Neural Computation 8(3), 1996, PP463-491.

A Smoothing Regularizer for Feedforward and Recurrent Neural Networks

Lizhong Wu and John Moody
Oregon Graduate Institute, Computer Science Dept., Portland, OR 97291-1000
Email: lwu@cse.ogi.edu and moody@cse.ogi.edu

Abstract

We derive a smoothing regularizer for dynamic network models by requiring robustness in prediction performance to perturbations of the training data. The regularizer can be viewed as a generalization of the first order Tikhonov stabilizer to dynamic models. For two layer networks with recurrent connections described by

$$Y(t) = \mathbf{f}\left(WY(t-\tau) + VX(t)\right) \ , \ \hat{Z}(t) = UY(t) \ ,$$

the training criterion with the regularizer is

$$D = \frac{1}{N}\sum_{t=1}^{N} ||Z(t) - \hat{Z}(\Phi, I(t))||^2 + \lambda {\rho_\tau}^2(\Phi) \ ,$$

where $\Phi = \{U, V, W\}$ is the network parameter set, $Z(t)$ are the targets, $I(t) = \{X(s), s = 1, 2, \cdots, t\}$ represents the current and all historical input information, N is the size of the training data set, ${\rho_\tau}^2(\Phi)$ is the regularizer, and λ is a regularization parameter. The closed-form expression for the regularizer for time-lagged recurrent networks is:

$$\rho_\tau(\Phi) = \frac{\gamma||U||||V||}{1-\gamma||W||}\left[1 - e^{\frac{\gamma||W||-1}{\tau}}\right] \ ,$$

where $||\ ||$ is the Euclidean matrix norm and γ is a factor which depends upon the maximal value of the first derivatives of the internal unit activations $f(\ )$. Simplifications of the regularizer are obtained for simultaneous recurrent nets ($\tau \mapsto 0$), two-layer feedforward nets, and one layer linear nets. We have successfully tested this regularizer in a number of case studies and found that it performs better than standard quadratic weight decay.

1 Introduction

One technique for preventing a neural network from overfitting noisy data is to add a regularizer to the error function being minimized. Regularizers typically smooth the fit to noisy data.[1] Well-established techniques include ridge regression (see Hoerl & Kennard (1970*a*, 1970*b*)), and more generally spline smoothing functions or Tikhonov stabilizers that penalize the m^{th}-order squared derivatives of the function being fit, as in Tikhonov & Arsenin (1977), Eubank (1988), Hastie & Tibshirani (1990) and Wahba (1990). These methods have recently been extended to networks of radial basis functions (Powell (1987), Poggio & Girosi (1990), Girosi, Jones & Poggio (1995)) and several heuristic approaches have been developed for sigmoidal neural networks, for example, quadratic weight decay (Plaut, Nowlan & Hinton (1986)), weight elimination (Scalettar & Zee (1988),Chauvin (1990),Weigend, Rumelhart & Huberman (1990)), soft weight sharing (Nowlan & Hinton (1992)) and curvature-driven smoothing (Bishop (1993)).[2] Quadratic weight decay (which is equivalent to ridge regression) and weight elimination are frequently used "on-line" during stochastic gradient learning. The other regularizers listed above are not generally used with on-line algorithms, but rather with "batch" or deterministic optimization methods. All previous studies on regularization have concentrated on feedforward neural networks. To our knowledge, recurrent learning with regularization has not been reported before.

In Section 2 of this paper, we develop a smoothing regularizer for general dynamic models which is derived by considering perturbations of the training data. We demonstrate that this regularizer corresponds to a dynamic generalization of the well-known first order Tikhonov stabilizer. We then present a closed-form expression for our regularizer for two layer feedforward and recurrent neural networks, with standard weight decay being a special case. In Section 3, we evaluate our regularizer's performance on a number of applications, including regression with feedforward and recurrent neural networks and predicting the U.S. Index of Industrial Production. The advantage of our regularizer is demonstrated by comparing to standard weight decay in both feedforward and recurrent modeling. Finally, we discuss several related questions and conclude our paper in Section 4.

2 Smoothing Regularization

2.1 Prediction Error for Perturbed Data Sets

Consider a training data set $\{P : Z(t), X(t)\}$, where the targets $Z(t)$ are assumed to be generated by an unknown dynamical system $F^*(I(t))$ and an unobserved noise process:

$$Z(t) = F^*(I(t)) + \varepsilon^*(t) \quad \text{with} \quad I(t) = \{X(s), s = 1, 2, \cdots, t\} \ . \tag{1}$$

Here, $I(t)$ is the information set containing both current and past inputs $X(s)$, and the $\varepsilon^*(t)$ are independent random noise variables with zero mean and variance σ^{*2}. Consider next a dynamic network model $\hat{Z}(t) = F(\Phi, I(t))$ to be trained on data set P, where Φ represents a set of network parameters, and $F(\ )$ is a network transfer function which is assumed to be nonlinear and dynamic. We assume that $F(\ )$ has good approximation capabilities, such that $F(\Phi_P, I(t)) \approx F^*(I(t))$ for learnable parameters Φ_P.

Our goal is to derive a smoothing regularizer for a network trained on the actual data set P that in effect optimizes the expected network performance (prediction risk) on perturbed test data sets of form $\{Q$:

[1] Other techniques to prevent overfitting include early stopping of training (which can be viewed as having an effect similar to weight decay [Sjöberg & Ljung (1992, 1995)]) and using prior knowledge in the form of hints (see Abu-Mostafa (1995), Tresp, Hollatz & Ahmad (1993), and references therein). Smoothing regularization can be viewed as a special class of hints.

[2] Two additional papers related to ours, but dealing only with feed forward networks, came to our attention or were written after our work was completed. These are Bishop (1995) and Leen (1995). Also, Moody & Rögnvaldsson (1995) have recently proposed several new classes of smoothing regularizers for feedforward nets.

$\tilde{Z}(t), \tilde{X}(t)\}$. The elements of Q are related to the elements of P via small random perturbations $\varepsilon_z(t)$ and $\varepsilon_x(t)$, so that

$$\tilde{Z}(t) = Z(t) + \varepsilon_z(t) \,, \tag{2}$$

$$\tilde{X}(t) = X(t) + \varepsilon_x(t) \,. \tag{3}$$

The $\varepsilon_z(t)$ and $\varepsilon_x(t)$ have zero mean and variances ${\sigma_z}^2$ and ${\sigma_x}^2$ respectively. The training and test errors for the data sets P and Q are

$$D_P = \frac{1}{N}\sum_{t=1}^{N}[Z(t) - F(\Phi_P, I(t))]^2 \tag{4}$$

$$D_Q = \frac{1}{N}\sum_{t=1}^{N}[\tilde{Z}(t) - F(\Phi_P, \tilde{I}(t))]^2 \;\;, \tag{5}$$

where Φ_P denotes the network parameters obtained by training on data set P, and $\tilde{I}(t) = \{\tilde{X}(s), s = 1, 2, \cdots, t\}$ is the perturbed information set of Q. With this notation, our goal is to minimize the expected value of D_Q, while training on D_P.

Consider the prediction error for the perturbed data point at time t:

$$d(t) = [\tilde{Z}(t) - F(\Phi_P, \tilde{I}(t))]^2 \,. \tag{6}$$

With Eqn (2), we obtain

$$\begin{aligned} d(t) &= [Z(t) + \varepsilon_z(t) - F(\Phi_P, I(t)) + F(\Phi_P, I(t)) - F(\Phi_P, \tilde{I}(t))]^2, \\ &= [Z(t) - F(\Phi_P, I(t))]^2 + [F(\Phi_P, I(t)) - F(\Phi_P, \tilde{I}(t))]^2 + [\varepsilon_z(t)]^2 \\ &\quad +2[Z(t) - F(\Phi_P, I(t))][F(\Phi_P, I(t)) - F(\Phi_P, \tilde{I}(t))] + 2\varepsilon_z(t)[Z(t) - F(\Phi_P, \tilde{I}(t))]. \end{aligned} \tag{7}$$

Assuming that $\varepsilon_z(t)$ is uncorrelated with $[Z(t) - F(\Phi_P, \tilde{I}(t))]$ and averaging over the exemplars of data sets P and Q, Eqn(7) becomes

$$\begin{aligned} D_Q &= D_P + \frac{1}{N}\sum_{t=1}^{N}[F(\Phi_P, I(t)) - F(\Phi_P, \tilde{I}(t))]^2 + \frac{1}{N}\sum_{t=1}^{N}[\varepsilon_z(t)]^2 \\ &\quad +\frac{2}{N}\sum_{t=1}^{N}[Z(t) - F(\Phi_P, I(t))][F(\Phi_P, I(t)) - F(\Phi_P, \tilde{I}(t))] \,. \end{aligned} \tag{8}$$

The third term, $\sum_{t=1}^{N}[\varepsilon_z(t)]^2$, in Eqn(8) is independent of the weights, so it can be neglected during the learning process. The fourth term in Eqn(8) is the cross-covariance between $[Z(t) - F(\Phi_P, I(t))]$ and $[F(\Phi_P, I(t)) - F(\Phi_P, \tilde{I}(t))]$. We argue in Appendix A that this term can also be neglected.

2.2 The Dynamic Smoothing Regularizer and Tikhonov Correspondence

The above analysis shows that the expected test error D_Q can be minimized by minimizing the objective function D:

$$D = \frac{1}{N}\sum_{t=1}^{N}[Z(t) - F(\Phi, I(t))]^2 + \frac{1}{N}\sum_{t=1}^{N}[F(\Phi_P, I(t)) - F(\Phi_P, \tilde{I}(t))]^2 \;\;. \tag{9}$$

In Eqn (9), the second term is the time average of the squared disturbance $||\hat{\tilde{Z}}(t) - \hat{Z}(t)||^2$ of the trained network output due to the input perturbation $||\tilde{I}(t) - I(t)||^2$. Minimizing this term demands that small changes

in the input variables yield correspondingly small changes in the output. This is the standard smoothness prior, namely that if nothing else is known about the function to be approximated, a good option is to assume a high degree of smoothness. Without knowing the correct functional form of the dynamical system F^* or using such prior assumptions, the data fitting problem is ill-posed.

