	0 (0.00 %)	Five	0 (0.00 %)
	Six	3 (0.54 %)	Six	0 (0.00 %)	Six	0 (0.00 %)	Six	0 (0.00 %)
Three	One	0 (0.00 %)	One	0 (0.00 %)	One	0 (0.00 %)	One	0 (0.00 %)
	Two	348 (72.96 %)	Two	164 (34.38 %)	Two	262 (54.93 %)	Two	123 (25.79 %)
	Three	128 (26.83 %)	Three	308 (64.57 %)	Three	215 (45.07 %)	Three	330 (69.18 %)
	Four	0 (0.00 %)	Four	5 (1.05 %)	Four	0 (0.00 %)	Four	24 (5.03 %)
	Five	1 (0.21 %)	Five	0 (0.00 %)	Five	0 (0.00 %)	Five	0 (0.00 %)
	Six	0 (0.00 %)	Six	0 (0.00 %)	Six	0 (0.00 %)	Six	0 (0.00 %)
Four	One	0 (0.00 %)	One	0 (0.00 %)	One	0 (0.00 %)	One	0 (0.00 %)
	Two	92 (63.89 %)	Two	15 (10.42 %)	Two	53 (36.81 %)	Two	9 (6.25 %)
	Three	49 (34.03 %)	Three	121 (84.03 %)	Three	91 (63.19 %)	Three	112 (77.78 %)
	Four	1 (0.69 %)	Four	8 (5.56 %)	Four	0 (0.00 %)	Four	23 (15.97 %)
	Five	2 (1.39 %)	Five	0 (0.00 %)	Five	0 (0.00 %)	Five	0 (0.00 %)
	Six	0 (0.00 %)	Six	0 (0.00 %)	Six	0 (0.00 %)	Six	0 (0.00 %)
Five	One	0 (0.00 %)	One	0 (0.00 %)	One	0 (0.00 %)	One	0 (0.00 %)
	Two	18 (50.00 %)	Two	3 (8.33 %)	Two	11 (30.56 %)	Two	0 (0.00 %)
	Three	16 (44.44 %)	Three	29 (80.56 %)	Three	25 (69.44 %)	Three	21 (58.33 %)
	Four	1 (2.78 %)	Four	4 (11.11 %)	Four	0 (0.00 %)	Four	15 (41.67 %)
	Five	1 (2.78 %)	Five	0 (0.00 %)	Five	0 (0.00 %)	Five	0 (0.00 %)
	Six	0 (0.00 %)	Six	0 (0.00 %)	Six	0 (0.00 %)	Six	0 (0.00 %)
Six	One	0 (0.00 %)	One	0 (0.00 %)	One	0 (0.00 %)	One	0 (0.00 %)
	Two	1 (50.00 %)	Two	0 (0.00 %)	Two	0 (0.00 %)	Two	0 (0.00 %)
	Three	1 (50.00 %)	Three	2 (100.00 %)	Three	2 (100.00 %)	Three	2 (100.00 %)
	Four	0 (0.00 %)	Four	0 (0.00 %)	Four	0 (0.00 %)	Four	0 (0.00 %)
	Five	0 (0.00 %)	Five	0 (0.00 %)	Five	0 (0.00 %)	Five	0 (0.00 %)
	Six	0 (0.00 %)	Six	0 (0.00 %)	Six	0 (0.00 %)	Six	0 (0.00 %)

Table 2: Summary table respect different training functions

3 EXPERIMENTS METHODOLOGY

In this section we describe our methodology shortly. In order to ease our work we integrate the implementation of neural networks algorithms supplied by Matlab with a simple Smalltalk framework, implemented as a glue layer to process classification results generated by Matlab script.

We do not report the source code since it is a bit tedious and doesn't show anything of interest respect our goal: Matlab script just use the supplied training functions (change just one line of code for each different classification) while Smalltalk framework aggregate data, builds histograms and produces their TEXrepresentation (which we embed directly in this document). Both Matlab script, both Smalltalk framework are of public domain and described in section 6.

