January 1996

Smoothing regularizers for projective basis function networks

John E. Moody

Thorsteinn S. Ragnvaldson

OGI CSE Technical Report 96-006

Smoothing Regularizers for Projective Basis Function Networks

John E. Moody[1] and Thorsteinn S. Rögnvaldsson[2]

Dept. of Computer Science and Engineering
Oregon Graduate Institute of Science and Technology
P.O. Box 91000 Portland, Oregon 97291-1000, U.S.A.

Abstract:
Smoothing regularizers for radial basis functions have been studied extensively, but no general smoothing regularizers for *projective basis functions* (PBFs), such as the widely-used sigmoidal PBFs, have heretofore been proposed. We derive new classes of algebraically-simple m^{th}-order smoothing regularizers for networks of projective basis functions $f(W,x) = \sum_{j=1}^{N} u_j g\left[\boldsymbol{x}^T \boldsymbol{v}_j + v_{j0}\right] + u_0$, with general transfer functions $g[\cdot]$. These regularizers are:

$$R_G(W,m) = \sum_{j=1}^{N} u_j^2 \|\boldsymbol{v}_j\|^{2m-1} \qquad \text{Global Form}$$

$$R_L(W,m) = \sum_{j=1}^{N} u_j^2 \|\boldsymbol{v}_j\|^{2m} \qquad \text{Local Form} \quad .$$

With appropriate constant factors, these regularizers bound the corresponding m^{th}-order smoothing integral

$$S(W,m) = \int d^D \boldsymbol{x} \boldsymbol{\Omega}(\boldsymbol{x}) \left\| \frac{\partial^m f(W,\boldsymbol{x})}{\partial \boldsymbol{x}^m} \right\|^2 \quad .$$

In the above expressions, $\{\boldsymbol{v}_j\}$ are the projection vectors, W denotes all the network weights $\{u_j, u_0, \boldsymbol{v}_j, v_0\}$, and $\boldsymbol{\Omega}(x)$ is a weighting function (not necessarily the input density) on the D-dimensional input space. The global and local cases are distinguished by different choices of $\boldsymbol{\Omega}(x)$.

These simple algebraic forms $R(W,m)$ enable the direct enforcement of smoothness without the need for costly Monte Carlo integrations of $S(W,m)$. The regularizers are tested on illustrative sample problems and compared to quadratic weight decay. The new regularizers are shown to yield better generalization errors than weight decay when the implicit assumptions in the latter are wrong. Unlike weight decay, the new regularizers distinguish between the roles of the input and output weights and capture the interactions between them.

[1] moody@cse.ogi.edu
[2] denni@cse.ogi.edu

1 Introduction: What is the right bias?

Regularization is a technique for reducing prediction risk by balancing model bias and model variance. A regularizer $R(W)$ imposes prior constraints on the network parameters W. Using squared error as the most common example, the objective functional that is minimized during training is

$$E = \frac{1}{2M} \sum_{i=1}^{M} [y^{(i)} - f(W, \boldsymbol{x}^{(i)})]^2 + \lambda R(W) \quad , \tag{1}$$

where $y^{(i)}$ are target values corresponding to the inputs $\boldsymbol{x}^{(i)}$, M is the number of training patterns, and the regularization parameter λ controls the importance of the prior constraints relative to the fit to the data. Several approaches can be applied to estimate λ (see for example Eubank (1988), Hastie & Tibshirani (1990) or Wahba (1990)) in order to minimize the prediction risk by optimizing the bias/variance tradeoff.

Regularization reduces model variance at the cost of introducing some model bias. An important question arises: What is the right bias? (Geman, Bienenstock & Doursat 1992). A good choice of the regularizer $R(W)$ will result in lower expected prediction error than will a poor choice.[3]

Weight decay is often used effectively, but it is an *ad hoc* technique that controls weight values rather than the fit to the data directly. It is thus not necessarily optimal. Weight decay is not appropriate for arbitrary function parameterizations, since it will give very different results, depending upon whether a function is parameterized, for example, as $f(w, x)$ or as $f(w^{-1}, x)$.

Since many real world problems are intrinsically smooth, we propose that an appropriate bias to impose is to favor solutions with low m^{th}-order curvature. Direct penalization of curvature is a parametrization-independent approach. The desired regularizer is the standard D dimensional curvature functional of order m:

$$S(W, m) = \int d^D x \boldsymbol{\Omega}(\boldsymbol{x}) \left\| \frac{\partial^m f(W, \boldsymbol{x})}{\partial \boldsymbol{x}^m} \right\|^2 \quad . \tag{2}$$

Here $\| \; \|$ denotes the ordinary euclidean tensor norm and $\partial^m / \partial \boldsymbol{x}^m$ denotes the m^{th} order differential operator. The weighting function $\boldsymbol{\Omega}(\boldsymbol{x})$ ensures that the integral converges and determines the region over which we require the function to be smooth. $\boldsymbol{\Omega}(\boldsymbol{x})$ is not required to be equal to the input density $p(\boldsymbol{x})$, and will most often be different.

The use of smoothing functionals $S(W)$ like (2) has been extensively studied for smoothing splines (Eubank 1988, Hastie & Tibshirani 1990, Wahba 1990) and for radial basis function (RBF) networks (Powell 1987, Poggio & Girosi 1990, Girosi, Jones & Poggio 1995). However, no general class of smoothing regularizers that directly enforce smoothness $S(W, m)$ for *projective basis functions* (PBFs), such as the widely used sigmoidal PBFs, has hitherto been proposed.

Since explicit enforcement of smoothness using (2) requires costly, impractical Monte-Carlo integrations,[4] we derive new, algebraically-simple regularizers $R(W, m)$ that tightly bound $S(W, m)$. In this paper, we focus on the ubiquitous PBF networks. We derive and test the corresponding regularizers for RBFs in the companion paper (Moody & Rögnvaldsson 1996).

[3] Regularizers can be viewed as being part of a more general class of biases called "hints" (Abu-Mostafa 1995). Hints can include soft or hard constraints that enforce symmetries, positivity, monotonicity, smoothness, and so on.

[4] Note that (2) is not just one integral, but actually $\mathcal{O}(D^m)$ integrals, since the norm of the operator $\partial^m / \partial x^m$ has $\mathcal{O}(D^m)$ terms. This is extremely expensive to compute for large D or large m.

2 Derivation of Simple Regularizers from Smoothing Functionals

We consider single hidden layer networks with D input variables, N_h nonlinear hidden units, and N_o linear output units. For clarity, we set $N_o = 1$, and drop the subscript on N_h (the derivation is trivially extended to the case $N_o > 1$). Thus, our network function is

$$f(\boldsymbol{x}) = \sum_{j=1}^{N} u_j g[\theta_j, \boldsymbol{x}] + u_0 \tag{3}$$

where $g[\cdot]$ are the nonlinear transfer functions of the internal hidden units, $\boldsymbol{x} \in R^D$ is the input vector[5] , θ_j are the parameters associated with internal unit j, and W denotes all parameters in the network.