It is straightforward to see that the second term in Eqn (9) corresponds to the standard first order Tikhonov stabilizer.[3] Expanding to first order in the input perturbations ε_x, the expectation value of this term becomes:

$$\begin{aligned} \left\langle \frac{1}{N}\sum_{t=1}^{N}[F(\Phi_P,\tilde{I}(t)) - F(\Phi_P,I(t))]^2 \right\rangle &\approx \frac{1}{N}\sum_{t=1}^{N}\left\langle \left[\sum_{s=1}^{t}\frac{\partial F(\Phi_P,I(t))}{\partial X(s)}\varepsilon_x(s)\right]^2 \right\rangle \\ &= \sigma_x^2\frac{1}{N}\sum_{t=1}^{N}\sum_{s=1}^{t}\left[\frac{\partial F(\Phi_P,I(t))}{\partial X(s)}\right]^2 \quad . \end{aligned} \tag{10}$$

If the dynamics are trivial, so that the mapping F^* has no recurrence, then:

$$\frac{\partial F(\Phi_P,I(t))}{\partial X(s)} = 0 \ \text{ for } \ s \neq t \ , \tag{11}$$

and Eqn (10) reduces to

$$\sigma_x^2\frac{1}{N}\sum_{t=1}^{N}\left[\frac{\partial F(\Phi_P,X(t))}{\partial X(t)}\right]^2 \quad . \tag{12}$$

This is the usual first order Tikhonov stabilizer weighted by the empirical distribution.

In Eqns (12) and (10), σ_x^2 plays the role of a regularization parameter λ that determines the compromise between the degree of smoothness of the solution and its fit to the noisy training data. This is the usual bias/variance tradeoff (see Geman, Bienenstock & Doursat (1992)).

A reasonable choice for the value of σ_x^2 is to set it proportional to the average squared nearest neighbor distance in the input data. For normalized input data (eg. where each variable has zero mean and unit variance), one can estimate the average nearest neighbor distance as:

$$\lambda \approx \sigma_x^2 \approx \kappa N^{-2/\mathcal{D}} \ , \tag{13}$$

where $\mathcal{D}$ is the intrinsic dimension of the input data (less than or equal to the number of input variables) and κ is a geometrical factor (of order unity in low dimensions).

To summarize this section, the training objective function D of Eqn (9) can be written in approximate form as:

$$D \approx \frac{1}{N}\sum_{t=1}^{N}[Z(t) - F(\Phi,I(t))]^2 + \lambda\frac{1}{N}\sum_{t=1}^{N}\sum_{s=1}^{t}\left[\frac{\partial F(\Phi_P,I(t))}{\partial X(s)}\right]^2 \ , \tag{14}$$

where the second term is a dynamic generalization of the first order Tikhonov stabilizer.

2.3 Form of the Proposed Smoothing Regularizer for Two Layer Networks

Consider a general, two layer, nonlinear, dynamic network with recurrent connections on the internal layer [4] as described by

$$\begin{aligned} Y(t) &= \mathbf{f}\left(WY(t-\tau) + VX(t)\right) \\ \hat{Z}(t) &= UY(t) \ , \end{aligned} \tag{15}$$

[3] Bishop (1995) has independently made this observation for the case of feedforward networks.

[4] Our derivation can easily be extended to other network structures.

where $X(t)$, $Y(t)$ and $\hat{Z}(t)$ are respectively the network input vector, the hidden output vector and the network output; $\Phi = \{U, V, W\}$ is the output, input and recurrent connections of the network; $\mathbf{f}(\ )$ is the vector-valued nonlinear transfer function of the hidden units; and τ is a time delay in the feedback connections of hidden layer which is pre-defined by a user and will not be changed during learning. τ can be zero, a fraction, or an integer, but we are interested in the cases with a small τ.[5]

When $\tau = 1$, our model is a recurrent network as described by Elman (1990) and Rumelhart, Hinton & Williams (1986) (see Figure 17 on page 355). When τ is equal to some fraction smaller than one, the network evolves $\frac{1}{\tau}$ times within each input time interval.[6] When τ decreases and approaches zero, our model is the same as the network studied by Pineda (1989), and earlier, widely-studied recurrent networks[7] (see, for example, Grossberg (1969), Amari (1972), Sejnowski (1977) and Hopfield (1984)). In Pineda (1989), τ was referred to as the *network relaxation time scale*. Werbos (1992) distinguished the recurrent networks with zero τ and non-zero τ by calling them *simultaneous recurrent networks* and *time-lagged recurrent networks* respectively.

We show in Appendix B that minimizing the second term of Eqn(9) can be obtained by smoothing the output response to an input perturbation at every time step, and we have:

$$||\hat{\tilde{Z}}(t) - \hat{Z}(t)||^2 \leq {\rho_\tau}^2(\Phi_P)||\tilde{X}(t) - X(t)||^2 \ \text{ for } \ t = 1, 2, \ldots, N \ . \tag{16}$$

We call ${\rho_\tau}^2(\Phi_P)$ the *output sensitivity* of the trained network Φ_P to an input perturbation. ${\rho_\tau}^2(\Phi_P)$ is determined by the network parameters only and is independent of the time variable t.

We obtain our new regularizer by training directly on the expected prediction error for perturbed data sets Q. Based on the analysis leading to Eqns (9) and (16), the training criterion thus becomes

$$D = \frac{1}{N}\sum_{t=1}^{N}[Z(t) - F(\Phi, I(t))]^2 + \lambda {\rho_\tau}^2(\Phi) \ . \tag{17}$$

As in Eqn (14), the coefficient λ in Eqn(17) is a regularization parameter that measures the degree of input perturbation $||\tilde{I}(t) - I(t)||^2$. Note that the subscript P has been dropped from Φ, since D is now the training objective function for any set of weights. Also note in comparing Eqn (17) to Eqn (14) that the sum over the past history indexed by s no longer appears, and that a trivial factor $\frac{1}{N}\sum_{t=1}^{N} \equiv 1$ has been dropped. These simplifications are due to our minimizing the *zero-memory response* at each time step during training as described after Eqn (71) in Appendix B.

The algebraic form for $\rho_\tau(\Phi)$ as derived in Appendix B is:

$$\rho_\tau(\Phi) = \frac{\gamma||U||||V||}{1 - \gamma||W||}\left\{1 - \exp\left(\frac{\gamma||W|| - 1}{\tau}\right)\right\} \ , \tag{18}$$

for time-lagged recurrent networks ($\tau > 0$). Here, $||\ ||$ denotes the Euclidean matrix norm.[8] The factor γ

[5] When the time delay τ exceeds some critical value, a recurrent network becomes unstable and lies in oscillatory modes. See, for example, Marcus & Westervelt (1989).

[6] When τ is a fraction smaller than one, the hidden node's function can be described by:

$$Y(t + k\tau - 1) = \mathbf{f}\left(WY(t + (k-1)\tau - 1) + VX(t)\right) \ \text{ for } \ k = 1, 2, \ldots, \frac{1}{\tau}.$$

The input $X(t)$ is kept fixed during the above evolution.

[7] These were called *additive networks* .

[8] The Euclidean norm of a real $M \times N$ matrix A is

$$||A|| = \left[tr(A^T A)\right]^{\frac{1}{2}} = \left(\sum_{i=1}^{M}\sum_{j=1}^{N} a_{ij}^2\right)^{\frac{1}{2}}$$

where A^T is the transpose of A and a_{ij} is the element of A.

depends upon the maximal value of the first derivatives of the activation functions of the hidden units and is given by:

$$\gamma = \max_{t,j} \mid f_j{}'(o_j(t)) \mid , \tag{19}$$

where j is the index of hidden units and $o_j(t)$ is the input to the j^{th} unit. In general, $\gamma \leq 1$. [9] To insure stability and that the effects of small input perturbations are damped out, it is required that

$$\gamma||W|| < 1 \ . \tag{20}$$

The regularizer Eqn(18) can be deduced for the simultaneous recurrent networks in the limit $\tau \mapsto 0$ by:

$$\rho(\Phi) \equiv \rho_0(\Phi) = \frac{\gamma||U||||V||}{1-\gamma||W||} \ . \tag{21}$$

If the network is feedforward, $W = 0$ and $\tau = 0$, Eqns (18) and (21) become

$$\rho(\Phi) = \gamma||U||||V|| \ . \tag{22}$$

Moreover, if there is no hidden layer and the inputs are directly connected to the outputs via U, the network is an ordinary linear model, and we obtain

$$\rho(\Phi) = ||U|| \ , \tag{23}$$

which is standard quadratic weight decay (Plaut *et al.* 1986) as is used in ridge regression (see Hoerl & Kennard (1970*a*, 1970*b*)).

The regularizer (Eqn(22) for feedforward networks and Eqn (18) for recurrent networks) was obtained by requiring smoothness of the network output to perturbations of data. We therefore refer to it as a smoothing regularizer. Several approaches can be applied to estimate the regularization parameter λ, as in Eubank (1988), Hastie & Tibshirani (1990) and Wahba (1990). We will not discuss this subject in this paper.