Those tools have been used as follow: in a first stage, we wrote AWK script to setup dataset for Matlab consumption. As a second step, for each training function of interest, we did a classification using a Matlab script, finally we aggregate results, mainly generating Table 2.

4 TRAINING NEURAL NETS

In this section we describe four attempts, each one of them is dedicated to a particular training function and in the corresponding subsection we report a plot of precision against epochs and a short commentary about obtained results. Before approaching each section, it is necessary to understand data reported in Table 2.

In Table 2 we report on columns the four training functions under study, while on rows subsets of samples, ie the set of samples used for training (called *All*), the set of samples kept secret (called *Secrets*) and, for each class c, a set of secret samples belonging to c. For each classification function f and samples set c, the pair (c, f) (ie, a cell) is the distribution of samples after a classification has been performed, training with f and relative to samples subset c. In other words, the distribution shown in each cell is computed comparing classification results with known targets. Three examples can be quite useful:

- pair $(All, RProp)$ report the distribution of samples produced by a classification using *ResilientPropagation* training function over the set of samples reserved for training. We can compare this distribution with the leftmost one reported in Table 1 to have a comparison about classification's behavior;
- pair $(Secrets, SCG)$ report the distribution of samples produced by a classification using *ScaledConjugateGradient* training function over the set of samples kept secret. In this case, we can compare this distribution with the rightmost one reported in Table 1;
- pair $(Three, LM)$ report the distribution of samples produced by a classification using $Levenberg - Marquardt$ training function over the set of samples kept secret. The distribution shows the "variance" of classified samples where we know that the sample belongs to class *Three*, in particular: take any secret sample s that we know from the dataset that $s \in Three$, hence the pair $(Three, LM)$ says that 69.18% of them are correctly classified in *Three*, while 25.79% are misclassified in *Two* and 5.03% are misclassified in *Four*.

For each experiment the structure of the network remains fixed: a feed-forward nets, using one hidden layer with 10 neurons, with sigmoid activations and training up to 1000 epochs.

In the next sections we comment each experiment looking at the described table distributions and plotting training functions' performances.

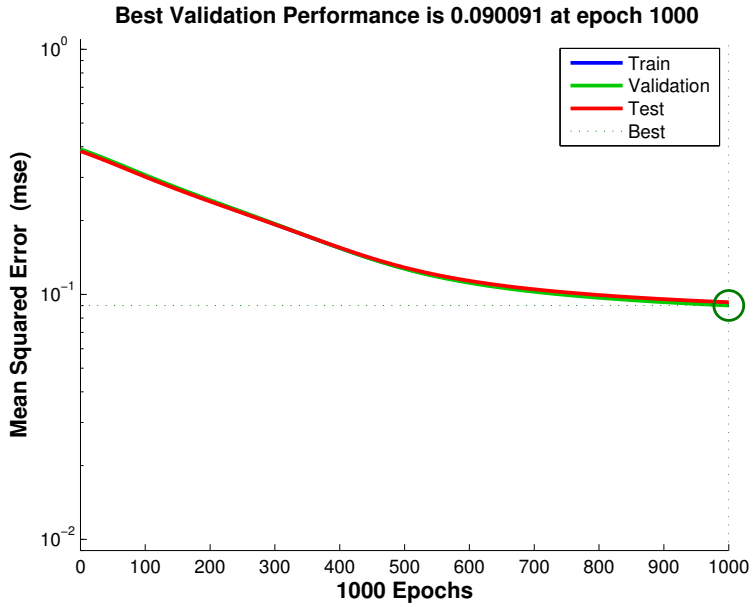

Figure 1: Performance against epochs for *Gradient Descent with Momentum* training function

4.1 *Gradient Descent with Momentum*

This is the first experiment we performed, just to have a "base case", in order to compare other powerful training functions. As we can see, both distributions of samples reserved for training and those kept secret are quite different from the real ones, only class *Two* is matched in some way, although overestimated.

Respect individual classes, the classification is wrong for all of them but class *Two*, where taken a secret sample $s \in Two$ from the dataset, s is correctly classified 91.06 times out of 100. It seems that this training function remains centered on the first classes since the great majority of samples are classified to belong to classes *Two* and *Three*.

