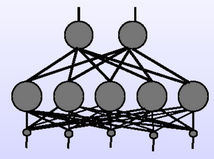
LEARNING IN RADIAL BASIS FUNCTION NETWORKS AND REGULARIZATION NETWORKS

Kudová P. and Neruda R. (petra@cs.cas.cz,roman@cs.cas.cz),
Institute of Computer Science, Academy of Sciences of the Czech Republic



Abstract

We discuss two approaches to supervised learning, namely regularization networks and RBF networks, and demonstrate their performance on experiments. We claim that the performance of these two models is comparable, so the RBF networks can be used as a cheaper alternative to regularization networks.

Motivation

The problem of *learning from examples* is a subject of great interest. It can be formulated as follows. We are given a set of examples $\{(\vec{x}_i, y_i) \in R^d \times R\}_{i=1}^m$ that was obtained by random sampling of some real function f, generally in the presence of noise. Our goal is to recover the function f from data, or find the best estimate of it, with respect to generalisation.
Artificial neural networks are typically used in such situations. There is a good supply of network architectures and corresponding supervised learning algorithms. Moreover the problem has been thoroughly studied as a function approximation problem. Since it is ill-posed, regularisation techniques are used.

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

FIGURE 1: Regularization scheme. Φ is a *stabilizer*, $\gamma > 0$ *the regularization parameter.*

Regularization Networks

Poggio and Smale in [PS03] proposed a learning algorithm (Algorithm 1) derived from the regularization scheme (Fig. 1). They choose the hypothesis space as a Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}_K$ defined by an explicitly chosen, symmetric, positive-definite function $K_{\vec{x}}(\vec{x}') = K(\vec{x}, \vec{x}')$. As a stabilizer the norm of the function in $\mathcal{H}_K$ is taken. They proved that the problem has an unique solution.
The power of the Algorithm 1 is in its simplicity and effectiveness. It's drawbacks are the high model complexity and explicit parameters (γ, for Gaussian kernels also width d). We use cross-validation to estimate these parameters.

Input: Data set $\{\vec{x}_i, y_i\}_{i=1}^m \subseteq X \times Y$ **Output:** Function f.

1. Choose a symmetric, positive-definite function $K_{\vec{x}}(\vec{x}')$, continuous on $X \times X$.
2. Create $f : X \to Y$ as $f(\vec{x}) = \sum_{i=1}^m c_i K_{\vec{x}_i}(\vec{x})$ and compute $\vec{c} = (c_1, \ldots, c_m)$ by solving

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

where I is the identity matrix, K is the matrix $K_{i,j} = K(\vec{x}_i, \vec{x}_j)$, and $\vec{y} = (y_1, \ldots, y_m)$, $\gamma > 0$ is real number.

Algorithm 1

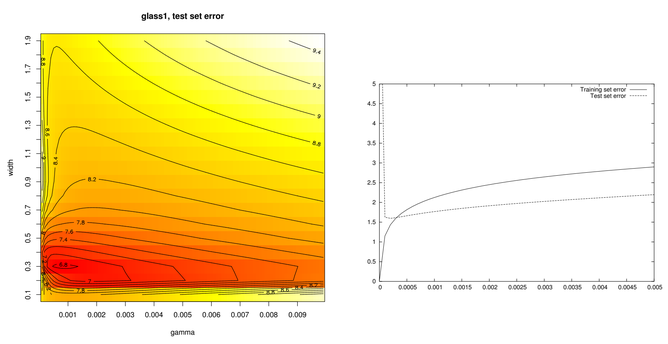
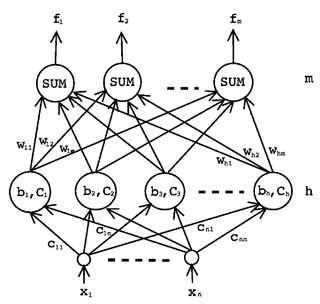


FIGURE 2: a) The dependence of the error function (computed on test set) on parameters γ a d. b) The relation between γ and training and testing error.