After including a regularization term in training, the weight update equation becomes

$$\begin{aligned} \Delta\Phi &= -\eta\nabla_\Phi D \\ &= -\eta\{\nabla_\Phi D_P + \lambda\nabla_\Phi[{\rho_\tau}^2(\Phi)]\} \ , \end{aligned} \tag{24}$$

where η is a learning rate. With our smoothing regularizer, $\nabla_\Phi[{\rho_\tau}^2(\Phi)]$ is computed as:

$$\frac{\partial {\rho_\tau}^2}{\partial u_{ij}} = \frac{2{\rho_\tau}^2}{\|U\|^2}u_{ij} \tag{25}$$

$$\frac{\partial {\rho_\tau}^2}{\partial v_{ij}} = \frac{2{\rho_\tau}^2}{\|V\|^2}v_{ij} \tag{26}$$

$$\frac{\partial {\rho_\tau}^2}{\partial w_{ij}} = \frac{2\gamma{\rho_\tau}^2}{\|W\|}\left[\frac{1}{\tau\left[1-\exp\left(\frac{1-\gamma\|W\|}{\tau}\right)\right]} + \frac{1}{1-\gamma\|W\|}\right]w_{ij} \tag{27}$$

where u_{ij}, v_{ij} and w_{ij} are the elements of U, V and W respectively. For simultaneous recurrent networks, Eqn (27) becomes

$$\frac{\partial \rho^2}{\partial w_{ij}} = \frac{2\gamma\rho^2}{\|W\|}\left(\frac{1}{1-\gamma\|W\|}\right)w_{ij} \ . \tag{28}$$

[9] For instance, $f'(x) = [1-f(x)]f(x)$ if $f(x) = \frac{1}{1+e^{-x}}$. In this case, $\gamma = \max \mid f'(x)) \mid = \frac{1}{4}$ at $x = 0$. If $|x|$ is much larger than 0, then $f(x)$ operates in its asymptotic region, and $\mid f'(x)) \mid$ will be far less than $\frac{1}{4}$. In fact, γ is exponentially small in this case.

When standard weight decay is used, the regularizer for Eqn(15) is:

$$\rho^2_{\text{weight decay}}(\Phi) = ||U||^2 + ||V||^2 + ||W||^2 \ . \tag{29}$$

The corresponding update equations for this case are:

$$\frac{\partial \rho^2}{\partial u_{ij}} = 2u_{ij} \tag{30}$$

$$\frac{\partial \rho^2}{\partial v_{ij}} = 2v_{ij} \tag{31}$$

$$\frac{\partial \rho^2}{\partial w_{ij}} = 2w_{ij} \ . \tag{32}$$

In contrast to our smoothing regularizer, quadratic weight decay treats all network weights identically, makes no distinction between recurrent and input/output weights, and takes into account no interactions between weight values.

In the next section, we evaluate the new regularizer in a number of tests. In each case, we compare the networks trained with the smoothing regularizer to those trained with standard weight decay.

3 Empirical Tests

In this section, we demonstrate the efficacy of our smoothing regularizer via three empirical tests. The first two tests are on regression with some synthetic data, and the third test is on predicting the monthly U.S. Index of Industrial Production.

3.1 Regression with Feedforward Networks

We form a set of data generated by a pre-defined function G. The data are contaminated by some degree of zero-mean Gaussian noise before being used for training. Our task is to train the networks so that they estimate G. We will first study function estimation with feedforward networks, and then extend it to the case recurrent networks.

Data:

The data in this test are synthesized with the function

$$s(x) = \left[\frac{1}{1+e^{-a(x+b)}} - \frac{1}{1+e^{-a(x-b)}}\right] + \varepsilon \ , \tag{33}$$

where x is uniformly distributed within $[-10, 10]$, ε is normally distributed with zero mean and variance σ^2, and a and b are two constants. In our test, we set $a = 1$ and $b = 5$.

Model:

The model we have used for the above data is a two-hidden unit, feedforward network with sigmoidal functions at hidden units and a linear function at a single output unit. It can be described as

$$Z(x) = u_0 + \frac{u_1}{1+e^{-(v_1 x + v_{10})}} + \frac{u_2}{1+e^{-(v_2 x + v_{20})}} \ . \tag{34}$$

The model overall has 7 weight parameters.

Table 1: Comparison the performances (as measured by Eqn(35)) between the feedforward networks trained with the smoothing regularizer and those trained with standard weight decay for the function estimation. The results shown are the mean and the standard deviation over 10 models with different initial weights.

Number of Train -ing Patterns	Noise Variance	With Standard Weight Decay	With Smoothing Regularizer
11	0.1	0.037±0.011	0.020±0.003
	0.5	0.137±0.003	0.076±0.028
	1.0	0.151±0.000	0.117±0.011
21	0.1	0.014±0.003	0.010±0.000
	0.5	0.061±0.004	0.048±0.042
	1.0	0.097±0.005	0.068±0.009
41	0.1	0.011±0.000	0.008±0.000
	0.5	0.038±0.001	0.028±0.000
	1.0	0.066±0.001	0.050±0.000

Performance Measure:

The criterion to evaluate the model performance is the true Mean Squared Error (MSE) minus the noise variance σ^2:

$$D = \int_{-x_0}^{x_0} [G(x) - Z(x)]^2 p(x) dx \ , \tag{35}$$

where $G(x)$ is the noiseless, source function as shown in the first part of Eqn(33), $Z(x)$ is the network response function as given by Eqn(34), and $p(x)$ is the probability density of x. In this experiment, $p(x)$ is uniformly distributed within $[-x_0, x_0]$ and $x_0 = 10$.

Results:

Comparisons between the smoothing regularizer and standard weight decay are listed in Table 1. The performance comparisons are made for a number of cases. The numbers of training patterns are varied from 11, 21 and 41. The noise variances are from 0.1, 0.5 to 1.0. To observe the effect of the regularization parameters, we did not use their estimated values. Instead, the regularization parameters for both the smoothing regularizer and standard weight decay are varied from 0 to 0.1 with step-size 0.001. Figure 1 shows the downsampled training and test errors versus the regularization parameters. The performances in Table 1 are the optimal results over all these regularization parameters. This gives the best potential result each network can obtain. Unlike our other tests in real world applications, neither early stopping nor validation was applied in this test. Each network was trained over 5000 epochs. It was found that for all networks, the training error did not decrease significantly after 5000 training epochs. With these conditions, the task to prevent the network from over-training or over-fitting is completely dependent on the regularization. We believe that such results will more directly reflect, and more precisely compare, the efficacy of different regularizers.

Table 1 shows that the potential predictive errors with the smoothing regularizer are smaller than those with standard weight decay.

Figure 2 gives an example and compares the approximation functions obtained with standard weight decay and our smoothing regularizer to the true function. We can see that the function obtained with our smoothing regularizer is obviously closer to the true function than that obtained with standard weight decay.

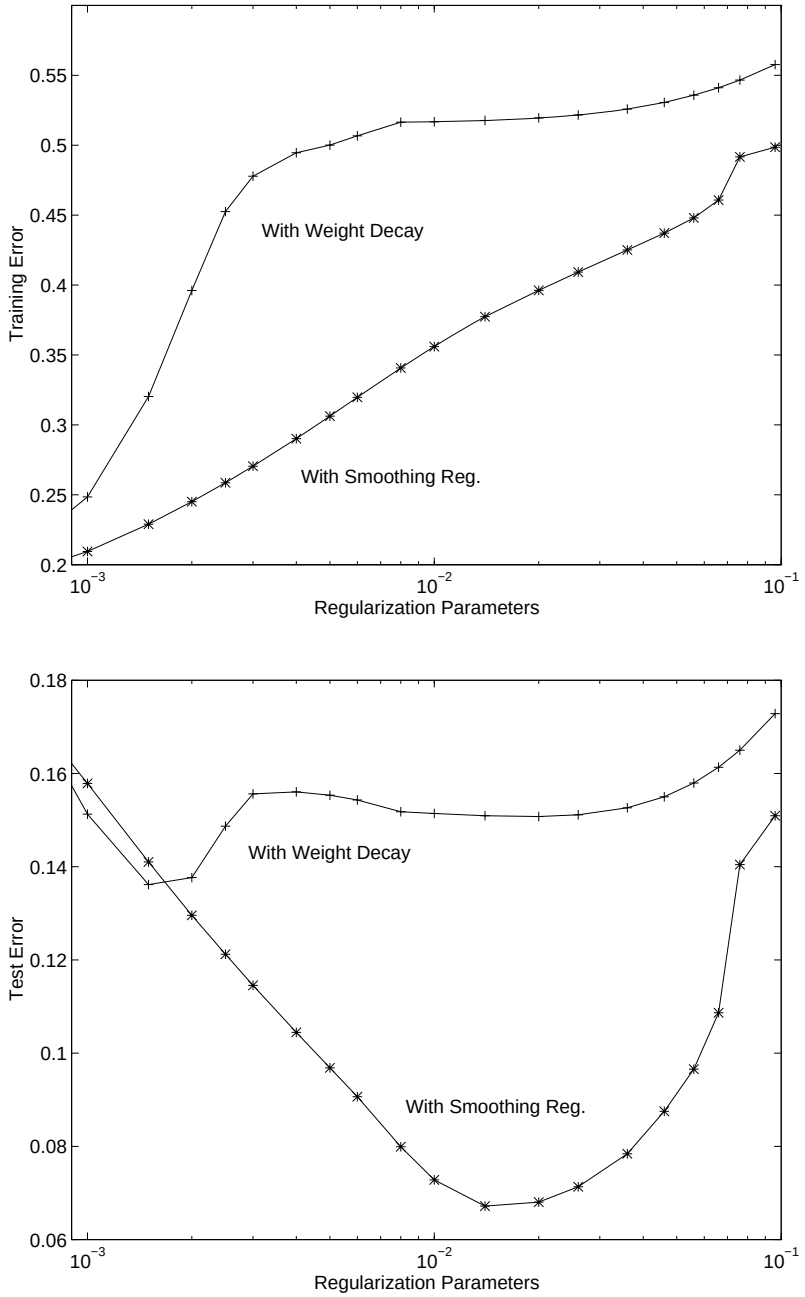

Figure 1: Training (upper panel) and test (lower panel) errors versus regularization parameters. Networks trained with ordinary weight decay are plotted by '+', and those trained with smoothing regularizers are plotted by '$\star$'. 10 different networks are shown for each case. The curves are the median errors of these 10 networks.