For regularizers $R(W)$, we will derive *strict upper bounds* for $S(W, m)$. We desire the regularizers to be as general as possible so that they can easily be applied to different network models. Without making any assumptions about $\boldsymbol{\Omega}(\boldsymbol{x})$ or $g(\cdot)$, we have the upper bound

$$S(W, m) \leq N \sum_{j=1}^{N} u_j^2 \int d^D x \boldsymbol{\Omega}(\boldsymbol{x}) \left\| \frac{\partial^m g[\theta_j, \boldsymbol{x}]}{\partial \boldsymbol{x}^m} \right\|^2, \tag{4}$$

which follows from the inequality $\left(\sum_{i=1}^{N} a_i\right)^2 \leq N \sum_{i=1}^{N} a_i^2$. We consider two possible options for the weighting function $\boldsymbol{\Omega}(\boldsymbol{x})$. One is to require *global* smoothness, in which case $\boldsymbol{\Omega}(\boldsymbol{x})$ is a very wide function that covers all relevant parts of the input space (e.g. a very wide gaussian distribution or a constant distribution). The other option is to require *local* smoothness, in which case $\boldsymbol{\Omega}(\boldsymbol{x})$ approaches zero outside small regions around some reference points (e.g. the training data).

2.1 Projective Basis Representations

Projective basis functions (PBFs) are of the form

$$g[\theta_j, \boldsymbol{x}] = g\left[\boldsymbol{x}^T \boldsymbol{v}_j + v_{j0}\right], \tag{5}$$

where $\theta_j = \{\boldsymbol{v}_j, v_{j0}\}$, $\boldsymbol{v}_j = (v_{j1}, v_{j2}, \ldots, v_{jD})$ is the j^{th} projection vector, the vector of weights connecting hidden unit j to the inputs, and v_{j0} is the bias, offset, or threshold. Denoting $z_j(\boldsymbol{x}) \equiv \boldsymbol{x}^T \boldsymbol{v}_j + v_{j0}$, the most commonly used PBFs $g[z]$ are sigmoids, such as tanh$[z]$ and the logistic function $g[z] = (1 + \exp(-z))^{-1}$. Other nonlinear transfer functions $g[z]$ include erf$[z]$, cos$[z]$, and monomials z^p.[6]

For PBFs, expression (4) simplifies to

$$S(W, m) \leq N \sum_{j=1}^{N} u_j^2 \|\boldsymbol{v}_j\|^{2m} I_j(W, m), \tag{6}$$

with

$$I_j(W, m) \equiv \int d^D x \boldsymbol{\Omega}(\boldsymbol{x}) \left(\frac{d^m g[z_j(\boldsymbol{x})]}{dz_j^m} \right)^2 . \tag{7}$$

[5] Throughout, we use small letter boldface to denote vector quantities.

[6] Some examples of the practical application of non-sigmoidal $g[z]$ are presented in Moody & Yarvin (1992). Nonsigmoidal PBFs can perform better than sigmoids for some highly nonlinear problems.

2.2 Global weighting

For the global case, we select a gaussian form for the weighting function

$$\Omega_G(\boldsymbol{x}) = (\sqrt{2\pi}\sigma)^{-D} \exp\left[\frac{-\|\boldsymbol{x}\|^2}{2\sigma^2}\right] \tag{8}$$

and require σ to be large. The gaussian simplifies evaluation of the smoothing integral considerably, since it is both separable and spherically symmetric. Integrating out all dimensions, except the one associated with the projection vector $\boldsymbol{v}_j$, we are left with

$$I_j(W,m) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{\infty} dx e^{-x^2/2\sigma^2} \left(\frac{d^m g[z_j(x)]}{dz_j^m}\right)^2 . \tag{9}$$

If $(d^m g[z]/dz^m)^2$ approaches zero outside a region that is small compared to σ, we can bound (9) by setting the exponential in the integrand equal to unity. This implies

$$I_j(W,m) \leq \frac{I(m)}{\|\boldsymbol{v}_j\|} \text{ with } I(m) \equiv \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{\infty} dz \left(\frac{d^m g[z]}{dz^m}\right)^2 , \tag{10}$$

where equality holds in the global limit $\sigma \to \infty$. Defininging the global regularizer to be

$$R_G(W,m) \equiv \sum_{j=1}^{N} u_j^2 \|\boldsymbol{v}_j\|^{2m-1} , \tag{11}$$

the bound of equation (6) becomes

$$S(W,m) \leq N I(m) R_G(W,m) , \tag{12}$$

where the index G emphasizes the fact that this upper bound is used in the global case of large σ. Since λ absorbs all multiplicative factors, we need only weigh expression (11) into the training objective function.

2.3 Local weighting

For the local case, we consider weighting functions of the general form

$$\Omega_L(\boldsymbol{x}) = \frac{1}{M} \sum_{i=1}^{M} \Omega(\boldsymbol{x}^{(i)}, \sigma) \tag{13}$$

where $\boldsymbol{x}^{(i)}$ are a set of points and $\Omega(\boldsymbol{x}^{(i)}, \sigma)$ is a function such as $(\sqrt{2\pi}\sigma)^{-D} \exp\left[-\|\boldsymbol{x} - \boldsymbol{x}^{(i)}\|^2/2\sigma^2\right]$ that decays rapidly for large $\|\boldsymbol{x} - \boldsymbol{x}^{(i)}\|$. We require that $\lim_{\sigma\to 0} \Omega(\boldsymbol{x}^{(i)}, \sigma) = \delta(\boldsymbol{x} - \boldsymbol{x}^{(i)})$, where $\delta(\cdot)$ is the delta function. Thus, when the $\boldsymbol{x}^{(i)}$ are the training data points, the limiting distribution of (13) is the *empirical distribution*.

In the local limit $\sigma \to 0$, the integral $I_j(W,m)$ of (7) simplifies, and (6) becomes

$$S(W,m) \leq \frac{N}{M} \sum_{j=1}^{N} u_j^2 \|\boldsymbol{v}_j\|^{2m} \sum_{i=1}^{M} \left(\frac{d^m g[z_j(\boldsymbol{x}^{(i)})]}{dz_j^m}\right)^2 . \tag{14}$$

In theory, we could compute the expression within parentheses in (14) for each input pattern $\boldsymbol{x}^{(i)}$ during training and use it as our regularization cost. However, this requires explicit design for each transfer function form and also becomes increasingly complicated as we go to higher m. To construct a simpler and more general form, we instead assume that the m^{th} derivative of the transfer function is bounded and define

$$C_{LP}(m) \equiv \max_z \left(\frac{d^m g[z]}{dz^m} \right)^2 , \tag{15}$$

and

$$R_L(W,m) \equiv \sum_{j=1}^{N} u_j^2 \|\boldsymbol{v}_j\|^{2m} \quad . \tag{16}$$

This gives the bound

$$S(W,m) \leq N C_{LP}(m) R_L(W,m) \tag{17}$$

for the maximum local curvature of the function (the additional index L emphasizes that it is an upper bound in the local limit). We propose using expression (16) as the smoothing regularization term.

3 Simulation Studies

In this section we demonstrate the value of using smoothing regularizers on two simple problems which illustrate two key differences between smoothing and quadratic weight decay. Both problems have few input variables and more internal units than inputs, and are thus not examples of cases where we expect our regularizers to work optimally. However, considering low dimensional problems enables us to do extensive simulation studies.

We use standard sigmoidal PBF networks for both problems, and study the effects of additive noise, sparse data sets, and the choice of order of the smoother. We train the networks using the RPROP algorithm (Riedmiller & Braun 1993), which was empirically found to be the learning algorithm that converged quickest and reached the lowest errors. The training data are i.i.d. for both the inputs and the additive noise in the targets. The input variables are sampled separately, and are hence uncorrelated with each other, and are normalized to have zero mean and unit variance.