RBF Neural Networks



$$y(\vec{x}) = \varphi\left(\frac{\| \vec{x} - \vec{c} \|_C}{b}\right)$$

$$f_s(\vec{x}) = \sum_{j=1}^{h} w_{js}\varphi\left(\frac{\| \vec{x} - \vec{c}_j \|_{C_j}}{b_j}\right)$$

FIGURE 3: a) RBF network architecture b) RBF network function

An RBF network represents a relatively new model of NN, as a network with local units. We are using a general model with weighted norms (see Fig. 3). The two most significant algorithms, *Three step learning* and *Gradient learning*, are sketched in Algorithm 2 and Algorithm 3. See also [NK04].

Input: Data set $\{\vec{x}_i, \vec{y}_i\}_{i=1}^N$ **Output:** $\{\vec{c}_i, b_i, C_i, w_{ij}\}_{i=1..h}^{j=1..m}$

1. Set the centers $\vec{c}_i$ by a vector quantization algorithm.
2. Set the widths b_i and matrices C_i by unsupervised optimization.
3. Set the weights w_{ij} by solving $\Phi W = D$.

$$D_{ij} = \sum_{t=1}^{N} \vec{y}_{tj} e^{-\left(\frac{\|\vec{x}_t - c_i\|_{C_i}}{b_i}\right)^2}, \Phi_{qr} = \sum_{t=1}^{N} e^{-\left(\frac{\|\vec{x}_t - c_q\|_{C_q}}{b_q}\right)^2} e^{-\left(\frac{\|\vec{x}_t - c_r\|_{C_r}}{b_r}\right)^2}$$

Algorithm 2

Input: Data set $\{\vec{x}_i, \vec{y}_i\}_{i=1}^N$ **Output:** $\{\vec{c}_i, b_i, C_i, w_{ij}\}_{i=1..h}^{j=1..m}$

1. Put a part (10%) of data aside as an evaluation set ES, keep the rest as a training set TS .
2. $\forall j$ $\vec{c}_j(i) \leftarrow$ random sample from TS_1, $\forall j$ $b_j(i), \Sigma_j^{-1}(i) \leftarrow$ small random value, $i \leftarrow 0$
3. $\forall j, p(i)$ in $\vec{c}_j(i), b_j(i), \Sigma_j^{-1}(i)$: $\Delta p(i) \leftarrow -\epsilon\frac{\delta E_1}{\delta p} + \alpha\Delta p(i-1)$, $p(i) \leftarrow p(i) + \Delta p(i)$
4. $E_1 \leftarrow \sum_{\vec{x}\in TS_1}(f(\vec{x}) - y_i)^2$, $E_2 \leftarrow \sum_{\vec{x}\in TS_2}(f(\vec{x}) - y_i)^2$
5. If E_1 and E_2 are decreasing, $i \leftarrow i + 1$, go to 3, else STOP. If E_2 started to increase, STOP.

Algorithm 3

Experiments

Described algorithms were tested on chosen tasks from Proben1 [Pre94] data repository and on the prediction of flow rate on the river Ploucnice. In all experiments we use the normalized error $E_{ts} = 100\frac{1}{N}\sum_{i=1}^{N} ||\vec{y}_i - f(\vec{x}_i)||^2$.

	RN			RBF			MLP		
	E_{ts}	d	γ	E_{ts}	std	arch	E_{ts}	std	arch
cancer1	**1.60**	1.0	0.0002	**1.64**	0.16	20	**1.60**	0.41	4+2
cancer2	**2.99**	1.4	0.0002	**2.89**	0.07	20	**3.40**	0.33	8+4
cancer3	**2.76**	1.3	0.0005	**2.74**	0.20	20	**2.57**	0.24	16+8
cancer	**2.45**			**2.42**			**2.52**		
glass1	**6.75**	0.3	0.0008	**6.59**	0.32	15	**9.75**	0.41	16+8
glass2	**7.28**	0.3	0.0014	**7.85**	0.43	15	**10.27**	0.40	16+8
glass3	**6.48**	0.2	0.0017	**6.95**	0.26	15	**10.91**	0.48	16+8
glass	**6.84**			**7.13**			**10.31**		
hearta1	**4.44**	1.9	0.0008	**4.84**	0.25	30	**4.76**	1.14	32+0
hearta2	**4.32**	1.9	0.0012	**4.66**	0.08	30	**4.52**	1.10	16+0
hearta3	**4.45**	1.9	0.0008	**4.54**	0.06	30	**4.81**	0.87	32+0
hearta	**4.40**			**4.68**			**4.70**		