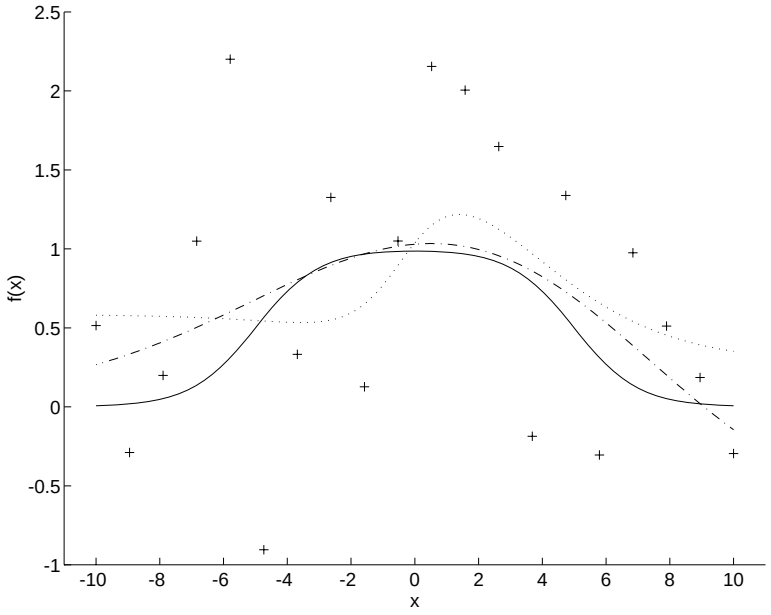

Figure 2: Comparing the estimated function obtained with our smoothing regularizer (dashed curve) and that with standard weight decay (dotted curve) to the true function (solid curve). The '+' plots 21 training patterns that are uniformly distributed along the x axis and normally distributed along the $f(x)$ direction with noise variance 1. The model is a two-node, feedforward network.

3.2 Regression With Recurrent Networks

Data:

For recurrent network modeling, we synthesized a data sequence of N samples with the following dynamic function:

$$x(t) = 10\left(\frac{2t}{N-1} - 1\right) \tag{36}$$

$$y_1(t) = \frac{1}{1+e^{-a[x(t)+y_2(t-1)+b]}} \tag{37}$$

$$y_2(t) = \frac{1}{1+e^{-a[x(t)+y_1(t-1)-b]}} \tag{38}$$

$$s(t) = y_1(t) + y_2(t) + \varepsilon(t), \tag{39}$$

where $t = 0, 1, \cdots, N-1$, $\varepsilon(t)$ is normally distributed with zero mean and variance σ^2, and $a = 1$ and $b = 5$ are two constants. Two dummy variables, $y_1(t)$ and $y_2(t)$, evolve from their previous values. In our test, we initialize $y_1(t) = y_2(t) = 0$.

Model:

The model we have used for the above data is a two-hidden unit, recurrent network with sigmoidal functions at hidden units and a linear function at a single output unit. The output of a hidden unit is time-delayed and fed back to another hidden unit input. It can be described as

$$y_1(t) = \frac{1}{1+e^{-[w_1 y_2(t-1)+v_1 x(t)+v_{10}]}} \tag{40}$$

$$y_2(t) = \frac{1}{1+e^{-[w_2 y_1(t-1)+v_2 x(t)+v_{20}]}} \tag{41}$$

$$z(t) = u_0 + u_1 y_1(t) + u_2 y_2(t) \tag{42}$$

where $y_1(t)$ and $y_2(t)$ correspond to the two hidden-unit outputs. The model overall has 9 weight parameters.

Results:

The performance measure is the same as Eqn(35) in the case for feedforward networks. Table 2 lists the performances of the recurrent networks trained with standard weight decay and those with our smoothing regularizer. The table shows the results with the best value of regularization parameters. It again shows that, in all case, the smoothing regularizer outperforms standard weight decay. For all networks listed in Table 2, the numbers of training patterns are varied from 11, 21 and 41. The noise variances are from 0.1, 0.5 to 1.0. The regularization parameters for both the smoothing regularizer and standard weight decay are varied from 0 to 0.1 with step-size 0.001.

3.3 Predicting the U.S. Index of Industrial Production

Data:

The Index of Industrial Production (IP) is one of the key measures of economic activity. It is computed and published monthly. Our task is to predict the one-month rate of change of the index from January 1980 to December 1989 for models trained from January 1950 to December 1979. The exogenous inputs we have used include 8 time series such as the index of leading indicators, housing starts, the money supply M2, the $S\&P$ 500 Index. These 8 series are also recorded monthly. In previous studies by Moody, Levin & Rehfuss (1993), with the same defined training and test data sets, the normalized prediction errors of the one month rate of change were 0.81 with the **neuz** neural network simulator, and 0.75 with the **proj** neural network simulator. [10]

[10]The **neuz** networks were trained using stochastic gradient descent, early stopping via a validation set, and the PCP

Table 2: Comparison between the recurrent networks trained with the smoothing regularizer and those trained with standard weight decay for the function estimation task. The results shown are averaged over 10 different initial weights.

Number of Train -ing Patterns	Noise Variance	With Standard Weight Decay	With Smoothing Regularizer
11	0.1	0.035±0.006	0.020±0.002
	0.5	0.123±0.008	0.067±0.007
	1.0	0.151±0.000	0.111±0.015
21	0.1	0.014±0.001	0.009±0.000
	0.5	0.058±0.002	0.037±0.001
	1.0	0.095±0.004	0.071±0.001
41	0.1	0.009±0.000	0.006±0.000
	0.5	0.032±0.001	0.024±0.005
	1.0	0.057±0.001	0.039±0.016

Model:

We have simulated feedforward and recurrent neural network models. Both models consist of two layers. There are 9 input units in the recurrent model, which receive the 8 exogenous series and the previous month IP index change. We set the time-delayed length in the recurrent connections $\tau = 1$. The feedforward model is constructed with 36 input units, which receive 4 time-delayed versions of each input series. The time-delay lengths are 1, 3, 6 and 12, respectively. The activation functions of hidden units in both feedforward and recurrent models are *tanh* functions. The number of hidden units varies from 2 to 6. Each model has one linear output unit.

Training:

We have divided the data from January 1950 to December 1979 into four non-overlapping sub-sets. One sub-set consists of 70% of the original data and each of the other three subsets consists of 10% of the original data. The larger sub-set is used as training data and the three smaller sub-sets are used as validation data. These three validation data sets are respectively used for determination of early stopped training, selecting the regularization parameter and selecting the number of hidden units.

We have formed 10 random training-validation partitions. For each training-validation partition, three networks with different initial weight parameters are trained. Therefore, our prediction committee is formed by 30 networks.

The error of committee is the average of errors of all committee members. All networks in the committee are trained simultaneously and stopped at the same time based on the committee error of a validation set. The value of the regularization parameter and the number of hidden units are determined by minimizing the committee error on separate validation sets.

Results:

Table 3 lists the results over the test data set. The performance measure is the normalized prediction error as

regularization method proposed by Levin, Leen & Moody (1994) The **proj** networks were trained using the Levenburg-Marquardt algorithm, and network pruning after training was accomplished via the methods described in Moody & Utans (1992). The internal layer nonlinearities for the **neuz** networks were sigmoidal, while some of the **proj** networks included quadratic nonlinearities as described in Moody & Yarvin (1992).

Table 3: Normalized prediction errors for the one-month rate of return on the U.S. Index of Industrial Production (Jan. 1980 - Dec. 1989). Each result is based on 30 networks.

Model	Regularizer	Mean $\pm$ Std	Median	Max	Min	Committee
Recurrent	Smoothing	0.646±0.008	0.647	0.657	0.632	0.639
Networks	Weight Decay	0.734±0.018	0.737	0.767	0.704	0.734
Feedforward	Smoothing	0.700±0.023	0.707	0.729	0.654	0.693
Networks	Weight Decay	0.745±0.043	0.748	0.805	0.676	0.731

used in Moody *et al.* (1993), which is defined as

$$D_Q = \frac{\sum_{t\in Q}[S(t) - \hat{S(t)}]^2}{\sum_{t\in Q}[S(t) - \bar{S}]^2} \ , \tag{43}$$

where $S(t)$ stands for the observations, Q represents the test data set and $\bar{S}$ is the mean of $S(t)$ over the training data set. This measure evaluates prediction accuracy by comparing to a trivial predictor that uses the mean of the training data as its prediction.

Table 3 also compares the out-of-sample performance of recurrent networks and feedforward networks trained with our smoothing regularizer to that of networks trained with standard weight decay. The results are based on 30 networks. As shown, the smoothing regularizer again outperforms standard weight decay with 95% confidence (in t-distribution hypothesis) in both cases of recurrent networks and feedforward networks. We also list the median, maximal and minimal prediction errors over 30 predictors. The last column gives the committee results, which are based on the simple average of 30 network predictions. We see that the median, maximal and minimal values and the committee results obtained with the smoothing regularizer are all smaller than those obtained with standard weight decay, in both recurrent and feedforward network models.

Figure 3 plots the changes of prediction errors with the regularization parameter in recurrent neural network modeling. As shown by the figure, the prediction error over the training data set increases with the regularization parameter, the prediction errors over the validation and test data sets first decrease and then increase with the regularization parameter. The optimal regularization parameter with the least validation error is 0.8 with our smoothing regularizer and 0.03 with standard weight decay. In all cases, we have found that the regularization parameters should be larger than zero to achieve optimal prediction performance. This confirms the necessity of regularization during training in addition to early stopped training.