The experimental procedure is as follows: First, we scan 8 orders of magnitude of λ, in steps of $\Delta \log 10(\lambda) = 0.5$, in order to find the λ value that gives the lowest geometric[7] mean generalization error. For each value of λ, the mean is computed over 10 networks with different initial conditions and different randomly-generated training sets. An example of such a trace is shown in fig. 1. The generalization error is computed by integrating over a lattice of 201^D points of the noise-free target function. Secondly, we train 100 networks with different initial weights and different training sets, using the λ value that resulted in the lowest generalization error.

The procedure is repeated for each regularization method (local/global smoothers of different orders and weight decay with or without including bias weights). Generalization performances of the different regularization methods are then compared pairwise, using a Wilcoxon rank test[8] (Kendall & Stuart 1972). Two methods are considered significantly different if the null hypothesis (equal average generalization performance) is rejected at the 95% confidence level.

As a base-level comparison, we also fit 100 linear models and 100 unregularized neural networks to test if our regularized networks perform significantly better. In all experiments reported here, the regularized

[7] The generalization errors seem log-normally distributed, which is why we consider the geometric mean.

[8] We also did a paired t-test on the log of the generalization errors. This gave the same results as the Wilcoxon test, and we report only the latter.

networks do significantly better than both a linear model or an unregularized model (using the test criteria described above).

3.1 Sigmoidal Bump

The first problem is the one dimensional "bump problem" suggested by Wu & Moody (1996). The target function is

$$t(x) = 0.5\tanh\left[(x+5)/2\right] - 0.5\tanh\left[(x-5)/2\right], \tag{18}$$

which can be realized with a linear output neural network with one hidden layer of at least two logistic or tanh hidden units. For this problem, we generate training sets $\{(x_i, y_i); i = 1, \ldots, M\}$ of sizes $M \in \{11, 21, 41\}$ with noisy targets $y_i \equiv t(x_i) + \epsilon_i$, randomly sampled from $-10 \leq x \leq 10$, and add gaussian noise ϵ of variance $s^2 \in \{0.1, 0.5, 1.0\}$. These noise levels correspond to signal-to-noise ratios of $\{1.2, 0.55, 0.39\}$, defined as the ratio of the standard deviation of the target function, evaluated over the sampling region, and the standard deviation of the gaussian noise.

Table 1 summarizes our results on this problem using networks with four hidden sigmoidal units (four internal units avoids some problems with local minima that occur when using only two internal units). The smoothing regularizers do significantly better than weight decay when the problems are noisy. However, when the bias weights are excluded from the weight decay, the difference essentially disappears.

3.2 Bilinear Function

The second problem is the two dimensional bilinear function

$$t(x_1, x_2) = x_1 x_2. \tag{19}$$

This example was used by Friedman & Stuetzle (1981) to demonstrate projection pursuit regression. It is the simplest example of a function that has interactions between the input variables. The function can be well-fitted by a one hidden layer network with four sigmoidal hidden units by expressing the function in the form $t(x_1, x_2) = 0.5(x_1 + x_2)^2 - 0.5(x_1 - x_2)^2$ and approximating the quadratic functions with a superposition of two tanh sigmoids.

We generate training sets of sizes $M \in \{20, 40, 100\}$, randomly sampled from the space $-1 \leq \{x_1, x_2\} \leq 1$, and add gaussian noise with standard deviation $s \in \{0.1, 0.2, 0.5\}$, which corresponds to signal-to-noise ratios of $\{3.33, 1.67, 0.67\}$. The student networks have 8 hidden tanh units and one linear output. Figure 1 illustrates, for the special case of $s = 0.2$ and a training set with 40 data points, how the generalization performance improves when higher order smoothing regularizers are used instead of weight decay or first order smoothers, which yield inferior solutions. As shown in table 2, this is true when the network is trained on few data points with lots of noise, as well as when it is trained on many data points with little noise. In contrast to the previous problem, excluding bias weights from the quadratic weight decay term does not help.

3.3 Discussion

Weight decay performs poorly on the sigmoid bump problem because the magnitudes of the required bias weights are very different from the magnitudes of the other weights. The target network function thus fits poorly into the assumption that bias weights and non-bias weights all have the same gaussian prior. However, if weight decay is not applied to the bias weights, the problem fits weight decay much better, and the

Smoothing regularizer	Size of training set	Weight decay			w/out bias		
		Signal/Noise			Signal/Noise		
		0.39	0.55	1.2	0.39	0.55	1.2
Global, $m = 1$	41	+	0	0	0	0	0
	21	+	+	0	0	0	0
	11	0	0	0	0	0	0
Local, $m = 1$	41	+	+	0	0	0	0
	21	+	+	0	0	+	0
	11	0	+	+	0	0	0
Global, $m = 2$	41	+	+	0	+	0	0
	21	+	+	0	0	+	0
	11	0	+	0	0	0	0
Local, $m = 2$	41	+	+	0	+	0	0
	21	+	+	0	0	0	0
	11	–	+	0	–	0	0
Global, $m = 3$	41	+	+	0	+	0	–
	21	+	0	0	0	0	0
	11	0	+	0	–	0	0
Local, $m = 3$	41	+	+	0	+	0	0
	21	+	0	0	0	0	0
	11	0	0	0	0	0	0

Table 1: Pairwise performance comparisons on the one dimensional bump problem defined in eq.(18). A '+' means that use of the smoother results in significantly lower generalization error than weight decay (with or without bias weights). A '–' means that the smoother results in significantly higher generalization error than weight decay. A '0' means that the difference is insignificant (i.e. less than required for rejection of the equal means hypothesis at a 95% confidence level).

performance is improved. The equivalent treatment of bias and non-bias weights is thus not always the proper thing to do (whereas it is often quite reasonable to expect a function to be smooth).

Simply removing the bias weights from the weight decay cost is no "cure for all ills". This is demonstrated in the bilinear problem, where weight decay performs poorly because it lacks any form of interaction between the weights. To fit the bilinear problem, the resulting function must be close to quadratic, which can be done with a neural network of the form

$$f(W, x) = u_0 + u\text{tanh}[vx + v_0] - u\text{tanh}[vx - v_0] \tag{20}$$

by expanding it around $x = 0$, requiring that $u_0 = -2v\text{tanh}[v_0]$ and $uv^2\text{tanh}[v_0](\text{tanh}^2[v_0] - 1) = 1$. This gives

$$f(W, x) = x^2 + \mathcal{O}(x^4 v^2), \tag{21}$$

showing that constructing a good quadratic fit from two sigmoids requires that $v^2 \to 0$ and $u \to \infty$. That is, the connections between inputs and hidden units must approach zero while the connections between hidden units and the output must approach infinity. This is completely contrary to the weight decay assumptions, where there is no interaction between weights in different layers.

