Neural Processing Letters **65:** 97–104, 2002.

The Equivalence of Support Vector Machine and Regularization Neural Networks

PÉTER ANDRÁS
Department of Psychology, University of Newcastle upon Tyne, Newcastle upon Tyne, NE1 7RU, UK. e-mail: peter.andras@ncl.ac.uk

Abstract. We show in this brief paper the equivalence of the support vector machine and regularization neural networks. We prove both implication sides of the equivalence in a generally applicable way. The novelty lies in the effective construction of the regularization operator corresponding to a given support vector machine formulation. We give also a short introductory description of both neural network approximation frameworks.

Key words: approximation, equivalent neural networks, regularization, support vector machine

1. Introduction

Recent papers [2, 3, 5, 7, 8, 10, 11] show that the two top-level neural network approximation frameworks are the application of support vector machine theory and regularization theory to neural networks. Both frameworks fall into the general category of Bayesian neural network methods that are based on optimization of neural networks in the context of some prior distribution over the parameter space of neural networks [1, 13].

In previous works Smola et al. [10] and Girosi [5] have shown that the problem formulation of regularization neural networks can be transformed into a problem that corresponds to a support vector machine neural network. They have shown that the inverse is true in specific cases (e.g., Smola et al. [10] calculated the regularization operator for specific cases of kernels). However, they did not show in general that the inverse correspondence works, i.e., that the problem formulation of a general support vector machine neural network can be transformed into a problem that corresponds to a regularization neural network. We note that Smola et al. [10] mention this general correspondence as a conjecture in a footnote.

In this paper we prove in a compact and general manner that the two problem formulation, i.e., support vector machine and regularization formulation, are equivalent. The main novelty of the paper is that we show how to effectively construct the regularization operator corresponding to a given support vector machine problem.

The two problem formulations are presented briefly in Section 2. The theorems showing both implications of the equivalence relation are given in Section 3. The paper is closed by conclusions in Section 4.

2. Problem Formulation in the Two Approximation Frameworks

In this section we briefly present the problem formulation of neural network optimization in the case of support vector machine and regularization neural networks. Our goal is to approximate the functional relationship $f: \mathbb{R}^d \to \mathbb{R}$ that characterizes the data $D = \{(x^1, y_1), (x^2, y_2), \ldots, (x^n, y_n)\}$, where $x^i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$, and $i = 1, \ldots, m$.

2.1. THE SUPPORT VECTOR MACHINE NEURAL NETWORKS

The parameters of support vector machine neural networks are calculated by solving a support vector machine problem [12] in the context of some data. For general overview of the method see Scholkopf et al. [9] or Burges [2]. Here we present the problem formulation in the general case of approximation.

We construct the approximating function g as a combination of kernel functions corresponding to the so called support vectors. We set the maximum accepted error to be $\varepsilon > 0$. The support vectors are those x^i vectors that effectively determine the solution of the approximation problem, the other data vectors representing redundant information in the context of ε-precision approximation. The kernel function corresponds to the transformation of the data vectors into a high (possibly infinite) dimensional space where the original functional relationship is approximated by a linear function (for details see [2, 9]). In order to qualify as a kernel function, the function $K: \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ has to have a decomposition of form

$$K(x, y) = \langle \Phi(x), \Phi(y) \rangle \tag{2.1}$$

where $\Phi: \mathbb{R}^d \to H$ is a transformation of d-dimensional vectors into the vector space H, and $\langle \cdot, \cdot \rangle$ denotes the scalar product in H.

The support vector machine problem, which leads to the calculation of parameters of the corresponding neural network is formulated as follows:

$$\min_{\Theta} L_P \tag{2.2}$$

$$L_P = \sum_{i=1}^{n} \max(|y_i - g_\Theta(x^i)| - \varepsilon, 0) + \tfrac{1}{2}||\omega||^2 \tag{2.3}$$

where Θ represents the set of all parameters of the approximating function (i.e., the weights and parameters of the corresponding neural network) and $\omega \in H$ characterizes the approximating function.