We have observed and compared the weight histogram of the networks trained with our smoothing regularizer and those with standard weight decay. As demonstrated in Figure 4, although the distribution has heavy tail, most weights parameters in the networks with the smoothing regularizer are more concentrated on small values. Its distribution is more like a real symmetric α-stable ($S\alpha S$) [11] distribution rather than a Gaussian distribution. This is also consistent with the soft weight-sharing approach proposed by Nowlan & Hinton (1992), in which a Gaussian mixture is used to model the weight distribution. We thus believe that our smoothing regularizer provides a more effective means to differentiate "essential" large weights from "irrelevant" small weights than does standard weight decay.

With and Without Early Stopping of Training:

The results shown in Table 3 and Figures 3 and 4 are for networks trained with both the regularization and early stopping techniques. From Figure 3, we see that the prediction performance is far worse than the optimum if the network is trained with just early stopping and no regularization ($\lambda = 0$).

Another case is that the network is trained with regularization and without early stopping. We compare the

[11] See, for example, Shao & Nikias (1993).

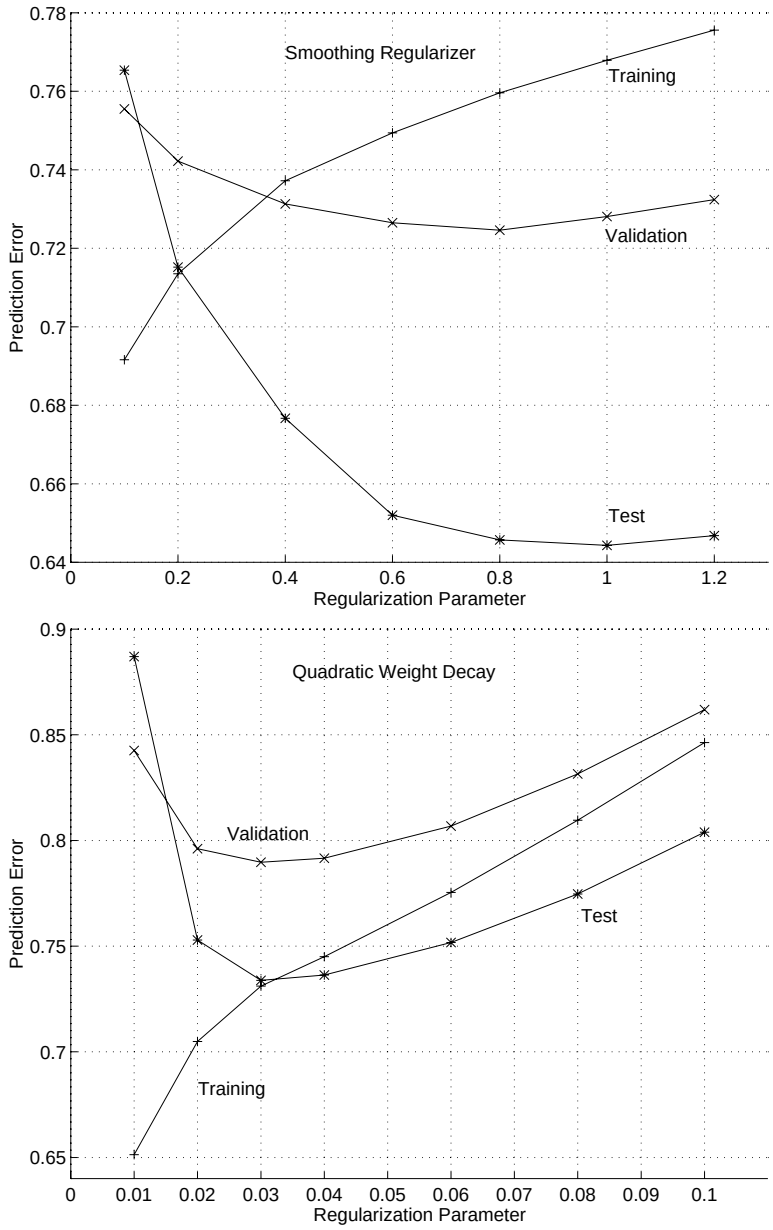

Figure 3: Regularization parameter vs normalized prediction errors for the task of predicting the U.S. Index of Industrial Production. The example given is for a recurrent network trained with either the smoothing regularizer (upper panel) or standard weight decay (lower panel). For the smoothing regularizer, the optimal regularization parameter which leads to the least validation error is 0.8 corresponding to a test error of 0.646. For standard weight decay, the optimal regularization parameter is 0.03 corresponding to a test error of 0.734. The new regularizer thus yields a 12% reduction of test error relative to that obtained using quadratic weight decay.

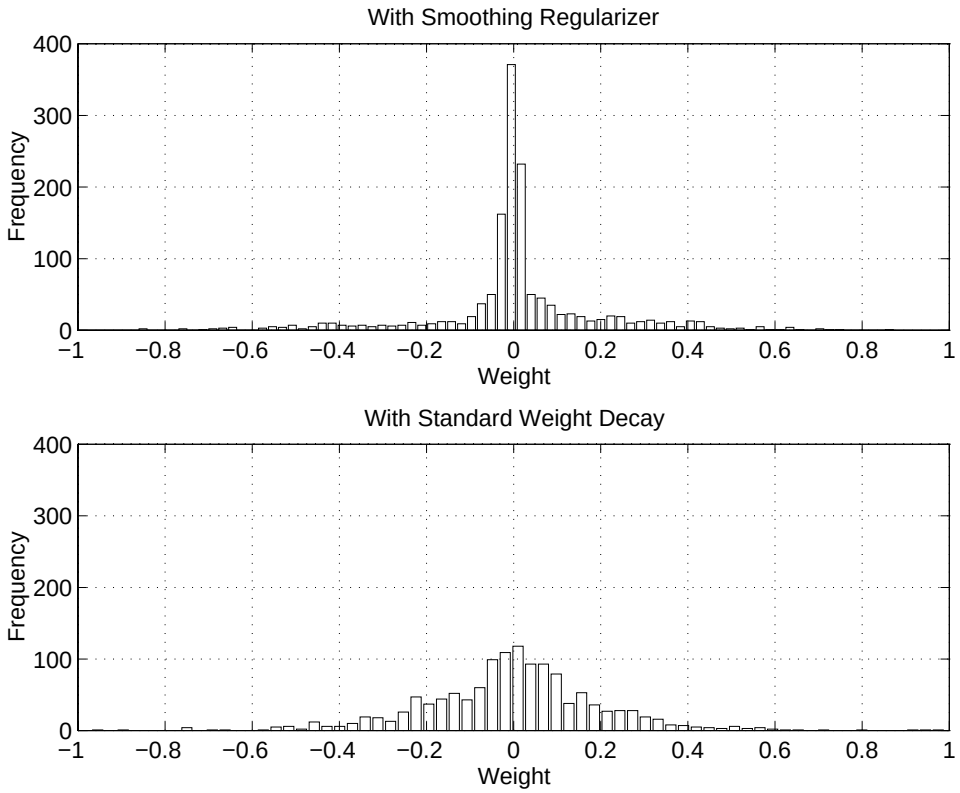

Figure 4: Comparison of the weight histogram between the recurrent networks trained with our smoothing regularizer and those with standard weight decay. Each histogram summarizes 30 networks trained on the IP task. The smoothing regularizer yields a symmetric α-stable (or lepto-kurtic) distribution of weights (large peak near zero and long tails), whereas the quadratic weight decay produces a distribution that is closer to a Gaussian. The smoothing regularizer thus distinguishes more sharply between "essential" (large) weights and "non-essential" (near-zero-valued) weights. The near-zero-valued weights can be pruned.

Table 4: Comparison of prediction performance of the networks trained with and without early stopping of training. Results given in the table are the normalized prediction errors for the IP task as those shown in Table 3. All the results are based on 30 recurrent networks. Whether trained with early stopping or not, the networks are both trained with the smoothing regularizer.

Training	Mean $\pm$ Std	Median	Max	Min	Committee
With Early Stopping	0.646$\pm$0.008	0.647	0.657	0.632	0.639
Without Early Stopping	0.681$\pm$0.057	0.664	0.938	0.643	0.657

performances between the networks trained with regularization and early stopping and the networks trained with regularization but without early stopping in Table 4. For the latter networks, those 10% of the data originally used for early stopping are now used in training. All other training conditions are the same for both cases. From the table, we see that the performance of networks without early stopping is slightly worse than those with regularization and early stopping simultaneously. However, the difference is small in terms of the median or committee errors, even though the deviation of prediction errors has increased.

Stability Analysis:

In Section 2, we found that Eqn(20) (i.e. $\gamma||W|| < 1$) must hold to insure stability and that the effects of small input perturbations are damped out. Figure 5 shows the value of $\gamma||W||$ of the trained networks. The networks trained with the optimal regularization parameter satisfy the inequality, and those networks trained with regularization parameters which are much larger or smaller than the optimal value do not satisfy the stability requirement.

4 Concluding Remarks and Discussions

Regularization in learning can prevent a network from overtraining. Several techniques have been developed in recent years, but all these are specialized for feedforward networks. To our best knowledge, a regularizer for a recurrent network has not been reported previously.

We have developed a smoothing regularizer for recurrent neural networks that captures the dependencies of input, output, and feedback weight values on each other. The regularizer covers both simultaneous and time-lagged recurrent networks, with feedforward networks and single layer, linear networks as special cases. Our smoothing regularizer for linear networks has the same form as standard weight decay. The regularizer developed depends on only the network parameters, and can easily be used.

