Institute of Computer Science
Academy of Sciences of the Czech Republic

Learning of Generalized Regularization Networks

Petra Kudová

Technical report No. 851

December 2003

Pod Vodárenskou věží 2, 182 07 Prague 8, phone: (+420)26605 3630, fax: (+4202) 86 585 789,
e-mail:petra@cs.cas.cz

Learning of Generalized Regularization Networks [1]

Petra Kudová

Technical report No. 851

December 2003

Abstract:

In this paper we discuss regularization techniques and show that from regularization principles one can derive approximation schemes that are equivalent to feedforward neural networks with one hidden layer, which we will refer to as *regularization networks*. Several learning algorithms for regularization networks are discussed.

Keywords:
feedforward neural networks, regularization networks, aproximation theory.

[1]This work was supported by GA ČR grant No. 201/02/0428.

0.1. Introduction

Feedforward neural networks, as systems with the ability to learn autonomously a given task, can be very useful in many real life applications, namely prediction, classification, control, etc. In general such a network can be viewed as a black-box with several inputs and outputs. It is able to modify its parameters so as to fit the desired behaviour described by a set of examples, that is a set containing possible inputs together with corresponding outputs. This learning is known as *supervised learning* or *learning from examples.*

It turns out, that there is a close connection between the supervised learning of neural networks and the function approximation problem. While training a neural network, we are given a set of input/output pairs $\{\vec{x_i}, \vec{y_i}\}_{i=1}^m$ (called *training set*) and we are looking for the function that represents the relation between the inputs $\vec{x_i}$ and the outputs $\vec{y_i}$ the best. A function in some a priori given form is considered, corresponding to the type and architecture of the network as well as some other "reasonable" assumption about the approximation function (such as smoothness).

The aim of this paper is to show the connection between the function approximation theory and supervised learning of neural networks, show the derivation of regularization networks and derive algorithms suitable for learning of regularization networks. In the section 2 we will start with the summary of the regularization techniques ([7, ?])and show how different types of regularization schemes, corresponding to neural networks with one hidden layer, can be derived. In the section 3 we will move from theory to practial aspects and describe several algorithms that can be used for learning of derived networks.

0.2. The regularization approach

Given a set $\{(\vec{x}_i, y_i) \in R^d \times R\}_{i=1}^m$ of data that was obtained by random sampling of some function f (belonging to some space of functions X defined on R^d), generally in the presence of noise, our goal is to recover the function f from the data, or find the best estimate of it. One way is to minimize the error called *Empirical Risk*, i.e. find a function that minimizes the functional $H[f] = \sum_{i=1}^m (f(\vec{x}_i) - y_i)^2$ over a chosen function space, *hypothesis space* $\mathcal{H}$. Since this problem is generally ill-posed, an additional member called *stabilizer* or *regularization member* is added to $H[f]$. The stabilizer express some a priori knowledge about the function f, that is typically the *smoothness*, in the sense that two similar inputs correspond to two similar outputs. We get:

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

where $\Phi[f]$ is a stabilizer and $\gamma > 0$ is a *regularization parameter* that controls the trade-off between the smoothness and closeness to data.

Poggio, Smale [7] proposed an algorithm derived from Tikhonov regularization [8]. They choose a symmetric, positive-definite kernel function K and defined the stabilizer by means of the norm $||\cdot||_K$ in $\mathcal{H}_K$, that is the Reproducing Kernel Hilbert Space (RKHS) determined by the kernel K. So, a solution is found by minimizing the functional:

$$H[f] = \frac{1}{m} \sum_{i=1}^{m} (y_i - f(\vec{x}_i))^2 + \gamma ||f||_K^2, \tag{2}$$

over the hypothesis space $\mathcal{H}_K$. If we take the functional derivative with respect to f, apply it to $\bar{f} \in \mathcal{H}$, and set equal to 0:

$$\frac{1}{m} \sum_{i=1}^{m} (y_i - f(\vec{x}_i)) \bar{f}(\vec{x}_i) - \gamma \langle f, \bar{f} \rangle = 0 \tag{3}$$

where $\langle \cdot, \cdot \rangle$ is an inner product in RKHS. By setting $\bar{f} = K_{\vec{x}} = K(\cdot, \vec{x})$ we get

$$f(\vec{x}) = \sum_{i=1}^{m} c_i K_{\vec{x}_i}(\vec{x}) \qquad c_i = \frac{y_i - f(\vec{x}_i)}{m\gamma}. \tag{4}$$