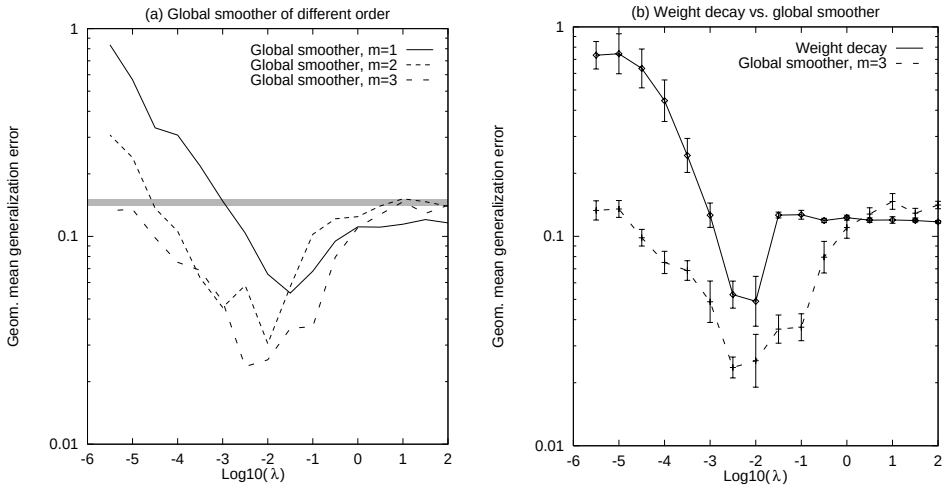

Figure 1: (**a**) Generalization errors on the x_1x_2 problem, with 40 training data points and a signal-to-noise ratio of 2/3, for different values of the regularization parameter and different orders of the smoothing regularizer. For each value of λ, 10 networks have been trained and averaged (geometric average). The best generalization error decreases with increasing order of the smoothing regularizer. The shaded area shows the 95% confidence bands for the average performance of a linear model on the same problem. (**b**) Similar plot for the $m = 3$ smoother compared to the standard weight decay method. Error bars mark the estimated standard deviation of the mean generalization error of the 10 networks.

4 Quality of the Regularizers: Approximations *vs* Bounds

With appropriate multiplicative factors, eqs. (11) and (16) are strict upper bounds to the smoothness functional $S(W, m)$, eq. (2), in the global and local cases. However, questions arise as to how tight the bounds are and how well the regularizers $R(W, m)$ track the curvature functional $S(W, m)$. If the bounds are not sufficiently tight, then penalizing $R(W, m)$ might not have the effect of penalizing $S(W, m)$.

In this section, we discuss the nature of the bounds, present approximations to $S(W, m)$ that are proportional to the bounds, and present empirical results that demonstrate that penalizing $R(W, m)$ does in fact have the effect of penalizing $S(W, m)$ for networks of projective basis units.

Note that for the proposed regularizers $R(W, m)$ to be effective in penalizing $S(W, m)$, we need only have an approximate monotonic relationship between them. In fact, we will argue and demonstrate empirically that an approximate linear relationship between R and S holds.

Ignoring for a moment the effect of a nonzero and finite σ, our derivation involves two steps that can influence the tightness of the bounds. The first is the inequality (4). The second approximations are the bound in (10) for the global form and the bound in (15) for the local form. In the global limit $\sigma \to \infty$, the bound (10) becomes an equality.

Smoothing regularizer	Sample size	Weight decay			w/out bias		
		Signal/Noise			Signal/Noise		
		0.67	1.7	3.3	0.67	1.7	3.3
Global, $m = 1$	100	–	–	–	–	–	–
	40	+	–	–	–	–	–
	20	+	–	–	0	–	–
Local, $m = 1$	100	0	0	–	0	0	–
	40	+	0	0	0	–	–
	20	+	0	–	0	0	–
Global, $m = 2$	100	0	+	+	0	+	+
	40	+	+	+	+	0	0
	20	+	+	+	0	+	+
Local, $m = 2$	100	+	+	+	+	+	+
	40	+	+	+	+	+	+
	20	+	+	+	+	+	+
Global, $m = 3$	100	+	+	+	+	+	+
	40	+	+	+	+	+	+
	20	+	+	+	0	+	+
Local, $m = 3$	100	+	+	+	+	+	+
	40	+	+	+	+	+	+
	20	+	+	+	0	+	+

Table 2: Pairwise performance comparisons on the two dimensional $x_1 x_2$ problem defined in eq.(19). Notation follows table 1.

4.1 The Uncorrelated Hidden Unit Approximation

The inequality (4) should not introduce significant error for problems in which the internal unit activities are *uncorrelated*, since the bound can be replaced by an approximation. A set of functions $a_j(\boldsymbol{x})$ is uncorrelated if:

$$\int d^D x \Omega(\boldsymbol{x}) a_j(\boldsymbol{x}) a_k(\boldsymbol{x}) \approx 0 \quad \text{for} \quad j \neq k \quad . \tag{22}$$

This yields the following approximate relationship:

$$\int d^D x \Omega(\boldsymbol{x}) \left(\sum_{j=1}^{N} a_j(\boldsymbol{x}) \right)^2 \approx \int d^D x \Omega(\boldsymbol{x}) \sum_{j=1}^{N} \left(a_j(\boldsymbol{x}) \right)^2 \quad . \tag{23}$$

Under the uncorrelated internal unit assumption, the bound of equation (12) for the global case can be replaced by the approximation:

$$S_G(W, m) \approx I(m) R_G(W, m) \quad , \tag{24}$$

Note that the right hand sides differs from that in equation (12) only by a factor of N, so this approximation is proportional to the bounds.

In general, it is not always the case that the algebraic forms of an approximation and a strict upper bound differ only by a constant factor. For our regularizers, the constant factor N doesn't matter, since it can be absorbed into the regularization parameter λ (along with the value of the integral $I(m)$). Since λ is selected on a case by case basis, such constant factors are irrelevant. In practical terms then, there is no difference between using the upper bound (12) or the uncorrelated approximation (24).

How good is the uncorrelated internal unit assumption? Our empirical results presented below indicate that it is a good approximation for PBFs when the dimensionality of the input space gets large. The probability of having significant correlation between two internal units decreases exponentially with the input space dimension, and is very small already for moderate numbers of variables. Furthermore, even in low dimensions, the possible positive overlap between internal units decreases for many transfer functions (e.g. sigmoids or gaussians) with increasing order of the derivative. The accuracies of the approximation thus improves with increasing input dimension, and with increasing m. This is a very nice effect, since many real problems deal with many (10 or more) input variables.

4.2 Quality of Other Approximations

For the global case, the bound (10) is fairly tight and approaches equality as σ gets large. For the special case of sigmoidal erf$[\cdot]$ units, $I(W,m)$ can be evaluated exactly, and the approximation error in bound (10) can be shown to be $O(\sigma^{-2})$. This analysis is presented in appendix A. Note that the numerical differences between the popular logistic units and appropriately scaled erf$[\cdot]$ units are small.

For the local case, the approximation error in the bound (15) depends on whether the averages

$$\frac{1}{M}\sum_{i=1}^{M}\left(\frac{d^m g[z(\boldsymbol{x}_j^{(i)})]}{dz^m}\right)^2 \tag{25}$$

for PBFs varies much with j or not. We show in appendix B that they do not vary much.[9] Thus, the local forms behave well, almost as well as the global forms. This analysis is confirmed by extensive simulation results below.

4.3 Empirical Comparisons of $R(W,m)$ *vs* $S(W,m)$

For the regularizers $R(W,m)$ to be effective in penalizing $S(W,m)$, an approximate monotonically-increasing relationship must hold between them. The uncorrelated internal unit assumption implies that this relationship is linear.