A series of empirical tests has demonstrated the efficacy of this regularizer and its superior performance relative to standard quadratic weight decay. Empirical results show that the smoothing regularizer yields a real symmetric α-stable ($S\alpha S$) weight distribution, whereas standard quadratic weight decay produces a normal distribution. We therefore believe that our smoothing regularizer provides a more reasonable constraint during training than standard weight decay does. Our regularizer keeps "essential" weights large as needed and, at the same time, makes "non-essential" weights assume values near to zero.

We conclude with several additional comments. As described in Eqn(19), to bound the first derivatives of the activation functions in the hidden units, we have used their maximal value without considering different nonlinearities for different nodes and ignoring their changes with time. We have extended our smoothing regularizer to take into account these factors. Due to the page limit, we cannot include these extensions in this paper.

In the simulations conducted in this paper, we have fully searched over the regularization parameter values by using a validation data set. This helps us observe the effect of the regularization parameter, but it is time

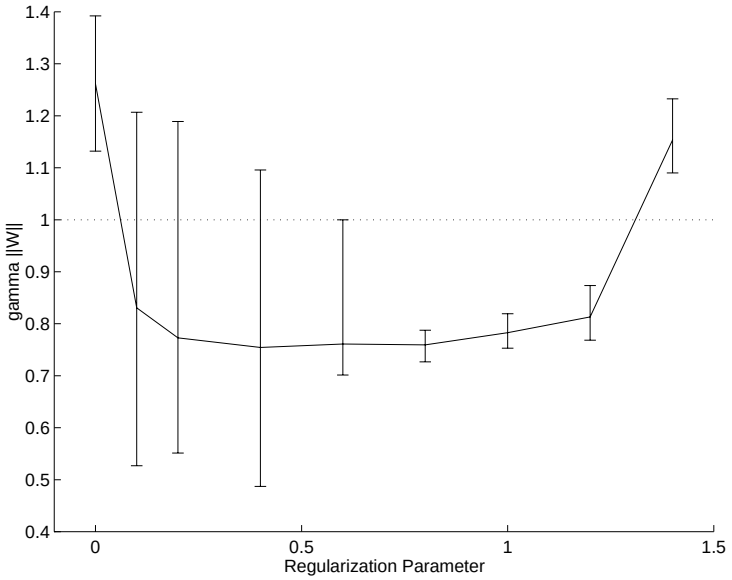

Figure 5: Regularization parameter vs $\gamma||W||$ of trained recurrent networks. The networks are trained to predict the U.S. Index of Industrial Production. For each regularization parameter, 30 networks have been trained. Each network has four hidden units. The smoothing regularizer and early stopping are both used during learning. From Figure 3, we have known the optimal regularization parameter for these networks is 0.8. This figure plots the mean values of $\gamma||W||$ of these 30 networks with the error bars indicating the maximal and minimal values. As shown, the networks with the optimal regularization parameter have $\gamma||W|| < 1$. This confirms the networks' stability, in the sense that the network response to any input perturbation will be smooth.

consuming if the network and the training data set are large.

Stability is another big issue for recurrent neural networks. There is alot of literature on this topic, for example, Hirsch (1989) and Kuan, Hornik & White (1994). In our derivation of the regularizer, we have found that $\gamma||W|| < 1$ must hold to insure the effects of small input perturbations are damped out. This inequality can be used for diagnosing the stability of trained networks as shown in Figure 5. It can also be appended to our training criterion Eqn(17) as an additional constraint.

Werbos & Titus (1978) proposed the following cost function for their pure robust model

$$D = \sum_{t=1}^{P} \sum_{i=1}^{N_o} \left[\frac{z_i(t) - \hat{z}_i(t)}{\sigma_{z_i}(1 - |w_i|)} \right]^2 \quad . \tag{44}$$

In the model, w_i was, in fact, a weight parameter in a feedback connection from the output to the input, but it was pre-defined in the range $0 < w_i < 1$ and kept fixed after being defined. Werbo and Titus's new cost function actually had a similar effect as our smoothing regularizer. As they claimed, the biggest advantage of their new cost function was its ability to shift smoothly in different environments.

A Neglecting the Cross-Covariance

We neglect the cross-covariance term in Eqn (8)

$$\frac{2}{N} \sum_{t=1}^{N} [Z(t) - F(\Phi_P, I(t))][F(\Phi_P, I(t)) - F(\Phi_P, \tilde{I}(t))] \tag{45}$$

for two reasons. First, its expectation value will be small, and secondly, its value can be rigorously bounded with no qualitative change in our proposed training objective function in Eqn (9).

Noting that the target noise ε^* is uncorrelated with the input perturbations ε_x and assuming that model bias can be neglected, the expectation value of Eqn (45) taken over possible training sets will be small:

$$\begin{aligned} \langle Z(t) - F(\Phi_P, I(t)) \rangle &= \langle F^*(I(t)) - F(\Phi_P, I(t)) \rangle + \langle \varepsilon^*(t) \rangle \\ &= \text{Model Bias} + 0 \approx 0 \quad . \end{aligned} \tag{46}$$

Note here that Φ_P are the weights obtained after training.

In addition, the expectation value of Eqn (45) taken over the input perturbations ε_x will be zero to first order in the ε_x:

$$\langle F(\Phi_P, \tilde{I}(t)) - F(\Phi_P, I(t)) \rangle \approx \left\langle \sum_{s=1}^{t} \frac{\partial F(\Phi_P, I(t))}{\partial X(s)} \varepsilon_x(s) \right\rangle = 0 \quad . \tag{47}$$

Of course, many nonlinear dynamical systems have positive Lyapunov exponents, and so the second order and higher order effects in these cases can't be ignored.

Although its expectation value will be small, the cross covariance term in Eqn (8) can be rigorously bounded. Using the inequality $2ab \leq a^2 + b^2$, we obtain:

$$\begin{aligned} & \frac{2}{N} \sum_{t=1}^{N} [Z(t) - F(\Phi_P, I(t))][F(\Phi_P, I(t)) - F(\Phi_P, \tilde{I}(t))] \\ \leq \quad & \frac{1}{N} \sum_{t=1}^{N} [Z(t) - F(\Phi_P, I(t))]^2 + \frac{1}{N} \sum_{t=1}^{N} [F(\Phi_P, I(t)) - F(\Phi_P, \tilde{I}(t))]^2 \\ = \quad & D_P + \frac{1}{N} \sum_{t=1}^{N} [F(\Phi_P, I(t)) - F(\Phi_P, \tilde{I}(t))]^2 \, . \end{aligned} \tag{48}$$

Minimizing the first term D_P and the second term $\frac{1}{N}\sum_{t=1}^{N}[F(\Phi_P, I(t)) - F(\Phi_P, \tilde{I}(t))]^2$ in Eqn (8) during training will thus automatically decrease the effect of the cross-covariance term. Using this bound, instead of the small expectation value approximation, will in effect multiply the first two terms in Eqn (8) by a factor of 2. However, this amounts to an irrelevant scaling factor and can be dropped. Thus, our proposed training objective function Eqn (9) will remain unchanged.

B Output Sensitivity of a Trained Network to its Input Perturbation

For a recurrent network of form given by Eqn (15):

$$Y(t) = \mathbf{f}\left(WY(t-\tau) + VX(t)\right) \ , \ \hat{Z}(t) = UY(t) \ , \tag{49}$$

this appendix studies the output perturbation[12]

$$\sigma_{\hat{z}}^2(t) = \| \ \hat{\tilde{Z}}(t) - \hat{Z}(t) \ \|^2 \tag{50}$$

in response to an input perturbation

$$\sigma_x^2(t) = \| \ \tilde{X}(t) - X(t) \ \|^2. \tag{51}$$

The output perturbation will depend on the weight parameter matrices U, V and W. The sizes of the U, V and W are $N_o \times N_h$, $N_h \times N_h$ and $N_h \times N_i$. The numbers of output, hidden and input units are N_o, N_h and N_i respectively.

By expressing the inputs to the hidden units as an N_h-dimensional column vector

$$O(t) = [o_1(t), \ldots, o_{N_h}(t)]^T = WY(t-\tau) + VX(t) \tag{52}$$

and using the mean value theorem, [13] we get

$$\mathbf{f}(\tilde{O}(t)) - \mathbf{f}(O(t)) = \mathbf{f}'(O^*(t))(\tilde{O}(t) - O(t)) \ , \tag{53}$$

where $\mathbf{f}(O(t)) = [f_1(o_1(t)), \ldots, f_{N_h}(o_{N_h}(t))]^T$, $\mathbf{f}(\tilde{O}(t)) = [f_1(\tilde{o}_1(t)), \ldots, f_{N_h}(\tilde{o}_{N_h}(t))]^T$ and $\mathbf{f}'(O^*(t))$ is a diagonal matrix with elements $[\mathbf{f}'(O^*(t))]_{jj} = {f_j}'(o_j^*(t))$. ${f_j}'(\ )$ is the first derivative of $f_j(\ )$ and $min(\tilde{o}_j(t), o_j(t)) \leq o_j^*(t) \leq max(\tilde{o}_j(t), o_j(t))$.

With Schwarz's inequality, the output disturbance can be expressed as:

$$\begin{aligned} \sigma_{\hat{z}}^2(t) &= \| \ U\mathbf{f}(\tilde{O}(t)) - U\mathbf{f}(O(t)) \ \|^2 \\ &\leq \gamma^2 \| \ U \ \|^2 \| \ \tilde{O}(t) - O(t) \ \|^2 \end{aligned} \tag{54}$$

where

$$\gamma = \max_{t,j} \mid f_j'(o_j^*(t)) \mid \ . \tag{55}$$

For a feedforward network, $O(t) = VX(t)$, we obtain

$$\sigma_{\hat{z}}^2(t) \leq (\gamma ||U|| ||V||)^2 \sigma_x^2(t) \ . \tag{56}$$

We now consider the case of recurrent networks. A recurrent network usually satisfies the following dynamic function: [14]

$$\tau \frac{dO(t)}{dt} = W\mathbf{f}(O(t)) - O(t) + VX(t+\tau), \tag{57}$$

[12] The time varying quantities in Eqns (50) and (51) should not be confused with their associated ensemble averages σ_z^2 and σ_x^2 defined in Section 2.