Poggio, Girrosi and Jones in [2] proposed another form of a smoothness functional based on Fourier transform:

$$\Phi[f] = \int_{R^d} d\vec{s} \frac{|\tilde{f}(\vec{s})|^2}{\tilde{G}(\vec{s})}, \tag{5}$$

where $\tilde{f}$ indicates the Fourier transform of f , $\tilde{G}$ is some positive function that goes to zero for $||s|| \to \infty$ (i.e. $1/\tilde{G}$ is a high-pass filter).

Under slight assumptions on $\tilde{G}$ (as to be symmetric, so that its Fourier transform G is real and symmetric) it is possible to show that the solution minimizing the functional (1) has the form:

$$f(x) = \sum_{i=1}^{m} c_i G(\vec{x} - \vec{x}_i) + \sum_{\alpha=1}^{k} d_\alpha \psi_\alpha(\vec{x}), \tag{6}$$

where $\{\psi_\alpha\}_{\alpha=1}^k$ is a basis in the k-dimensional space $\mathcal{N}$ of the functional Φ (in most cases a set of polynomials). Coefficients d_α and c_i depend on the data and satisfy the following linear system:

$$(G + \gamma I)\vec{c} + \Psi^T \vec{d} = \vec{y} \tag{7}$$

$$\Phi \vec{c} = 0 \tag{8}$$

where I is the identity matrix, and we defined:

$$\vec{y} = (y_1, \dots, y_m),\ \vec{c} = (c_1, \dots, c_m),\ \vec{d} = (d_1, \dots, d_k) \tag{9}$$

$$(G)_{ij} = G(\vec{x}_i - \vec{x}_j),\ (\Psi)_{\alpha i} = \psi_\alpha(\vec{x}_i) \tag{10}$$

The existence of the solution to the linear system above is guaranteed by the existence of the solution of minimization (1).

The real form of the stabilizer (5) depends on the choice of $\tilde{G}$:

- **Radial stabilizers** – those are radially symmetric ones, i.e. satisfying $\Phi[f(\vec{x})] = \Phi[f(R\vec{x})]$, where R is an arbitrary rotation matrix. The radial symmetry reflects the assumption, that all the input variables have equal relevance, that is, there are no privileged directions. They lead to a *radial basis function* $G(||x||)$ and the approximation model corresponds to the RBF networks [3].

  The important example is the *Gaussian function* $\tilde{G}(\vec{s}) = e^{-\frac{||\vec{s}||^2}{\beta}}$ resulting in the basis function $G(\vec{x}) = e^{-\frac{||\vec{x}||^2}{\beta}}$, where β is a positive parameter. As the Gaussian function is a positive definite function, $\Phi[f]$ is a norm, its null space contains only the zero element, and therefore the additional terms of equation (6) are not needed.

- **Tensor product stabilizers** – an alternative choice for $\tilde{G}(\vec{s})$ in the form $\tilde{G}(\vec{s}) = \Pi_{j=1}^d \tilde{g}(s_j)$, where $\tilde{g}$ is an one-dimensional function. This leads to a tensor product basis function $G(\vec{x}) = \Pi_{j=1}^d g(x_j)$, where g is the Fourier transform of $\tilde{g}$. For positive definite functions g the functional $\Phi[f]$ is a norm and its null space is empty.

  An interesting example is the choice $\tilde{g}(s) = \frac{1}{1+s^2}$, which leads to the basis function:

  $$G(\vec{x}) = \Pi_{j=1}^d e^{-|x_j|} = e^{-\sum_{j=1}^d |x_j|} = e^{-||\vec{x}||_{L1}}. \tag{11}$$

  This basis function is interesting from the point of view of hardware implementation, since it requires only the computation of L_1 norm (instead the usual Euclidean norm L_2).