To test for such a linear scaling, we generated a large number of randomly selected networks. For each such network, we computed the values of $R(W,m)$ and performed Monte Carlo integrations to compute $S(W,m)$. For each experiment, we fit a linear model

$$R(W,m) = \alpha + \beta S(W,m) \tag{26}$$

to the data, estimating the parameters α and β. The accuracy of the linear scaling is measured by the linear correlation $\langle RS\rangle/\sqrt{(\langle R^2\rangle\langle S^2\rangle)}$. Under the assumption of a linear relationship, the quality of the regularizers can thus be measured. If the linear correlation is high, using the regularizer $R(W,m)$ effectively penalizes the smoothing functional $S(W,m)$.

Figure 2 shows the correlation between the value of the true functional (2) and our regularizers for networks with 10 input units (D) and 10 internal units (N). The value of $S(W,m)$ is estimated through Monte Carlo integration. That is, we sample 10^5 input data patterns from a gaussian distribution with zero mean and unit variance, and replace the integration with a summation over these points. This is repeated 500 times,

[9] Note that for RBFs they can potentially vary a lot, due to different RBF widths. This renders the local forms for RBFs useless. Analysis and simulations are presented Moody & Rögnvaldsson (1996).

picking new random weights each time but keeping the network architecture constant. The network weights are sampled from a uniform distribution of width 10. This ensures that we sample networks that can be very nonlinear.

The correlation is very high for both the global and local forms, although the global form is slightly better. For the RBF case, only the global form is correlated with the true functional. To verify that this finding is not spurious, we repeat our Monte Carlo simulations for several different network architectures, using the same method for sampling weights. These results, which are shown in figure 3, show that the same conclusions hold as the number of hidden or input units increase or decrease. As anticipated, the regularizers are better estimates of $S(W, m)$ when the number of inputs grows or when the order m is increased.

The effect of a nonzero or finite σ depends on the particular choice of weighting function. For a gaussian weighting function, the limiting bounds should be correct to second order (i.e. σ^2 or σ^{-2} depending on the limit taken), which follows from the power series expansion of the gaussian. We show in appendix B that this is correct for a gaussian weighting function $\Omega(x)$ and sigmoidal projective basis units.

5 Weight Decay Type Approaches and Smoothing

5.1 Does Weight Decay Impose Smoothness?

There is no reason to expect a smoothing regularizer to be the best choice of regularizer for all problems, as well as there is no reason to expect any other regularizer to always work better than a smoother. We have therefore refrained from benchmark tests comparing our smoothing regularizers to the extensive list of all hitherto proposed regularization terms, and instead chosen to compare only with quadratic weight decay. There are, however, qualitative differences between smoothing regularization and any conventional type of weight decay cost that are worth noting.

Quadratic weight decay (Plaut, Nowlan & Hinton 1986), which is essentially Hoerl & Kennard's (1970*a*) (1970*b*) "ridge regression" applied to nonlinear models, is the regularization method most often used for both linear regression and neural networks. From a Bayesian viewpoint, weight decay imposes a zero mean, spherically symmetric, gaussian prior on the network weights (Lindley & Smith 1972, Buntine & Weigend 1991). The appropriateness of such an *ad hoc* prior has been questioned in the linear regression case (Smith & Campbell 1980), and it is equally questionable for nonlinear regression (e.g. neural networks).

As an alternative view, it is sometimes argued that quadratic weight decay corresponds to a smoothness constraint. As illustrated in figure 4, this is a misinterpretation if smoothness is measured by functionals of the form (2). Smoothness constraints necessarily create a coupling between the weights in different layers of the network[10], similar to the one described for the bilinear problem. No such coupling exists in quadratic weight decay.

Variations on the theme "weight decay" that have been presented in the neural network literature are e.g. weight elimination (Rumelhart 1988, Scalettar & Zee 1988, Chauvin 1990, Weigend, Rumelhart & Huberman 1990), Laplacian pruning (Williams 1995, Ishikawa 1996), and soft weight sharing (Nowlan & Hinton 1992). These all build on the concept of imposing a prior distribution on the model parameters, and their motivation vis-à-vis quadratic weight decay is that a non-gaussian prior on the parameters is more likely to be correct than a gaussian prior. None of these variations contain any form of coupling between different parameters in the model, and they can therefore not be said to correspond to smoothing.

[10] This is easily verified by constructing a neural network with sigmoidal internal units that reproduces e.g. a linear function.

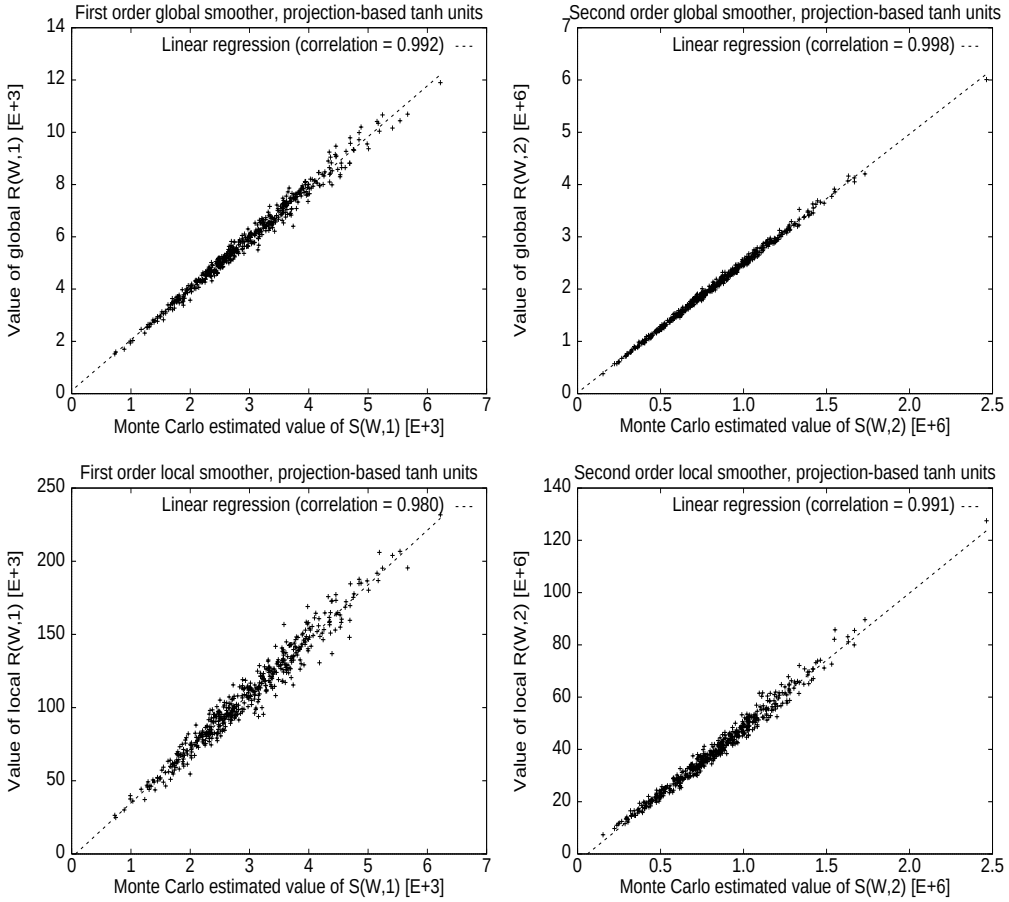

Figure 2: Linear correlation between $S(W, m)$ and $R(W, m)$ for a neural network with 10 input units, 10 internal tanh PBF units, and one linear output. The left column shows results for first order smoothing $(m = 1)$ and the right column shows results for second order smoothing $(m = 2)$. The top row shows the global form of the regularizer $R_G(W, m)$ and the bottom row shows results for the local form $R_L(W, m)$. Note that the correlation coefficients are very close to 1.0, confirming that penalizing $R(W, m)$ effectively penalizes the curvature functional $S(W, m)$.