[13] See, for example, Korn & Korn (1968). Also, assume that all $f_j(\ )$ are continuous and continuously differentiable.

[14] We can obtain $O(t+\tau) = WY(t) + VX(t+\tau)$ and $Y(t) = \mathbf{f}(O(t))$ by substituting the following approximation into Eqn(57):

$$\frac{dO(t)}{dt} \approx \frac{O(t+\tau) - O(t)}{\tau} \ .$$

Here, we assume that τ is small. Note that such a dynamic function has also been used to describe the evolution process of recurrent networks by other researchers, for example, Pineda (1988) and Pearlmutter (1989).

where $\frac{dO(t)}{dt} = [\frac{do_1(t)}{dt}, \ldots, \frac{do_{N_h}(t)}{dt}]^T$. If we define

$$\sigma_o^2(t) = \| \tilde{O}(t) - O(t) \|^2, \tag{58}$$

then

$$\frac{d\sigma_o^2(t)}{dt} = 2\big[\tilde{O}(t) - O(t)\big]^T \left[\frac{d\tilde{O}(t)}{dt} - \frac{dO(t)}{dt}\right]. \tag{59}$$

With Eqn(57) and by assuming $\tau > 0$

$$\frac{d\tilde{O}(t)}{dt} - \frac{dO(t)}{dt} = \frac{1}{\tau}\{W\left[\mathbf{f}(\tilde{O}(t)) - \mathbf{f}(O(t))\right] - \left[\tilde{O}(t) - O(t)\right] + V\left[\tilde{X}(t+\tau) - X(t+\tau)\right]\}. \tag{60}$$

We get

$$\begin{aligned}\frac{d\sigma_o^2(t)}{dt} &= \frac{2}{\tau}\{\left[\tilde{O}(t) - O(t)\right]^T W\left[\mathbf{f}(\tilde{O}(t)) - \mathbf{f}(O(t))\right] \\ &\quad -\| \tilde{O}(t) - O(t) \|^2 + \left[\tilde{O}(t) - O(t)\right]^T V\left[\tilde{X}(t+\tau) - X(t+\tau)\right]\} \ .\end{aligned} \tag{61}$$

Using the mean value theorem and Schwarz's inequality again, we obtain the following equations

$$\| \left[\tilde{O}(t) - O(t)\right]^T W\left[\mathbf{f}(\tilde{O}(t)) - \mathbf{f}(O(t))\right] \| \leq \ \gamma||W||\sigma_o^2(t) \tag{62}$$

for the first term in the right hand side of Eqn(61) and

$$\| \left[\tilde{O}(t) - O(t)\right]^T V\left[\tilde{X}(t+\tau) - X(t+\tau)\right] \| \leq \ \sigma_o(t)||V||\sigma_x(t+\tau) \tag{63}$$

for the third term of Eqn(61).

During the evolution process of the network, the input perturbation $\sigma_x(t)$ is assumed to be constant or to change more slowly than $\sigma_o(t)$. This is true when τ is small. [15] Therefore, the $\sigma_x(t)$ is replaced by σ_x in the following derivation. With Eqns (62) and (63), Eqn(61) becomes

$$\frac{d\sigma_o^2(t)}{dt} \leq \frac{2}{\tau}\left[\gamma||W||\sigma_o^2(t) - \sigma_o^2(t) + \sigma_o(t)||V||\sigma_x\right] \tag{64}$$

or

$$\frac{d\sigma_o(t)}{dt} \leq \frac{1}{\tau}\left[\left(\gamma||W|| - 1\right)\sigma_o(t) + ||V||\sigma_x\right] \tag{65}$$

due to $\sigma_o(t) > 0$. For notational clarity, define

$$a = \gamma||W|| - 1, \ \text{ and } \ b = ||V|| \ , \tag{66}$$

so that Eqn(65) becomes

$$\frac{d\sigma_o(t)}{dt} \leq \frac{1}{\tau}\left[a\sigma_o(t) + b\sigma_x\right] \ . \tag{67}$$

Integration of Eqn(67) from $t-1$ to t yields the solutions

$$\sigma_o(t) \leq \left[\sigma_o(t-1) + \frac{b\sigma_x}{a}\right]\exp\left(\frac{a}{\tau}\right) - \frac{b\sigma_x}{a} \quad \text{for } \ a \neq 0 \ ; \tag{68}$$

$$\sigma_o(t) \leq \sigma_o(t-1) + \frac{b\sigma_x}{\tau} \quad \text{for } \ a = 0 \ . \tag{69}$$

One sees that $\sigma_o(t)$ depends on the current input perturbation σ_x as well as its previous value $\sigma_o(t-1)$. $\sigma_o(t-1)$ again depends on its previous values, so the current $\sigma_o(t)$ is dependent on its all previous input

[15] See Footnote 6 in Section 2 for justification.

perturbations and the whole evolution process of the network learning. One also sees, to insure stability and that the effects of small input perturbations are damped out, the following inequality is required:

$$a < 0 \quad \text{or equivalently} \quad \gamma||W|| < 1 \ . \tag{70}$$

By replacing $\sigma_x(t)$ back, we can rewrite Eqn (68) as

$$\sigma_o(t) \leq \sigma_o(t-1) \exp\left(\frac{a}{\tau}\right) + \frac{b}{a}\left[\exp\left(\frac{a}{\tau}\right) - 1\right] \sigma_x(t) \ . \tag{71}$$

The first term in the right hand side is the zero-input response (when $\sigma_x(t) = 0$) and the second term is the zero-memory response (when $\sigma_o(t-1) = 0$). If we can minimize the zero-memory response at every time step t, $\sigma_o(0), \sigma_o(1), \ldots, \sigma_o(t-1)$ will all be small. Moreover, due to its monotonically decreasing response function, the zero-input response will damp out. Therefore, the zero-input response can be ignored and we can focus only on how to minimize the zero-memory response of $\sigma_o(t)$.

The zero-memory response of $\sigma_o(t)$ in Eqns (69) and (71) becomes

$$\sigma_o(t) \leq \frac{b\sigma_x(t)}{a}\left[\exp\left(\frac{a}{\tau}\right) - 1\right] \quad \text{for} \ \ a \neq 0 \ ; \tag{72}$$

$$\sigma_o(t) \leq \frac{b\sigma_x(t)}{\tau} \quad \text{for} \ \ a = 0 \ . \tag{73}$$

Substituting Eqns (72) and (73) into Eqn(54) along with the definitions of a and b in Eqn(66), we obtain

$$\sigma_{\hat{z}}^2(t) \leq \left\{\frac{\gamma||U||||V||}{1-\gamma||W||}\left[1 - \exp\left(\frac{\gamma||W|| - 1}{\tau}\right)\right]\right\}^2 \sigma_x^2(t) \quad \text{for} \ \ \gamma||W|| \neq 1 \ ; \tag{74}$$

$$\sigma_{\hat{z}}^2(t) \leq \left\{\frac{\gamma||U||||V||}{\tau}\right\}^2 \sigma_x^2(t) \quad \text{for} \ \ \gamma||W|| = 1 \ . \tag{75}$$

Figure 6 plots the function:

$$G_\tau(\gamma||W||) = \frac{1 - \exp\left(\frac{\gamma||W|| - 1}{\tau}\right)}{1 - \gamma||W||} \tag{76}$$

with $\gamma||W|| < 1$ and $\tau = 0.1, 0.5, 1, 2$. The figure depicts the effect of $\gamma||W||$ and τ to the regularizer. As shown, the regularizer becomes more and more sensitive to the change of recurrent weights as the time delay τ decreases. When $\tau \mapsto 0$, $\sigma_{\hat{z}}^2(t)$ is bounded by:

$$\sigma_{\hat{z}}^2(t) \leq \left\{\frac{\gamma||U||||V||}{1-\gamma||W||}\right\}^2 \sigma_x^2(t) \quad \text{for} \ \ \gamma||W|| < 1 \ . \tag{77}$$

When $W = 0$ and $\tau = 0$, the model becomes a feedforward network and the deduced form of Eqns (74) and (77) with $W = 0$ and $\tau = 0$ is the same as Eqn(56). Therefore, the forms for feedforward networks and single-layer, linear networks can also be expressed by Eqns (74) or (77), as special cases.

By defining

$$\rho_\tau(\Phi) = \frac{\gamma \parallel U \parallel\parallel V \parallel}{1 - \gamma \parallel W \parallel}\left[1 - \exp\left(\frac{\gamma||W|| - 1}{\tau}\right)\right] \ , \tag{78}$$

$$\rho(\Phi) \equiv \rho_0(\Phi) = \frac{\gamma \parallel U \parallel\parallel V \parallel}{1 - \gamma \parallel W \parallel} \tag{79}$$

we obtain

$$\sigma_{\hat{z}}^2(t) \leq {\rho_\tau}^2(\Phi)\sigma_x^2(t) \tag{80}$$

$$\leq \rho^2(\Phi)\sigma_x^2(t) \ . \tag{81}$$

Therefore, the network output disturbance $\sigma_{\hat{z}}^2(t)$ to its input perturbation $\sigma_x^2(t)$ can be approximated by Eqns (80) and (81). This concludes the derivation of Eqns (17), (18) and (21).