- **Additive stabilizers** – it is also possible to derive the class of *additive approximation schemes*, i.e. the schemes of the form $f(\vec{x}) = \sum_{\mu=1}^{d} f_\mu(x^\mu)$, where f_μ are one-dimensional functions. It can be done by the use of the stabilizer

$$\Phi[f] = \sum_{\mu=1}^{d} \frac{1}{\theta_\mu} \int_R ds \frac{|\tilde{f}_\mu(s)|^2}{\tilde{g}(s)}, \tag{12}$$

  where $\theta_\mu > 0$ are parameters allowing us to impose different degrees of smoothness on the different additive components. The minimum of this functional except the null space terms has again the form $f(\vec{x}) = \sum_{i=1}^{N} c_i G(\vec{x} - \vec{x}_i)$, where $G(\vec{x} - \vec{x}_i) = \sum_{\mu=1}^{d} \theta_\mu g(x^\mu - x_i^\mu)$. The additive components are not independent, because θ_μ are fixed.

0.3. Learning algorithms

In the previous section we derived several approximation schemes, that are all in the form

$$f(\vec{x}) = \sum_{i=1}^{m} c_i G(\vec{x}_i, \vec{p}_i, \vec{x}), \tag{13}$$

where m is the number of data points, $\vec{x}_i$ are the given data samples, G is some basis function, p_i are possible additional parameters of the basis function G and c_i are real coefficients, typically called *weights.* The scheme (13) is called *regularization network.* By the *generalized regularization network* we then understand the network, where the number of basis functions is lower than the number of data points.

Poggio, Girosi and Jones [7] proposed the special type of the generalized regularization network:

$$f(\vec{x}) = \sum_{\alpha=1}^{n} c_\alpha G(W\vec{x} - W\vec{t}_\alpha) \qquad n < m, \tag{14}$$

where W is matrix defining the transformation of the input space, t_α are heuristically chosen centers.

Now we introduce several algorithms dealing with the learning of generalized regularization networks. We start with the algorithm for the regularization network proposed by Poggio and Smale in [7]:

Algoritmus 0.3.1: Poggio, Smale

1. Start with data $\{\vec{x}_i, y_i\}_{i=1}^{m} \subseteq X \times Y$.
2. Choose a symmetric, positive-definite function $K_{\vec{x}}(\vec{x}') = K(\vec{x}, \vec{x}')$, continuous on $X \times X$.
3. Create $f : X \to Y$ by

$$f(\vec{x}) = \sum_{i=1}^{m} c_i K_{\vec{x}_i}(\vec{x}) \tag{15}$$

   and compute $\vec{c} = (c_1, \ldots, c_m)$ by solving

$$(m\gamma I + K)\vec{c} = \vec{y}, \tag{16}$$

   where I is the identity matrix, K is the square positive-definite matrix $K_{i,j} = K(\vec{x_i}, \vec{x_j})$, and $\vec{y} = (y_1, \ldots, y_m)$, $\gamma > 0$ is real number.

The linear system of equations (16) in m variables is well-posed, since K is positive-definite and $(m\gamma I + K)$ is strictly positive. The strength of this algorithm is in its simplicity. On the other hand, in real tasks, the data set can be really huge and the algorithms of type of 3.1, where the number of basis functions is the same as the number of data points, leads to models with high space complexity.

Therefore we are more interested in algorithms for generalized regularization networks. In our past work we have studied RBF neural networks in the form (17), that are in fact a special type of the generalized regularization network.

$$f(\vec{x}) = \sum_{j=1}^{h} w_j \varphi \left(\frac{\| \vec{x} - \vec{c}_j \|_{W_j}}{b_j} \right), \qquad h < m, \qquad \varphi \quad \text{some radial basis function} \tag{17}$$

Note that in our model each hidden unit j has possibly different norm matrix W_j. In [6] we described three main approaches to the RBF network learning *Three step learning*, *Gradient learning* and *Genetic learning* and also demonstrated them on several experiments.