In order to handle the high dimensional vector ω, we use the dual problem. By property (2.1) of kernel functions, the dual problem is:

$$\max_{\alpha, \alpha^* \in \mathbb{R}^n_+} L_D \tag{2.4}$$

$$L_D = \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)y_i - \varepsilon\sum_{i=1}^{n}(\alpha_i + \alpha_i^*) - \tfrac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(\alpha_i - \alpha_i^*)\cdot(\alpha_j - \alpha_j^*)K(x^i, x^j) \tag{2.5}$$

and the restrictions are

$$\sum_{i=1}^{n}(\alpha_i - \alpha_i^*) = 0 \tag{2.6}$$

$$\alpha_i, \alpha_i^* \in [0, C] \tag{2.7}$$

where $C > 0$ is a superior limit for the α-s. The support vectors are those x^i-s for which α_i or α_i^* is strictly positive (one of them is always zero).

The high dimensional hyperplane approximation of f is defined by ω and b, where

$$\omega = \sum_{i\in SV}(\alpha_i - \alpha_i^*)\Phi(x^i) \tag{2.8}$$

$$b = y_s - \langle\omega, \Phi(x^s)\rangle \tag{2.9}$$

$SV \subset \{1, 2, \ldots, n\}$ is the index set of the support vectors and x^s is one of the support vectors.

The corresponding approximating function is defined as:

$$g_\Theta(x) = \langle\omega, \Phi(x)\rangle + b = \sum_{i\in SV}(\alpha_i - \alpha_i^*)K(x, x^i) + b \tag{2.10}$$

The support vector neural network is composed by $m = card(SV)$ hidden neurons and an output neuron. The hidden neurons have as activation functions the half-fixed kernel functions $K(x, x^{i_j})$, where $i_j \in SV$ and $j = 1, \ldots, m$. The weights of the connections between the hidden neurons and the output neuron are $w_j = \alpha_{i_j} - \alpha_{i_j}^*$, $j = 1, m$. The output neuron calculates the network output by summing up the weighted outputs of the hidden neurons.

2.2. THE REGULARIZATION NEURAL NETWORKS

The regularization neural networks are based on the use of regularization theory in the context of neural network approximation. For an overview of the theory of regularization neural networks see Haykin [6] or Poggio and Girosi [8].

The approximating neural network has the mathematical representation

$$g(x) = \sum_{i=1}^{m} w_i g_i(x; \theta^i) \tag{2.11}$$

where w_i-s are the weights of the connections between the hidden neuron and the

output neuron of the network and θ^i-s are the parameter vectors of the activation functions of hidden neurons. Many times the classical squared error optimization problem is 'ill-posed' (i.e., has many similar quality solutions). In order to make it 'well-posed', we regularize it by applying a regularization operator P to the function represented by the neural network. The regularized version of the error functional is written as

$$E_R = \sum_{i=1}^{n} (y_i - g(x^i))^2 + \lambda ||Pg||^2 \tag{2.12}$$

The positive parameter λ controls the weight of the regularization restrictions.

The regularization operators determine the form of activation functions of hidden neurons. The regularization neural network corresponding to the data D and the regularization operator P has the mathematical representation [8]:

$$g(x) = \sum_{i=1}^{n} w_i G(x, x^i) \tag{2.13}$$

where the $G(x; x^i)$ functions are Green functions corresponding to the equations

$$P^* P(G(x, x^i)) = \delta_{x^i} \tag{2.14}$$

and the weights are determined from the matrix equation

$$w\Gamma = y \tag{2.15}$$

where $y = (y_1, \ldots, y_n)$, $w = (w_1, \ldots, w_n)$, and $\Gamma = [G(x^i, x^j)]_{i=1,n;j=1,n}$. (For a discussion on Green functions see [8].)

In other words this means that the obtained neural network has n hidden neurons and an output neuron. The hidden neurons have as activation functions the half-fixed Green functions $G(x, x^i)$. The weights w_i, of the connections between the hidden neurons and the output neuron are determined from the equation (2.15). The output neuron calculates the network output by summing up the weighted outputs of the hidden neurons.

3. The Equivalence of the Two Frameworks

In this section we prove both implication parts of the equivalence between the support vector machine neural networks and the regularization neural networks. First we show that for a given support vector machine neural network there is an equivalent regularization neural network. Second we show that the inverse of this is also true.

3.1. REFORMULATION OF SVM PROBLEMS AS REGULARIZATION PROBLEMS

We state and prove the theorem of the implication of existence of an equivalent regularization neural network for a given support vector machine neural network.