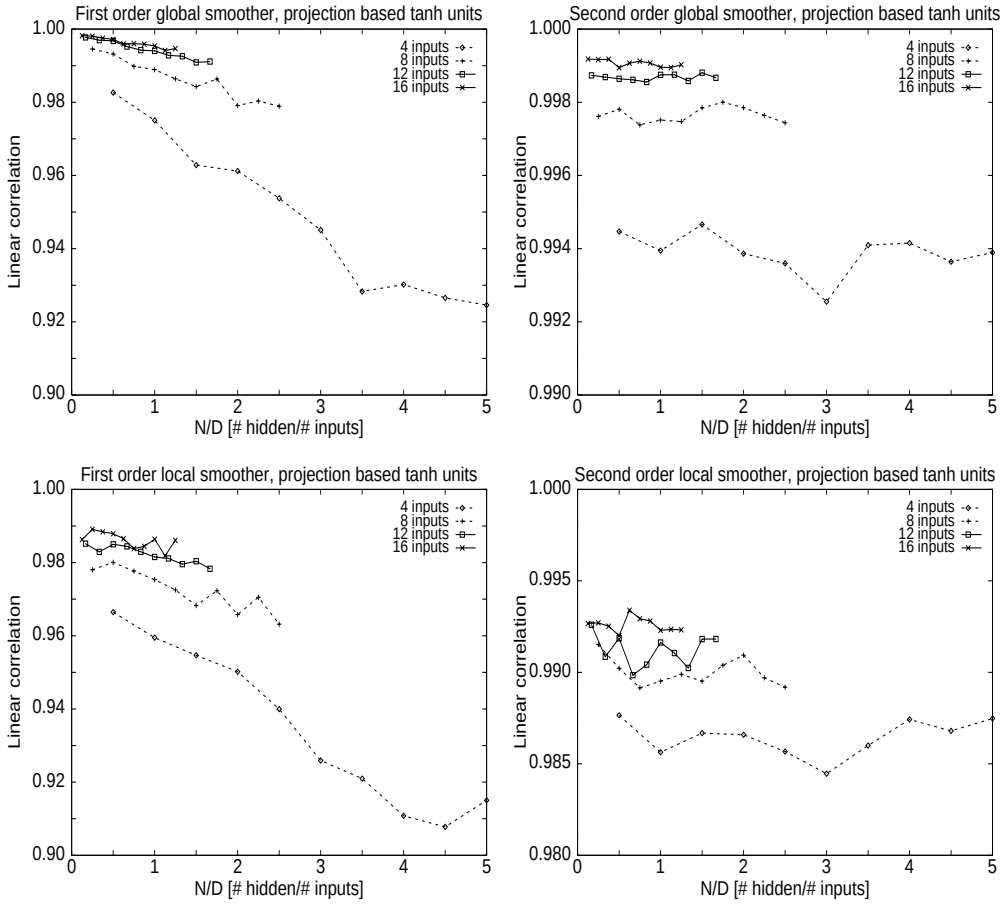

Figure 3: Linear correlation between $S(W, m)$ and $R(W, m)$ for different network architectures, using tanh PBF units. The left column shows results for first order smoothing $(m = 1)$ and the right column shows results for second order smoothing $(m = 2)$. The top row shows the global form of the regularizer $R_G(W, m)$, and the bottom row shows results for the local form $R_L(W, m)$. In both the first and second order cases, the linear correlations are higher for the global than for the local smoothers. In all cases, however, the regularizers $R(W, m)$ are highly correlated with the curvature functionals $S(W, m)$. Thus, $R(W, m)$ is proportional to $S(W, m)$, and penalizing $R(W, m)$ effectively penalizes $S(W, m)$.

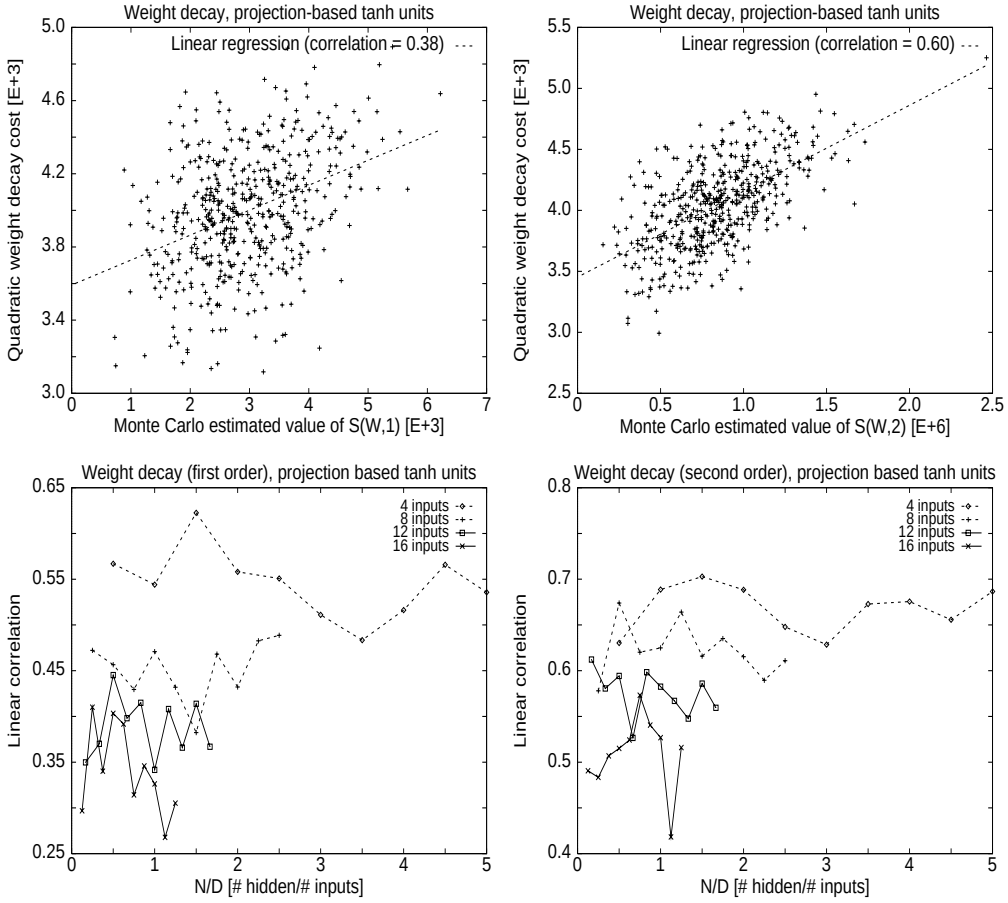

Figure 4: Linear correlation between $S(W, m)$ and the quadratic weight decay cost, using tanh PBF units. The left column shows results for first order smoothness $(m = 1)$ and the right column shows results for second order smoothness $(m = 2)$. The top row shows the result of generating 500 random networks, with 10 input units, 10 internal units, and one linear output, and comparing the Monte Carlo estimated value of $S(W, m)$ to the quadratic weight decay cost. The scatterplots exhibit substantially less correlation than those for the smoothing regularizers $R(W, m)$ shown in figure 2. The bottom row shows how the linear correlation coefficient varies with the network architecture (each point corresponds to 500 random networks). The correlation coefficients are significantly less than those for $R(W, m)$ shown in figure 3. It is quite clear that the weight decay cost $\|W\|^2$ is not a good estimate of $S(W, m)$.