The lower number of basis functions in generalized regularization networks is outweighted by the following obstacles: the learning problem is no more well-posed and more network parameters (such as centers $\vec{c}_j$, widths b_j and matrices W_j) have to be determined. We are again in a danger of *overfitting*, that means that our solution can go through all the data points (or most of them) and minimize the empirical risk $\sum_{i=1}^{N}(f(\vec{x}_i) - y_i)^2$ correctly, but oscillate too much and give a poor generalization results.

One way how to deal with this problem, is to add the regularization term to the error function similarly we did while minimizing the functional (1). (Note that now we are minimizing a function on the space of network parameters, while in section 2 and 5 we were minimizing a functional on some function space.)

Bishop in [1] proposed to measure the smoothness using the second derivatives of the approximating function:

$$E_{Bsh} = \sum_{i=1}^{m} (f(\vec{x}_i) - y_i)^2 + \gamma E_{reg}, \qquad E_{reg} = \sum_{i=1}^{m} \sum_{j=1}^{d} \left(\frac{\partial^2 f}{\partial x_j^2}(\vec{x}) \right)^2, \tag{18}$$

where x_j is the j-th coordinate of $\vec{x}$.

Our *Three step learning* adopted to the error function (18) leads to the algorithm:

Algoritmus 0.3.2: Three step algorithm based on Bishop's regularization

1. Set the centers $\vec{c}_j$ as random samples from the data set or find suitable representatives by a vector quantization.
2. Find the values for b_j a $\Sigma_j^{-1} = W_j^T W_j$ by minimizing the error function

$$E(b_1, \cdots, b_h; \Sigma_1^{-1}, \cdots, \Sigma_h^{-1}) = \frac{1}{2} \sum_{r=1}^{h} \left[\sum_{s=1}^{h} e^{-\left(\frac{\| c_s - c_r \|_{W_r}}{b_r} \right)^2} \left(\frac{\| c_s - c_r \|_{W_r}}{b_r} \right)^2 - P \right]^2, \tag{19}$$

   where P is an *overlap parameter.*

3. Find the values of w_j: Take derivative of (18) and put them equal to zero. Solve the linear system by pseudoinverse.

$$W = \Phi^+ D\ , \quad \Phi^+ = (\Phi^T \Phi)^{-1} \Phi \tag{20}$$

   where $D \in R^h$, $\Phi \in R^{h \times h}$ and

$$D_i = \sum_{t=1}^{m} d^{(t)} y_i(\vec{x}^{(t)}), \qquad y_j(\vec{x}^t) = e^{-\left(\frac{\| \vec{x}^t - c_j \|_{W_j}}{b_j} \right)^2} \tag{21}$$

$$\Phi_{qr} = \sum_{t=1}^{m} \left(y_q(\vec{x}^{(t)}) y_r(\vec{x}^{(t)}) + \lambda \sum_{i=1}^{d} \left(\frac{\partial^2 y_q^{(t)}}{\partial x_i^2} \frac{\partial^2 y_r^{(t)}}{\partial x_i^2} \right) \right). \tag{22}$$

This algorithm is similar to Algorithm 3.1 and is also quite simple. The trickier parts are the step 1 and 2, since they are in fact based on heuristics. The solving of linear system, though ill-posed, can be treated by available numerical methods. The main disadvantage of the algorithm, as well as in Algorithm 3.1, is the presence of parameter γ, that must be somehow estimated, and that may significantly influence the solution.

Our second approach – *Gradient learning* – is based on minimizing the error function by the gradient algorithm. In order to include regularization, one can simply replace the error function by the function (18), compute its derivatives and apply the gradient descent algorithm. In order to avoid the parameter γ , we will use the error function without the regularization member, but we will control the generalization ability of the network during learning, similary to cross-validation.

