Learning with Regularization Networks in Bang [*]

Petra Kudová

Institute of Computer Science
Academy of Sciences of the Czech Republic
Pod Vodárenskou věží 2, 182 07 Praha 8, Czech Republic
petra@cs.cas.cz

Abstract

In this paper we study learning with Regularization Networks (RN). RN are feedforward neural networks with one hidden layer. Since they have a very good theoretical background, we study their practical aspects and applicability. On experiments we demonstrate the role of the regularization parameter, compare RN with different kernels and parameter settings on benchmark data sets. Then we apply RN to a problem of a flow rate prediction, real data from Czech river Sázava are used. All experiments were made using the system Bang.

1 Introduction

The problem of *learning from examples* (also called *supervised learning*) is a subject of great interest. The need for a good supervised learning technique stems from a wide range of application areas, covering various approximation, classification, and prediction tasks.

In this work we study one learning approach, Regularization Network, a feedforward neural network with one hidden layer, derived from the regularization theory. While it is well studied from the mathematical point of view ([1, 2]), we are more interested in practical points of its application.

For our experiments we used the system Bang [4]. The system Bang was designed for experiments with AI methods and easy creation of hybrid models. It implements several learning algorithms, such as neural networks, decision trees, GA. We use it as an environment suitable for experimental study of learning algorithms. For such experimental study, extensive evaluations on various tasks and with various setup are essential. Since the individual evaluations are possible to be run in parallel, the access to HPC-resources was very beneficial in this respect.

In the next section we will briefly describe the RN learning technique. In section 3 we introduce framework that we are using to estimate extra parameters of the RN learning algorithm. Section 4 demon-

[*]This work was carried out under the HPC-EUROPA project (RII3-CT-2003-506079), with the support of the European Community - Research Infrastructure Action under the FP6 "Structuring the European Research Area" Programme. The author is also partially supported by the Program "Information Society" under project 1ET100300419, by the Institutional Research Plan AV0Z10300504 "Computer Science for the Information Society: Models, Algorithms, Appplications".

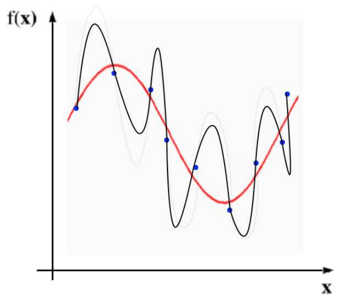

Figure 1: The problem of learning from examples

strates the behaviour of RN on some experiments.

2 Regularization Networks

Our problem can be formulated as follows. We are given a set of examples (pairs) $\{(\vec{x}_i, y_i) \in R^d \times R\}_{i=1}^N$ that was obtained by random sampling of some real function f, generally in the presence of noise (see 1). To this set we refer as *a training set*. Our goal is to recover the function f from data, or to find the best estimate of it. It is not necessary that the function exactly interpolates all the given data points, but we need a function with a good *generalization*, that is a function that gives relevant outputs also for the data not included in the training set.

This problem is generally ill-posed, there are many functions interpolating the given data points, but not all of them exhibit also the generalization ability.

Therefore we have to consider some a priori knowledge about the function. It is usually assumed that the function is *smooth*, in the sense that two similar inputs corresponds to two similar outputs and the function does not oscillate too much. This is the main idea of the regularization theory, where the solution is found by minimising the functional (1) containing both the data term and the smoothness information.

$$H[f] = \frac{1}{N}\sum_{i=1}^{N}(f(\vec{x}_i) - y_i)^2 + \gamma\Phi[f], \quad (1)$$

where Φ is called a *stabiliser* and $\gamma > 0$ is *the regularization parameter* controlling the trade-off between the closeness to data and the smoothness of the solution.

Poggio and Smale in [1] studied the Regularization Networks derived using a Reproducing Kernel Hilbert Space (RKHS) as the hypothesis space. Let $\mathcal{H}_K$ be an RKHS defined by a symmetric, positive-definite kernel function $K_{\vec{x}}(\vec{x}') = K(\vec{x}, \vec{x}')$. Then if we define the stabiliser by means of norm $||\cdot||_K$ in $\mathcal{H}_K$ and minimise the functional

$$\min_{f \in \mathcal{H}_K} H[f], \quad (2)$$

where

$$H[f] = \sum_{i=1}^{N}(y_i - f(\vec{x}_i))^2 + \gamma||f||_K^2 \quad (3)$$

over the hypothesis space $\mathcal{H}_K$, the solution of minimisation (3) is unique and has the form

$$f(\vec{x}) = \sum_{i=1}^{N} w_i K_{\vec{x}_i}(\vec{x}) \quad (4)$$

$$(\gamma I + K)\vec{c} = \vec{y}, \quad (5)$$

where I is the identity matrix, K is the matrix $K_{i,j} = K(\vec{x}_i, \vec{x}_j)$, and $\vec{y} = (y_1, \ldots, y_N)$.