5.2 Other Approaches to Smoothing Neural Networks

Smoothing splines (Eubank 1988, Hastie & Tibshirani 1990, Wahba 1990) and smoothing radial basis functions (Powell 1987, Poggio & Girosi 1990, Girosi *et al.* 1995) impose smoothness by requiring the forms of the hidden units $g[\cdot]$ to be Greens functions of the smoothing operator $S(\cdot)$. This approach can not be extended in a general way to networks of projective basis functions, because of the nature of Green's functions.

Our approach is substantially different from that of smoothing splines and smoothing RBFs, since we derive algebraically-simple smoothing regularizers for general classes of projective units $g[\theta, x]$. Our approach thus has the advantage that it can be applied to the types of networks most often used in practice.

Two proposals for smoothing regularizers in the first order case ($m = 1$) for PBF networks have previously been put forth. Wu & Moody (1996) have derived a smoothing regularizer for both two layer recurrent and feed forward PBF networks, by requiring the model to be robust with respect to small perturbations of the inputs. Van Vuuren (1994) has proposed a form for PBF networks based on a smoothing functional that uses the norm of the gradient, instead of the squared norm as in equation (4). Bishop (1995), Leen (1995), and Wu & Moody (1996) independently pointed out the correspondence between first order smoothing and training with noisy inputs. For second order smoothing ($m = 2$), Bishop (1991) has proposed an algorithm for imposing smoothness on gaussian RBFs during training by using a local smoothness requirement, similar to that discussed in section 2.3.

We are unaware of any previously presented and demonstrated methods for imposing general smoothness requirements of any order ($m = 1, 2, \ldots$ and higher) for PBF networks.

6 Summary

Our regularizers $R(W, m)$ are the first general class of m^{th}–order smoothing regularizers to be proposed for projective basis function networks. The regularizers are simple algebraic forms for smoothing functionals of any order:

$$\begin{aligned} R_G(W, m) &= \sum_{j=1}^{N} u_j^2 \|\boldsymbol{v}_j\|^{2m-1} \qquad \text{Global Form} \\ R_L(W, m) &= \sum_{j=1}^{N} u_j^2 \|\boldsymbol{v}_j\|^{2m} \qquad \text{Local Form} \quad . \end{aligned} \tag{27}$$

These regularizers $R(W, m)$ enable the direct enforcement of smoothness without the need for costly Monte Carlo integrations of the smoothness functional $S(W, m)$ defined in eq. (2). The forms of the regularizers differ fundamentally from quadratic weight decay, in that they distinguish the roles of the input weights $\boldsymbol{v}_j$ and output weights u_j, and capture the interactions between them. Our regularizers apply to PBFs with large classes of transfer functions $g[\cdot]$, including sigmoids.

Monte Carlo simulation results confirm that equations (27) are effective in penalizing the smoothness functional $S(W, m)$. Both the global and local forms are very good, with the global being slightly better.

In a companion paper (Moody & Rögnvaldsson 1996), we derive and test corresponding regularizers for radial basis functions.

Acknowledgements

Both authors thank Steve Rehfuss and Dr. Lizhong Wu for stimulating input. John Moody thanks Volker Tresp for a provocative discussion at a 1991 Neural Networks Workshop sponsored by the Deutsche Informatik Akademie. We gratefully acknowledge support for this work from ARPA and ONR (grant N00014-92-J-4062), NSF (grant CDA-9503968), the Swedish Institute, and the Swedish Research Council for Engineering Sciences (contract TFR-282-95-847).

Appendix A: Evaluation of $I(W,m)$ for Sigmoidal PBFs

For the special case of sigmoidal PBFs in the internal layer of the network, we can use $g(z) = \mathrm{erf}(z)$ to compute the integral $I_j(W,m)$ exactly and compare the behaviour of the smoothing functional with our bounds. With a gaussian weighting function of the form (8), some tedious but straightforward algebra yields the expressions[11]

$$I(W,2n+1) = I(W,1)\sum_{k,l=0}^{n}\frac{(-1)^{k+l}4^{k+l}[(2n)!]^2(2k+2l)!}{(2k)!(2l)!(n-k)!(n-l)!}\left(\frac{v_0}{4\sigma^2v^2+1}\right)^{2(k+l)}\times \sum_{i=0}^{k+l}\frac{1}{(2k+2l-2i)!i!}\left(\frac{\sigma^2v^2(4\sigma^2v^2+1)}{2v_0^2}\right)^{i} \quad \text{(A:1)}$$

$$I(W,2n) = I(W,1)\sum_{k,l=0}^{n-1}\frac{(-1)^{k+l}4^{k+l+1}[(2n-1)!]^2(2k+2l+2)!}{(2k+1)!(2l+1)!(n-k-1)!(n-l-1)!}\left(\frac{v_0}{4\sigma^2v^2+1}\right)^{2(k+l+1)}\times \sum_{i=0}^{k+l+1}\frac{1}{(2k+2l+2-2i)!i!}\left(\frac{\sigma^2v^2(4\sigma^2v^2+1)}{2v_0^2}\right)^{i} \quad \text{(A:2)}$$

where

$$I(W,1) = \frac{4}{\pi\sqrt{4\sigma^2v^2+1}}\exp[-2v_0^2/(4\sigma^2v^2+1)]. \quad \text{(A:3)}$$

In the global limit, $\sigma\to\infty$, we make the approximations

$$\sum_{i=0}^{n}\frac{1}{(2n-2i)!i!}\left(\frac{\sigma^2v^2(4\sigma^2v^2+1)}{2v_0^2}\right)^{i} \approx \frac{1}{n!}\left(\frac{2\sigma^4v^4}{v_0^2}\right)^{n}+\frac{1}{2!(n-1)!}\left(\frac{2\sigma^4v^4}{v_0^2}\right)^{n-1}, \quad \text{(A:4)}$$

$$\frac{1}{(4\sigma^2v^2+1)^n} \approx \frac{1}{(4\sigma^2v^2)^n}\left(1-\frac{n}{4\sigma^2v^2}\right), \quad \text{(A:5)}$$

$$I(W,1) \approx \frac{2}{\pi\sigma v}\left(1-\frac{1+4v_0^2}{8\sigma^2v^2}\right), \quad \text{(A:6)}$$

which require that $\sigma^2v^2 \gg \max[1,v_0^2]$. Inserting this into the exact expressions and keeping terms of order $(\sigma v)^{-2}$ gives

$$I(W,m) \approx \frac{2(2m-3)!!}{\pi\sigma v}\left(1+\frac{1-v_0^2(6m-10)}{8\sigma^2v^2(2m-3)}\right), \quad \text{(A:7)}$$

[11] The index on v_j is dropped.

where the first term corresponds to the global limit regularizer (11), which is thus correct to order $\sigma^{-2}\|v\|^{-2}$.