In Figure 1 we report a plot of performance against epochs: as we can see the error slowly decreases, taking all the fixed 1000 epochs to complete the learning.

4.2 *Resilient Propagation*

With this training function it is possible to have some enhancements respect the previous one: as we can see from Table 2, classification produces a samples' distribution where classes *One* and *Four* get increased, while the last two classes take no sample at all.

Respect individual classes, taken a secret sample $s \in Two$ from the dataset, s is correctly classified 82.83 times out of 100. On the other hand, taken a secret sample $s \in Three$ from the dataset, s is correctly classified 64.57 times out of 100; classification does a mistake for all other classes.

In Figure 2 we report a plot of performance against epochs: as we can see the error decreases faster than the previous training function's one, taking 147 epochs to complete the learning.

4.3 *Scaled Conjugate Gradient*

This training function produced quite surprising results, since it classifies secret samples in two classes only, *Two* and *Three* respectively. Maybe this

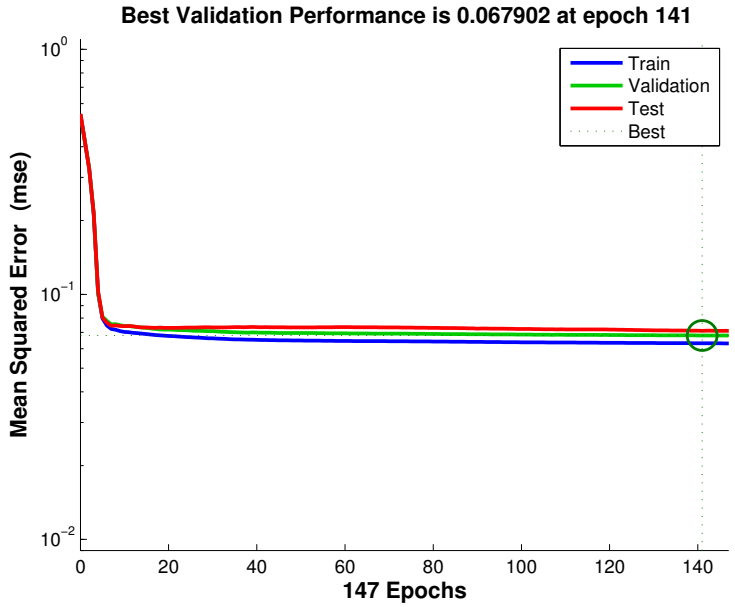

Figure 2: Performance against epochs for *Resilient Propagation* training function

strange behavior can be explained by the presence of some noise in the secret samples.

In Figure 3 we report a plot of performance against epochs: as we can see the error decreases in two main steps, remaining "constant" in each one of them, taking 30 epochs to complete the learning.

4.4 *Levenberg-Marquardt*

This training function is the one that produces a samples' distributions quite similar to the known ones, both for training samples both for samples kept secret.

Respect individual classes, taken a secret sample $s \in One$ from the dataset, s is correctly classified 54.29 times out of 100, while other trainings always behave worse. Taken a secret sample $s \in Two$ from the dataset, s is correctly classified 80.14 times out of 100. On the other hand, taken a secret sample $s \in Three$ from the dataset, s is correctly classified 69.18 times out of 100; classification does a mistake for all other classes.

In Figure 4 we report a plot of performance against epochs: as we can see the error decreases very fast, taking 27 epochs to complete the learning.

5 CLASSIFICATION RESULTS

In general, the classifications we've experimented with doesn't behave very well, both respect the training samples both those kept secret: the best training function is *Levenberg-Marquardt* for our purpose.

We could suggest that some improvements can be achieved if training consumes standardized data: we've tackled this approach writing a Matlab script and then using the results as inputs for new experiments, obtaining better classification of secret samples. We do not report those results here since it's only a sketch work, pointing the interested reader to repository described in section 6.