The solution (4) can be represented by a feed-forward neural network with one hidden layer and a linear ouput layer (see (2)). We refer to a network of this form as *Regularization Network*.

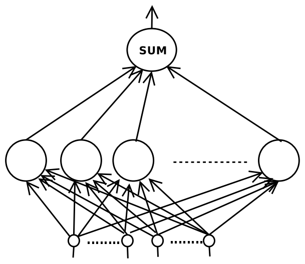

Figure 2: Regularization Network Scheme

3 Model Selection

As we will show in the next section, the choice of the explicit parameters (γ and the type of kernel) is crucial for the successful application of the RN learning algorithm. There exists no easy way for estimation of these parameters, usually some kind of an exhaustive search for parameters with the lowest cross-validation error is used.

First we need a measure of a real performance of the network, that enables us to say that one particular choice of parameters is better than another. We use k-fold cross-validation to estimate the real performance of the network and its generalization ability.

Suppose we are given a training set of data $TS = \{\vec{x}^i, y^i\}_{i=1}^N \subseteq \Re^n \times \Re$. We split this data set randomly into k folds $TS_1, \ldots, TS_k$ such as $\bigcup_{i=1}^k TS_i = TS$ and $TS_i \bigcap_{i \neq j} TS_j \neq 0$. Then f_i is the network obtained by the algorithm run on the data set $\bigcup_{j \neq i} TS_j$. The cross-validation error is given by

$$E_{cross} = \frac{1}{k} \sum_{i=1}^{k} \frac{1}{|TS_i|} \sum_{(\vec{x},y) \in TS_i} (f_i(\vec{x}) - y)^2. \tag{6}$$

In our experiments we usually use Gaussian kernels, so the choice of a kernel type is reduced to the choice of Gaussian width. We use the *Adaptive grid search* (Algorithm 3) for estimation of the regularization parameter and the width of Gaussian kernels, starting with a coarse grid and then creating a finer grid around the point with the lowest cross-validation error.

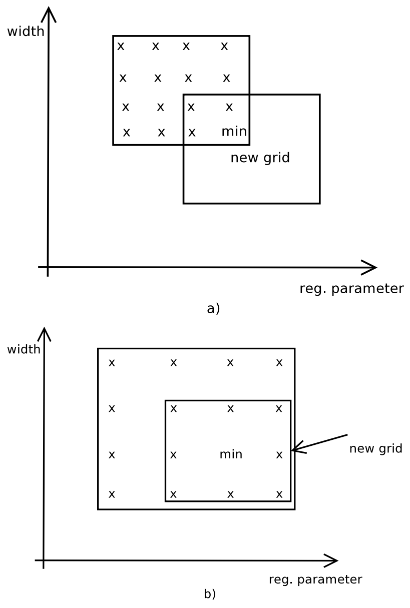

Figure 4: a) Move grid b) Create finer grid

4 Experiments

This section presents some results of our experiments, including both benchmark and real life problems. Implementation in the system Bang [4] is used. All experiments were run on Lomond [5].

In all experiments we use different data sets for training and testing (called *training set* and *testing set*). Kernel parameters and the regularization parameter are chosen by the method described in 3 us-

```
Input: Data set $\{\vec{x}^i, y^i\}_{i=1}^N \subseteq \Re^n \times \Re$
Output: Parameters $\gamma$ and a width $b$.

1. Create a set of couples $\{[\gamma, b]_i, i = 1, \ldots, K\}$, uniformly distributed in
   $< \gamma_{min}, \gamma_{max} > \times < b_{min}, b_{max} >$.
2. For each $[\gamma, b]_i$ for $i = 1, \ldots, K$ and for each couple evaluate the cross-
   validation error (6) $E^i_{cross}$.
3. Select the $i$ with the lowest $E^i_{cross}$.
4. If the couple $[\gamma, b]_i$ is at the border of the grid, move the grid. (see
   Fig. 4a)
5. If the couple $[\gamma, b]_i$ is inside the grid, create finer grid around this couple.
   (see Fig. 4b)
6. Go to 2 and iterate until cross-validation error stops decreasing.
```