Algoritmus 0.3.3: Gradient learning with the test for generalization

1. Split the data set DS to *training set* TS_1 and *testing set* TS_2 (for instance 90% for training set, 10% for testing set, depending on the number of data points).
2. $i \leftarrow 0$
   $\forall j\ \vec{c}_j(i) \leftarrow$ random sample from TS_1
   $\forall j\ b_j(i), \Sigma_j^{-1}(i) \leftarrow$ small random value
3. For all j and $p(i)$ in $\vec{c}_j(i), b_j(i), \Sigma_j^{-1}(i)$:
$$\Delta p(i) \leftarrow -\epsilon\frac{\delta E_1}{\delta p} + \alpha\Delta p(i-1), \qquad p(i) \leftarrow p(i) + \Delta p(i) \tag{23}$$
$$E_1 \leftarrow \sum_{\vec{x}\in TS_1} (f(\vec{x}) - y_i)^2 \qquad \text{(error on the training set)} \tag{24}$$
4. $i \leftarrow i + 1$
5. $E_2 \leftarrow \sum_{\vec{x}\in TS_2}(f(\vec{x}) - y_i)^2$ (error on the testing set)
6. If both E_1 and E_2 are decreasing, go to 3. If E_2 started to increase, STOP.

The last algorithm we want to introduce here is the *Genetic learning.* It is the application of stochastic optimization technique known as *Genetic algorithms* on the minimization of the error function. Since it requires no computation of derivatives of the error function, but only an evaluation of it, we can use any form of the error function.

Algoritmus 0.3.4: Genetic learning with the regularization

1. Create random population P_0 of N feasible solutions. $i \leftarrow 0$
2. For all $I \in P_i$ compute:
$$cost(I) = \sum_{i=1}^{m}(f_I(\vec{x}_i) - y_i)^2 + \gamma E_{reg} \tag{25}$$
   where f_I is the function represented by individual I.
3. If the lowest cost in the population is sufficient STOP.
4. Create empty P_{i+1} and repeat until P_{i+1} is full:

   Selection: select 2 individuals I_1, I_2 (lower cost $\leftrightarrow$ higher probability).
   Crossover: $I_1', I_2' \leftarrow$ crossover(I_1, I_2).
   Mutation: mutate(I_1'), mutate(I_2').
   Add I_1', I_2' into P_{i+1}.
5. $i \leftarrow i + 1$, goto 2.

The regulatization member E_{reg} in the step 2 is either the Bishop's one from the equation (18) or the one based on Tikhonov regularization:

$$E_{reg} = \gamma \sum_{i=1}^{n} w_i^2 \tag{26}$$

More details can be found in [6] or [4].

0.4. Conclusion

We gave a summary of the use of regularization principles, showed how the regularization networks can be derived and how the different types of stabilizers lead to different types of regularization networks.

We proposed several algorithms for estimating the parameters of generalized regularization networks with radial basis functions. After slight adaptations, all of them can be used also for the other types of regularization networks from section 2.

We have tested these algorithms, without the regularization extension, for the network with the Gaussian basis function in our past work. The results of several experiments can be found in [6],[4] or [5].

Our future concern includes the study of the numerical properties of given algorithms (especially those based on solving linear systems), ways of estimation the regularization parameter γ as well as the number of basis function in GRN and tests of our algorithms, including the regularization extensions proposed in this paper and other models desribed in section 2.

References

[1] C.M. Bishop. Improving the generalization properties of Radial Basis Function neural networks. *NC*, 3:579–588, 1991.

[2] F. Girosi, M. Jones, and T. Poggio. Regularization theory and Neural Networks architectures. *Neural Computation*, 7,No.2:219–269, 1995.

[3] S. Haykin. *Neural Networks*. Macmillan College Publishing Company, New York, 1994.

[4] P. Kudová. Learning methods for RBF networks. Technical Report V-846, Institute of Computer Science, AS CR, Prague., 2001. p. 19-24.

[5] P. Kudová. Genetic and eugenic learning of RBF networks. Technical Report V-880, Institute of Computer Science, AS CR, Prague., 2002. p. 1-6.

[6] R. Neruda and P. Kudová. Hybrid learning of RBF networks. In P.M.A. Sloot, C.J.K. Tan, J.J. Dongarra, and A.G. Hoekstra, editors, *Computational Science*, pages 594–603. Berlin, Springer, 2002. Lecture Notes in Computer Science; 2331.

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

[8] A.N. Tikhonov and V.Y. Arsenin. *Solutions of Ill-posed Problems.* W.H. Winston, Washington, D.C, 1977.