THEOREM 3.1. *Let* $K: \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ *be a kernel function. Then the support vector machine approximation problem for the data* $D = \{(x^1, y_1), (x^2, y_2), \ldots, (x^n, y_n)\}$, *where* $x^i \in \mathbb{R}^d$ *and* $y_i \in \mathbb{R}$, *is determined by the functional*

$$L_{P,\varepsilon} = \sum_{i=1}^{n} \max(|y_i - g_\Theta(x^i)| - \varepsilon, 0) + \tfrac{1}{2}||\omega||^2 \tag{3.1}$$

where ω *is defined by (2.8).*

If the support vector machine approximation problem has a solution, then there exists a regularization neural network that is the same as the one obtained by solving the specified support vector machine problem.

Proof. Assume that the kernel functions can be written as

$$K(z, t) = \langle \Phi(z), \Phi(t) \rangle \tag{3.2}$$

where $z, t \in \mathbb{R}^d$. Let $\mathcal{K}$ be the linear closure of the set of $K(x, x^i)$, $i = 1, \ldots, n$.

Consider a function g, and define the projection of g on $\mathcal{K}$ to be $pr_{\mathcal{K}}g$. Then we get that $pr_{\mathcal{K}}g$ has the form

$$pr_{\mathcal{K}}g(x) = \sum_{i=1}^{n} a_i K(x, x^i) \tag{3.3}$$

Define the operator P_K over the linear closure of the set of functions $K(z, t)$ as

$$P_K K(z, t) = \Phi(t) \tag{3.4}$$

$$P_K(a_1 K(z, t_1) + a_2 K(z, t_2)) = a_1 P_K K(z, t_1) + a_2 P_K K(z, t_2) \tag{3.5}$$

Define the operator P by extending the operator P_K to all functions by:

$$Pg = P_K(pr_{\mathcal{K}}g) \tag{3.6}$$

It is immediate to see that the operator P is well defined.

For the norm of Pg we get

$$||Pg||^2 = \langle Pg, Pg \rangle = \sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j K(x^i, x^j) \tag{3.7}$$

Writing the corresponding regularized approximation problem we get for the regularized error functional

$$E_R = \sum_{i=1}^{n} \max(|y_i - g_\Theta(x^i)| - \varepsilon, 0) + \tfrac{1}{2}||P(g_\Theta, \varepsilon)||^2 \tag{3.8}$$

By considering the formula of ω (2.8) we can see that $\frac{1}{2}||P(g_\Theta, \varepsilon)||^2$ is the same as $\frac{1}{2}||\omega||^2$ (i.e., by the correspondence between the a_i-s and $\alpha_i - \alpha_i^*$-s). Thus the regularized error functional (3.8) is equivalent to the primary support vector machine functional (3.1).

Consequently the regularization neural network corresponding to E_R is the same as the neural network corresponding to the original support vector machine approximation problem. □

The importance of this proof is that it effectively shows that there exists the corresponding regularization problem and in consequence it proves the first implication part of the equivalence of the two approximation frameworks. Particular examples of kernels and corresponding regularization operators are given by Smola et al. [10]. In the case when the kernel functions are from a reproducing-kernel Hilbert space (RKHS) the obtained regularization operator is the identity operator [5].

3.2. REFORMULATION OF REGULARIZATION PROBLEMS AS SVM PROBLEMS

We state and prove the theorem of the inverse implication of existence of a support vector machine neural network for a given regularization neural network. Even though this proof is basically the same as the one given by Smola et al. [10], we present it for the sake of completeness.

THEOREM 3.2. *Let P be a regularization operator, and let us consider the regularized approximation problem determined by the regularized error functional*

$$E_R = \sum_{i=1}^{n} Err(y_i, g_\Theta(x^i)) + \lambda ||Pg_\Theta||^2 \tag{3.9}$$

where Θ represents all the parameters (i.e., w_i-s and θ^i-s) of g, and the data set is $D = \{(x^1, y_1), (x^2, y_2), \ldots, (x^n, y_n)\}$, where $x^i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$. The error function can be any error function like $Err(u, v) = (u - v)^2$, or $Err(u, v) = \max(|u - v| - \varepsilon, 0)$.

If the regularized approximation problem has a solution, the regularization neural network corresponding to this problem can also be obtained as a support vector machine neural network.

Proof. Define the function set

$$H = \{h_z(t) = PG_z(t) | z \in \mathbb{R}^d\} \tag{3.10}$$

where $G_z(t) = G(t, z)$, and G is the Green function of the operator P. Define the addition and scalar multiplication operators

$$h_{z^1}(t) + h_{z^2}(t) = h_{z^1 + z^2}(t) \tag{3.11}$$

$$\beta \cdot h_z(t) = h_{\beta \cdot z}(t) \tag{3.12}$$

where $z, z^1, z^2 \in \mathbb{R}^d$ and $\beta \in \mathbb{R}$.