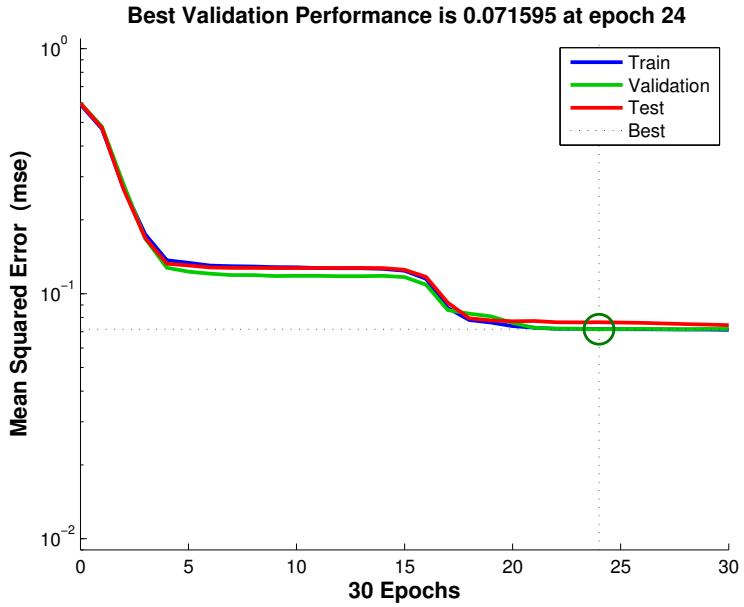

Figure 3: Performance against epochs for *Scaled Conjugate Gradient* training function

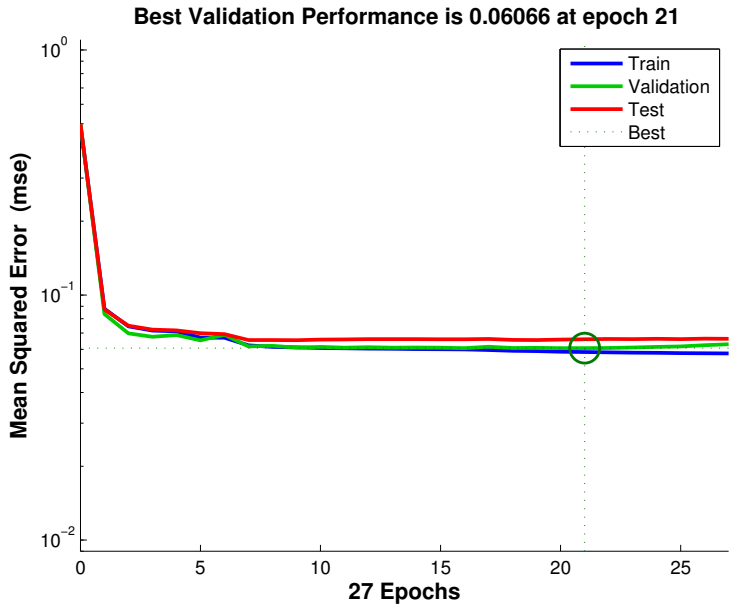

Figure 4: Performance against epochs for *Levenberg-Marquardt* training function

6 APPENDIX

6.1 *License*

```
The MIT License (MIT)

Copyright (c) 2014 Massimo Nocentini

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

6.2 *Project hosting*

All the content of this project has been versioned in a *Git* repository and it is of public domain:
`https://github.com/massimo-nocentini/neural-networks-exam`.

All code developed in *Pharo Smalltalk* is freely available at `http://smalltalkhub.com/#!/~MassimoNocentini/NeuralNetworksExam` and can be loaded directly via *Monticello* smalltalk browser with the following message:

```
MCHttpRepository
location: 'http://smalltalkhub.com/mc/MassimoNocentini/NeuralNetworksExam/main'
user: ''
password: ''
```

REFERENCES

[1] Warwick J Nash, Tracy L Sellers, Simon R Talbot, Andrew J Cawthorn and Wes B Ford (1994) "The Population Biology of Abalone (Haliotis species) in Tasmania. I. Blacklip Abalone (H. rubra) from the North Coast and Islands of Bass Strait", Sea Fisheries Division, Technical Report No. 48 (ISSN 1034-3288)

[2] Abalone dataset, `http://archive.ics.uci.edu/ml/machine-learning-databases/abalone/abalone.data`.