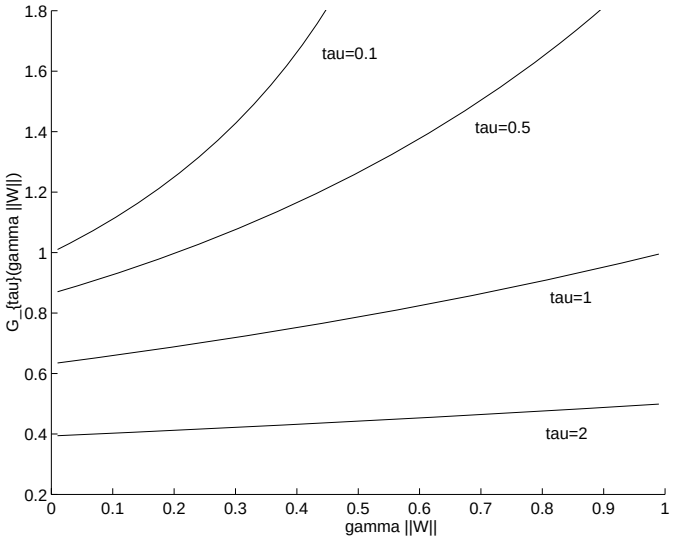

Figure 6: Change of $G_\tau(\gamma||W||)$ as the function of $\gamma||W||$ and τ. $G_\tau(\gamma||W||)$ is defined by Eqn(76).

Acknowledgements

We would like to thank T. Leen and the reviewers for their valuable comments on our first manuscript, T. Rögnvaldsson, S. Rehfuss and Laiwan Chan for their proofreading our revised manuscript, and F. Pineda of Johns Hopkins University for discussing Eqn(57) with us. We also thank the other members of the Neural Network Research Group at OGI for their various suggestions.

References

Abu-Mostafa, Y. (1995), 'Hints', *Neural Computation* **7**(4), 639–671.

Amari, S. (1972), 'Characteristics of random nets of analog neural-like elements', *IEEE Transactions on Systems, Man and Cybernetics* **SMC-2**, 643–653.

Bishop, C. (1993), 'Curvature-driven smoothing: a learning algorithm for feedforward networks', *IEEE Transactions on Neural Networks* **4**(5), 882–884.

Bishop, C. (1995), 'Training with noise is equivalent to Tikhonov regularization', *Neural Computation* **7**(1), 108–116.

Chauvin, Y. (1990), Dynamic behavior of constrained back-propagation networks, *in* D. Touretzky, ed., 'Advances in Neural Information Processing Systems 2', Morgan Kaufmann Publishers, San Francisco, CA, pp. 642–649.

Elman, J. (1990), 'Finding structure in time', *Cognition Science* **14**, 179–211.

Eubank, R. L. (1988), *Spline Smoothing and Nonparametric Regression*, Marcel Dekker, Inc.

Geman, S., Bienenstock, E. & Doursat, R. (1992), 'Neural networks and the bias/variance dilemma', *Neural Computation* **4**(1), 1–58.

Girosi, F., Jones, M. & Poggio, T. (1995), 'Regularization theory and neural networks architectures', *Neural Computation* **7**, 219–269.

Grossberg, S. (1969), 'On learning and energy-entropy dependence in recurrent and nonrecurrent signed networks', *Journal of Statistical Physics* **1**, 319–350.

Hastie, T. J. & Tibshirani, R. J. (1990), *Generalized Additive Models*, Vol. 43 of *Monographs on Statistics and Applied Probability*, Chapman and Hall.

Hirsch, M. (1989), 'Convergent activation dynamics in continuous time networks', *Neural Networks* **2**(5), 331–349.

Hoerl, A. & Kennard, R. (1970*a*), 'Ridge regression: applications to nonorthogonal problems', *Technometrics* **12**, 69–82.

Hoerl, A. & Kennard, R. (1970*b*), 'Ridge regression: biased estimation for nonorthogonal problems', *Technometrics* **12**, 55–67.

Hopfield, J. (1984), 'Neurons with graded response have collective computational properties like those of two-state neurons', *Proceedings of the National Academy of Science, USA* **81**, 3088–3092.

Korn, G. & Korn, T., eds (1968), *Mathematical Handbook for Scientists and Engineers*, McGraw-Hill Book Company.

Kuan, C., Hornik, K. & White, H. (1994), 'A convergence result for learning in recurrent neural networks', *Neural Computation* **6**(3), 420–440.

Leen, T. (1995), From data distributions to regularization in invariant learning, To appear in *Neural Computation*, 1995.

Levin, A. U., Leen, T. K. & Moody, J. E. (1994), Fast pruning using principal components, *in* J. Cowan, G. Tesauro & J. Alspector, eds, 'Advances in Neural Information Processing Systems 6', Morgan Kaufmann Publishers, San Francisco, CA.

Marcus, C. & Westervelt, R. (1989), Dynamics of analog neural networks with time delay, *in* D. Touretzky, ed., 'Advances in Neural Information Processing Systems 1', Morgan Kaufmann Publishers, San Francisco, CA.

Moody, J. E. & Utans, J. (1992), Principled architecture selection for neural networks: Application to corporate bond rating prediction, *in* J. E. Moody, S. J. Hanson & R. P. Lippmann, eds, 'Advances in Neural Information Processing Systems 4', Morgan Kaufmann Publishers, San Mateo, CA, pp. 683–690.

Moody, J. E. & Yarvin, N. (1992), Networks with learned unit response functions, *in* J. E. Moody, S. J. Hanson & R. P. Lippmann, eds, 'Advances in Neural Information Processing Systems 4', Morgan Kaufmann Publishers, San Mateo, CA, pp. 1048–55.

Moody, J. & Rögnvaldsson, T. (1995), Smoothing regularizers for feed-forward neural networks, Oregon Graduate Institute Computer Science Dept. Technical Report, submitted for publication, 1995.

Moody, J., Levin, U. & Rehfuss, S. (1993), 'Predicting the U.S. index of industrial production', *In proceedings of the 1993 Parallel Applications in Statistics and Economics Conference, Zeist, The Netherlands. Special issue of* Neural Network World **3**(6), 791–794.

Nowlan, S. & Hinton, G. (1992), 'Simplifying neural networks by soft weight-sharing', *Neural Computation* **4**(4), 473–493.

Pearlmutter, B. (1989), 'Learning state space trajectories in recurrent neural networks', *Neural Computation* **1**(2), 261–269.

Pineda, F. (1988), 'Dynamics and architecture for neural computation', *Journal of Complexity* **4**, 216–245.

Pineda, F. (1989), 'Recurrent backpropagation and the dynamical approach to adaptive neural computation', *Neural Computation* **1**(2), 161–172.

Plaut, D., Nowlan, S. & Hinton, G. (1986), Experiments on learning by back propagation, Technical Report CMU-CS-86-126, Carnegie-Mellon University.

Poggio, T. & Girosi, F. (1990), 'Networks for approximation and learning', *IEEE Proceedings* **78**(9).

Powell, M. (1987), Radial basis functions for multivariable interpolation: a review., *in* J. Mason & M. Cox, eds, 'Algorithms for Approximation', Clarendon Press, Oxford.

Rumelhart, D., Hinton, G. & Williams, R. (1986), Learning internal representations by error propagation, *in* D. Rumelhart & J. McClelland, eds, 'Parallel Distributed Processing: Exploration in the microstructure of cognition', MIT Press, Cambridge, MA, chapter 8, pp. 319–362.

Scalettar, R. & Zee, A. (1988), Emergence of grandmother memory in feed forward networks: learning with noise and forgetfulness, *in* D. Waltz & J. Feldman, eds, 'Connectionist Models and Their Implications: Readings from Cognitive Science', Ablex Pub. Corp.

Sejnowski, T. (1977), 'Storing covariance with nonlinearly interacting neurons', *Journal of Mathematical Biology* **4**, 303–321.

Shao, M. & Nikias, C. (1993), 'Signal processing with fractional lower order moments: Stable processes and their applications', *IEEE Proceedings* **81**(7), 986–1010.

Sjöberg, J. & Ljung, L. (1992), Overtraining, regularization and searching for minimum in neural nets, *in* 'Preprint 4th IFAC symposium on Adaptive Systems in Control and Signal Processing', pp. 669–674.

Sjöberg, J. & Ljung, L. (1995), Overtraining, regularization and searching for minimum with application to neural nets, To appear in *International Journal of Control*.

Tikhonov, A. N. & Arsenin, V. I. (1977), *Solutions of Ill-posed Problems*, Winston ; New York : distributed solely by Halsted Press. Scripta series in mathematics. Translation editor, Fritz John.

Tresp, V., Hollatz, J. & Ahmad, S. (1993), Network structuring and training using rule-based knowledge, *in* S. J. Hanson, J. D. Cowan & C. L. Giles, eds, 'Advances in Neural Information Processing Systems 4', Morgan Kaufmann Publishers, San Mateo, CA, pp. 871 – 878.

Wahba, G. (1990), *Spline models for observational data*, CBMS-NSF Regional Conference Series in Applied Mathematics.

Weigend, A., Rumelhart, D. & Huberman, B. (1990), Back-propagation, weight-elimination and time series prediction, *in* T. Sejnowski, G. Hinton & D. Touretzky, eds, 'Proceedings of the connectionist models summer school', Morgan Kaufmann Publishers, San Mateo, CA, pp. 105–116.

Werbos, P. (1992), Neurocontrol and supervised learning: An overview and evaluation, *in* D. White & D. Sofge, eds, 'Handbook of Intelligent Control', Van Nostrand Reinhold, New York.

Werbos, P. & Titus, J. (1978), 'An empirical test of new forecasting methods derived from a theory of intelligence: the prediction of conflict in Latin America', *IEEE Transactions Systems, Man & Cybernetics* **SMC-8**(9), 657–666.