FIGURE 4: Comparison of Regularization Network, RBF network and multilayerperceptron.

Task	Type	n	m	Train. set size	Test set size
Cancer	Class.	9	2	525	174
Glass	Class.	9	6	161	54
Heart	Approx.	35	1	690	230

FIGURE 5: Overview of data sets from Proben1 database.

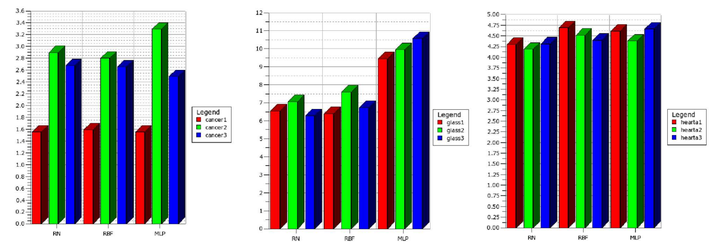


FIGURE 6: Comparison of RN, RBF, MLP: test set error.

	RN		RBF	
	time	units	time	units
cancer	9s (1 run of Alg1)	525	14s (100 iters of Alg3)	20
glass	1s (1 run of Alg1)	161	9s (100 iters of Alg3)	15
hearta	22s (1 run of Alg1)	690	74s (100 iters of Alg3)	30
ploucnice	55s (1 run of Alg1)	1000	28s (1 run of Alg2)	15

FIGURE 7: Time requirements, target processor was AMD Athlon(tm) XP 2100+.

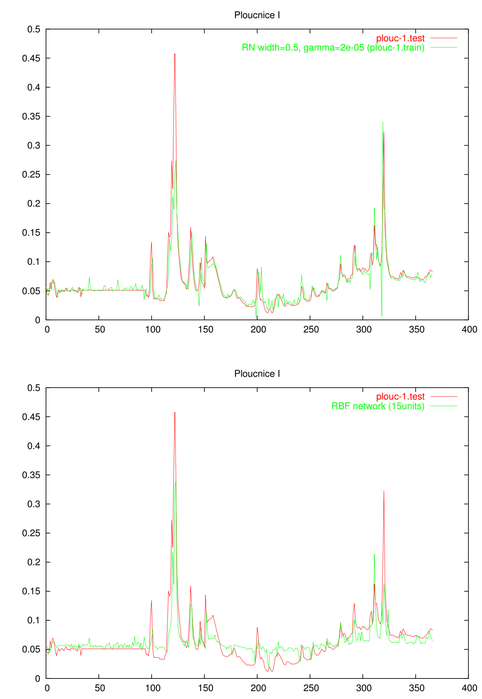


FIGURE 8: Prediction of the flow rate on the river Ploucnice: a) by RN b) by RBF

Conclusion

We showed that the performance of described algorithms are comparable and so the RBF networks can be used as an alternative to RN in applications where lower model complexity is desirable. On the Ploucnice task we demonstrated the applicability of both methods on real life problems.

References

[NK04] R. Neruda and P. Kudová. Learning methods for RBF neural networks. *Future Generations of Computer Systems*, 2004. In press.

[Pre94] L. Prechelt. Proben1 – a set of benchmarks and benchmarking rules for neural network training algorithms. Technical Report 21/94, Universitaet Karlsruhe, 1994.

[PS03] T. Poggio and S. Smale. The mathematics of learning: Dealing with data. *Notices of the AMS*, 50, No.5:537–544, 2003.

This research has been supported by the National Research Program Information Society project 1ET100300419.