Then H together with the above defined addition and scalar multiplication forms a vector space of functions.

We define the vector transformation $\Phi: \mathbb{R}^d \to H$ by

$$\Phi(z) = h_z(t) \tag{3.13}$$

Using (2.14) we write

$$G_z(t) = \langle G_z(q), \delta_t \rangle = \langle G_z, P^* P G_t \rangle = \langle P G_z, P G_t \rangle = \langle h_z, h_t \rangle \tag{3.14}$$

So, we find that

$$G(z^1, z^2) = \langle h_{z^1}(t), h_{z^2}(t) \rangle = \langle \Phi(z^1), \Phi(z^2) \rangle \tag{3.15}$$

Consequently G satisfies the condition to qualify as a kernel function and we can formulate a support vector machine problem using the G functions as kernels, which leads to the same neural network as the regularized approximation problem. The existence of this uniquely defined neural network is provided by the supposition of the theorem. (For further discussions related to this theorem see [10].) □

Note that the form of the support vector machine problem depends on the error function that we use in the definition of the original problem. To get the standard support vector machine problem that corresponds to the ε-precision approximation, we have to use the ε-insensitive error function, i.e., $Err(u, v) = \max(|u - v| - \varepsilon, 0)$.

4. Conclusions

We proved both implication sides of the equivalence of the support vector machine and regularization neural networks. The standard support vector machine neural networks (i.e., those that use ε-insensitive error in the calculations), which have a unique solution, correspond to a subset of the general class of regularization neural networks. Although in general, we can find a generalized support vector machine neural network for every regularization neural network, these neural networks correspond to support vector machine problems formulated with other, not ε-insensitive, error functions (e.g., squared error). We emphasize that the regularization networks typically use squared error function and the support vector machines use typically ε-insensitive error function in their problem formulation. The equivalence between the two formulations should be understood in a generalized sense, by allowing the use of all kinds of error functions in the formulation of both problems.

As we noted in the introduction, both methods are particular realization of the general Bayesian neural network methods [1, 13]. Their advantage is that they make

the theory operational. Having the equivalence of these methods we select the best method for each particular problem. Using the transformation relations between these methods we can co-design the best fitting support vector machine, regularization or Bayesian prior that leads to the optimal solution of the problem.

References

1. Buntine, W. L. and Weigend, A. S.: Bayesian back-propagation, *Complex Systems* **5** (1991), 604–643.
2. Burges, C. J. C.: A tutorial on support vector machines for pattern recognition, *Data Mining and Knowledge Discovery* **2**(2) (1998), 121–167.
3. Drucker, H., Wu, D. and Vapnik, V. N.: Support vector machines for spam categorization, *IEEE Transactions on Neural Networks* **10**(5) (1999), 1048–1054.
4. Ellacott, S. W.: Aspects of numerical analysis of neural networks, *Acta Numerica* **3** (1994), 145–202.
5. Girosi, F.: An equivalence between sparse approximation and support vector machines, *Neural Computation* **10**(6) (1998), 1455–1481.
6. Haykin, S.: *Neural Networks: A Comprehensive Foundation*, Macmillan Publishers: Englewood Cliffs, NJ, 1994.
7. Mattera, D., Palmieri, F. and Haykin, S.: Simple and robust methods for support vector machine expansions, *IEEE Transactions on Neural Networks* **10**(5) (1999), 1038–1047.
8. Poggio, T. and Girosi, F.: Networks for approximation and learning, *Proceedings of IEEE* **78**(9) (1990), 1481–1497.
9. Schölkopf, B., Burges, C. J. C. and Smola, A. J. (eds), *Advances in Kernel Methods. Support Vector Learning*, MIT Press: Cambridge, MA, 1999.
10. Smola, A., Schölkopf, B. and Müller, K.-R.: The connection between regularization operators and support vector kernels, *Neural Networks* **11**(4) (1998), 637–650.
11. Vapnik, V. N.: An overview of statistical learning theory, *IEEE Transactions on Neural Networks* **10**(5) (1999), 988–999.
12. Vapnik, V. N.: *The Nature of Statistical Learning Theory*, Springer-Verlag: New York, 1995.
13. Williams, P. M.: Bayesian regularization and pruning using a Laplace prior, *Neural Computation* **5** (1995), 140–153.