Figure 3: Adaptive grid search

ing the training set. Then RN learning is run on the training set and the error of the resulting network is evaluated on the testing set as:

$$E = 100 \frac{1}{N} \sum_{i=1}^{N} ||\vec{y}^i - f(\vec{x}^i)||^2,$$

where N is the number of data samples, $\vec{y}^i$ is the desired output for the input vector $\vec{x}^i$, $f(\cdot)$ is the network output and $|| \cdot ||^2$ denotes the Euclidean norm.

First we demonstrate the behaviour of RN learning on benchmark data sets, then we show its application on the prediction of river flow rate.

Benchmark data sets

The data collection Proben1 [6] was selected to compare different types of kernels. Table 1 lists the most common kernel functions. The Tab. 2 compares the errors achieved with these kernels on some data tasks from Proben1.

We have chosen the well known picture of Lenna to study the approximation and smoothing capabilities of Regularization Networks. Our training set contains 2500 samples representing the image of 50×50 pixels. The obtained RN was then used to generate 100×100 image.

In Fig. 7 are images obtained with RNs using Gaussian kernels of different widths and different regularization parameters. It is easy to see that the choice of these parameters is crucial for the performance of RN algorithm. It also demonstrate the role of the regularization parameter, the higher the regularization parameter the smoother the result. For too high regularization parameters we get solutions too far from training data, i.e. the black images.

Fig. 6 shows the training images with different level of noise and the solutions found by RN.

Prediction of river flow rate

We study the possible application of neural networks on the prediction of river flow rate [3] in cooperation with Faculty of Environmental Studies, University of J. E.

Gaussian	$K(x,y) = e^{-\|\|x-y\|\|^2}$
Inverse Multi-quadratic	$K(x,y) = (\|\|x-y\|\| + c^2)^{-1/2}$
Multi-quadratic	$K(x,y) = (\|\|x-y\|\| + c^2)^{1/2}$
Thin Plate Spline	$K(x,y) = \|\|x-y\|\|^{2n+1}$
Sigmoid	$K(x,y) = tanh(xy - \theta)$

Table 1: Kernel functions

kernel	cancer1	cancer2	cancer3	glass1	glass2	glass3
Gaussian $d = 0.5$	3.4	4.6	6.5	**7.4**	**7.8**	**7.3**
Gaussian $d = 1.0$	1.6	3.0	2.8	8.2	8.6	8.2
Gaussian $d = 2.0$	1.8	3.1	2.9	8.6	8.9	8.9
Inverse Multi-quadratic $c = 1.0$	**1.5**	**2.8**	**2.5**	8.0	8.3	7.8
Inverse Multi-quadratic $c = 2.0$	1.6	2.9	2.8	8.6	9.0	9.0
Inverse Multi-quadratic $c = 3.0$	1.8	3.0	2.8	8.8	9.0	9.4
Multi-quadratic $c = 1.0$	30.6	53.7	28.2	13.9	13.8	14.4
Multi-quadratic $c = 2.0$	7.2	10.5	5.1	17.8	12.7	14.9
Multi-quadratic $c = 3.0$	7.9	9.0	6.0	12.6	26.6	13.3
Sigmoid $\theta = 0.0$	4.0	4.7	4.5	11.1	10.9	12.1
Sigmoid $\theta = 2.0$	2.7	3.9	5.1	11.5	11.3	12.4
Sigmoid $\theta = 3.0$	1.9	3.6	2.9	9.5	8.9	9.0
Thin Plate Spline $n = 1$	**1.5**	2.9	2.6	13.9	8.8	11.5
Thin Plate Spline $n = 2$	29.8	15.6	12.1	2247.5	18.9	121.0

Table 2: Error on the testing set on some tasks from Proben1.

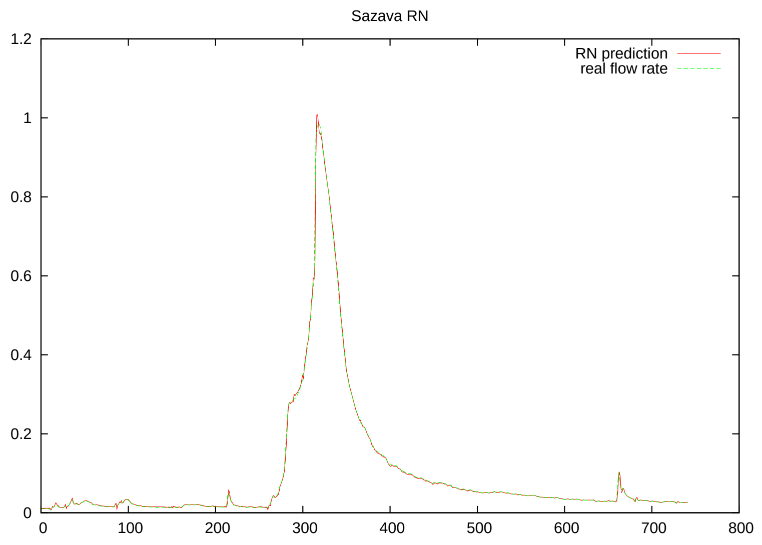

Figure 5: Prediction of flow rate on river Sázava by RN.

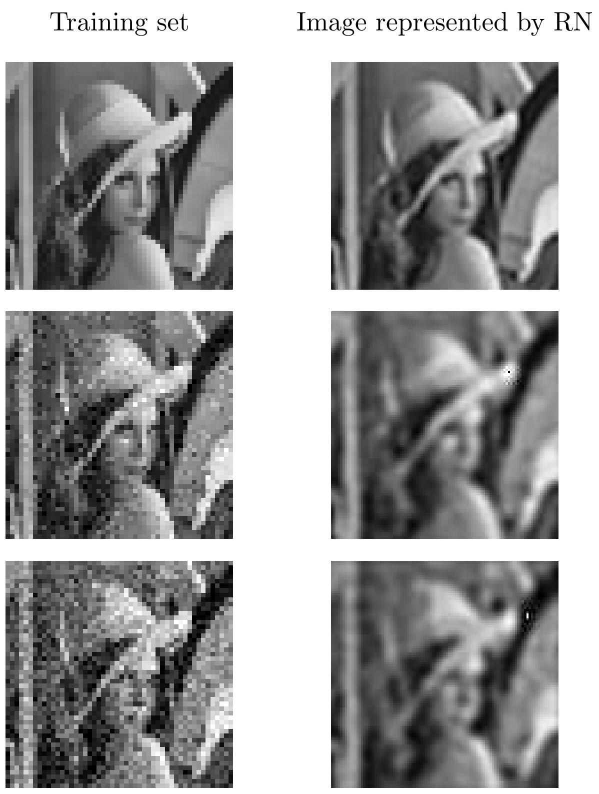

Figure 6: Approximation of Lenna image using RN. Original image of 50×50 pixels and 100×100 pixels image created using learned RN.

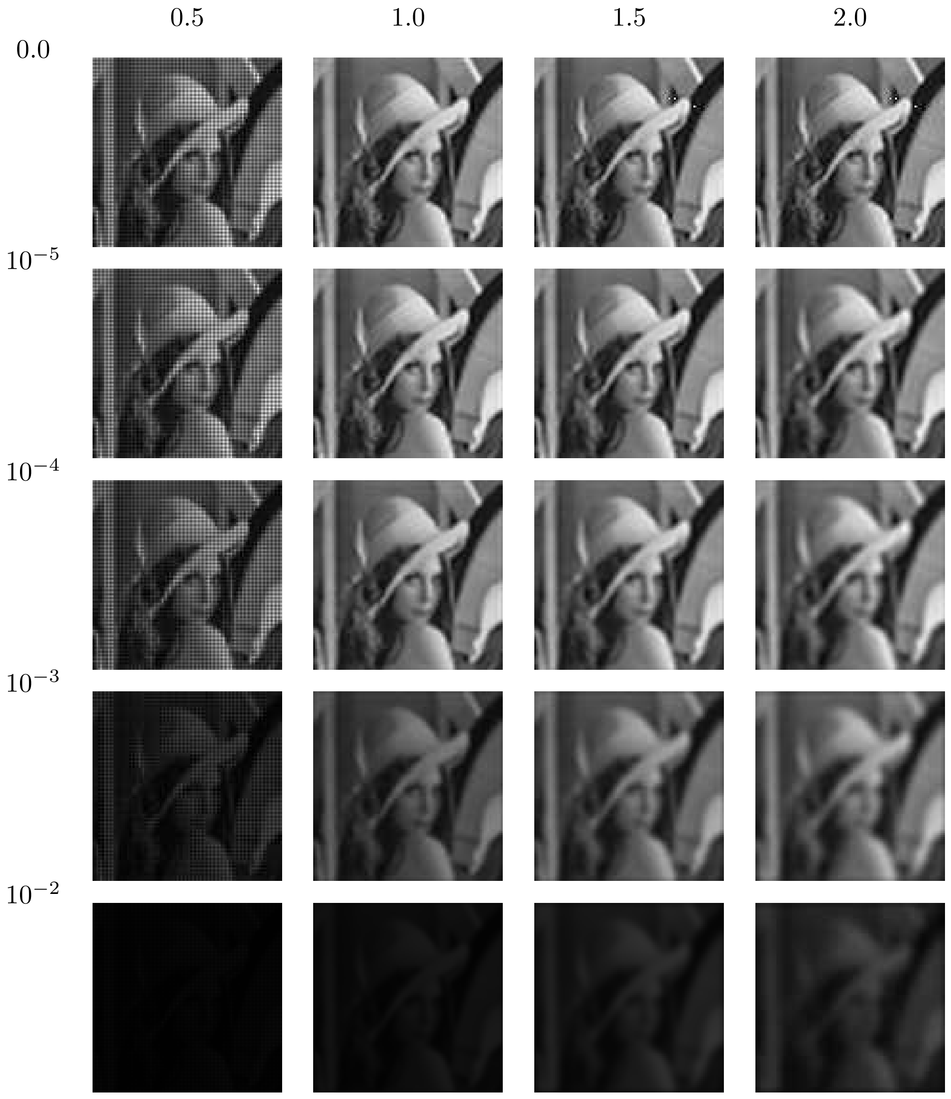

Figure 7: Images learned by RN on Lenna image (50×50 pixels) using Gaussian kernels with widths from 0.5 to 2.0 and regularization parameters from 0.0 to 0.01.

	E_{train}	E_{test}
1 back, 1 forecast	0.006	0.010
1 back, 2 forecast	0.024	0.027
2 back, 1 forecast	0.004	0.003
2 back, 2 forecast	0.017	0.030

Table 3: Errors achieved by RN on the prediction of flow rate on river Sázava.

Purkyně in Ústí nad Labem.

Data from the Czech river Sázava were used in this experiment. Data set was obtained in period from 1. 8. to 11. 9. 2002. It is sampled every hour and contains flood values from 14. 8. 2002, from 7:00 to 9:00 ($169 m^3 s^{-1}$).

Each data sample contains the present river flow rate, total rainfall and total rainfall for the moment of prediction (rain fall forecast is used in real application). The Tab. 3 lists error obtained by RN for different types of forecast. We predict one or two future samples based on one or two samples from history. The prediction is better if the network is given more information (2 samples) and if it has to predict only one sample (easier task).

An example of prediction of flow rate by RN is shown in Fig. 5.

5 Acknowledgement

All experiments presented in this paper were made during my visit at EPPC in November 2005. It would not be possible without an assistance of the EPCC team. Namely I would like to thank my host Prof. Ben Paechter and my contact Adam Carter for their scientific and technical support.

References

[1] T. Poggio, S. Smale (2003) The Mathematics of Learning: Dealing with Data, Notices of the AMS, 50/5, pp. 536–544.

[2] F. Girosi, M. Jones, T. Poggio (1995) Regularization Theory and Neural Networks Architectures, Neural Computation, 2/7, pp. 219–269.

[3] M. Neruda, R. Neruda, P. Kudová, K. Fiedlerová (2004) Rainfall-runoff modelling with Artificial Neural Networks and GIS tools. In D. Tropeano, M. Arattano, F. Maraga, C. Pelissero (eds.): Progress in surface and subsurface water studies at the plot and small basin scale. Italian National Research Council, Turin, 2004, pp. 101104.

[4] R. Neruda, P. Krušina, P. Kudová, P. Rydvan, G. Beuster (2004) Bang 3: A Computational Multi-Agent System. In Intelligent Agent Technology. Piscataway, 2004, pp. 563-564.

[5] Lomond machine. `http://www.epcc.ed.ac.uk/computing/services/sun/documents/hpc-intro/html/index.html`

[6] L. Prechelt (1994) PROBEN1 – A Set of Benchmarks and Benchmarking Rules for Neural Network Training Algorithms, Universitaet Karlsruhe, 21/94.