Note that the local limit, $\sigma \to 0$, can be treated in a similar manner, which gives

$$I(W,m) \approx \exp[-2v_0^2]\left(P_{2m-1}^{(1)}(v_0) + \sigma^2\|v\|^2 P_{2m}^{(2)}(v_0)\right) , \tag{A:8}$$

where $P_n^{(1)}(v_0)$ and $P_n^{(2)}(v_0)$ are two different polynomials of order n. The exponential weighting, and the oscillatory nature of the polynomials, supports our decision to ignore the dependence on v_0. Thus, the local limit can be said to be correct to order $\sigma^2\|v\|^2$.

Appendix B: Quality of the Local Form

To simplify the notation, we define

$$G(m, z_j(\boldsymbol{x})) \equiv \left(\frac{d^m g[z_j(\boldsymbol{x})]}{dz_j^m}\right)^2 \tag{B:1}$$

where

$$z_j(\boldsymbol{x}) \equiv \boldsymbol{x}^T \boldsymbol{v}_j + v_{j0} \tag{B:2}$$

With this definition, the quality of the local form

$$R_L(W,m) = \sum_{j=1}^{N} u_j^2 \|\boldsymbol{v}_j\|^{2m} \tag{B:3}$$

depends upon how much the sum

$$\frac{1}{M}\sum_{i=1}^{M} G(m, z_j(\boldsymbol{x}^{(i)})) \tag{B:4}$$

varies with j. If it varies very little, then the local form is a good approximation. If it varies a lot, then the local form is a poor approximation.

To simplify the evaluation of (B:4), we work in the limit $M \to \infty$, such that

$$\frac{1}{M}\sum_{i=1}^{M} G(m, z_j(\boldsymbol{x}^{(i)})) \to \int d^D x p(\boldsymbol{x}) G(m, z_j(\boldsymbol{x})) = \overline{G}(m,j) \tag{B:5}$$

where $p(\boldsymbol{x})$ is the input data probability density function. To avoid unnecessary complications, we also assume that the input data follow a zero mean gaussian distribution

$$p(\boldsymbol{x}) = \left(\frac{1}{\sigma\sqrt{2\pi}}\right)^D \exp\left[\frac{-\|\boldsymbol{x}\|^2}{2\sigma^2}\right]. \tag{B:6}$$

Sigmoidal Projective Basis Functions

Instead of the usual hyperbolic tangent, we employ $g(z) = \text{erf}(z)$ as our sigmoid function (tanh and erf are quite similar functions). This simplifies the derivatives considerably and enables us to write

$$G(m, z_j) = \frac{4}{\pi} H_{m-1}^2(z_j)\exp[-2z_j^2], \tag{B:7}$$

where $H_n(z)$ are Hermite polynomials of order n. Plugging (B:7) into eq. (B:5), using the gaussian $p(\boldsymbol{x})$, and integrating out all dimensions that are orthogonal to $\boldsymbol{v}_j$ results in

$$\begin{aligned}\overline{G}(m,j) &= \frac{16}{\pi^2\sigma\|\boldsymbol{v}_j\|\sqrt{2\pi}}\exp\left[\frac{-v_{j0}^2}{2\sigma^2\|\boldsymbol{v}_j\|^2}\right]\times \\ &\int_{-\infty}^{\infty} dt H_{m-1}^2(t)\exp\left[-t^2\frac{(4\sigma^2\|\boldsymbol{v}_j\|^2+1)}{2\sigma^2\|\boldsymbol{v}_j\|^2}+t\frac{v_{j0}}{\sigma^2\|\boldsymbol{v}_j\|^2}\right]. \qquad \text{(B:8)}\end{aligned}$$

Evaluating the integral gives the result

$$\begin{aligned}\overline{G}(m,j) &= \frac{8\sqrt{4\sigma^2\|\boldsymbol{v}_j\|^2+1}}{\pi^2\sigma^2\|\boldsymbol{v}_j\|^2}\exp\left[\frac{-v_{j0}^2(4\sigma^4\|\boldsymbol{v}_j\|^4-4\sigma^2\|\boldsymbol{v}_j\|^2-1)}{8\sigma^6\|\boldsymbol{v}_j\|^6}\right]\times \\ &\sum_{k=0}^{m-1}2^k k!\binom{m-1}{k}^2\left(\frac{2\sigma^2\|\boldsymbol{v}_j\|^2+1}{4\sigma^2\|\boldsymbol{v}_j\|^2+1}\right)^{m-k-1}\times \\ &H_{2m-2k-2}\left[\frac{v_{j0}}{\sqrt{8\sigma^4\|\boldsymbol{v}_j\|^4+6\sigma^2\|\boldsymbol{v}_j\|^2+1}}\right]. \qquad \text{(B:9)}\end{aligned}$$

In this form, however, the expression gives little insight as to how $\overline{G}(m,j)$ scales with the weights. If we can assume that $\sigma^2\|\boldsymbol{v}_j\|^2 > 1$, expression (B:9) simplifies to

$$\overline{G}(m,j) = \frac{32}{\pi^2 2^m\sigma\|\boldsymbol{v}_j\|}\exp\left[\frac{-v_{j0}^2}{2\sigma^2\|\boldsymbol{v}_j\|^2}\right]\sum_{k=0}^{m-1}4^k k!\binom{m-1}{k}^2 H_{2m-2k-2}\left[\frac{v_{j0}}{\sigma^2\|\boldsymbol{v}_j\|^2\sqrt{8}}\right]. \qquad \text{(B:10)}$$

Furthermore, if $v_{j0} < \sigma^2\|\boldsymbol{v}_j\|^2$ then the weight independent term dominates the sum, and the scaling is (ignoring m)

$$\overline{G}(m,j) \sim \frac{1}{\sigma\|\boldsymbol{v}_j\|}\exp\left[\frac{-v_{j0}^2}{2\sigma^2\|\boldsymbol{v}_j\|^2}\right], \qquad \text{(B:11)}$$

which, as expected, resembles the scaling of the corresponding global case. Since (B:11) only depends on the magnitude $\|\boldsymbol{v}_j\|$, the expectation value of $\overline{G}(m,j)$ should vary very little with j. For instance, if the probability density of the v_{jk} weights is spherically symmetric (excluding the bias weights for the moment), we can do the transformation

$$\langle\overline{G}^n(m,j)\rangle = \int d^D v p(\|\boldsymbol{v}\|)\overline{G}^n(m,\|\boldsymbol{v}\|) = \frac{2\pi^{D/2}}{\Gamma(D/2)}\int dv v^{D-1}p(v)\overline{G}^n(m,v), \qquad \text{(B:12)}$$

where $v = \|\boldsymbol{v}\|$. The index j is dropped on $\boldsymbol{v}$ since we are here taking an expectation over the distribution of weights. If the weight distribution is gaussian and D is large (larger than 10), then the integrand will be sharply peaked so that $\langle\overline{G}^n(m,j)\rangle \approx \langle\overline{G}(m,j)\rangle^n$, resulting in a small variance w.r.t. the weights v_{jk}.

If the bias weights v_{j0} follow a gaussian distribution, with variance σ_0, then we have

$$\langle\overline{G}^n(m,j)\rangle = \frac{1}{\sigma^n\|\boldsymbol{v}\|^n\sigma_0\sqrt{2\pi}}\int dv_0\exp\left[\frac{-v_0^2}{2}\left(\frac{1}{\sigma_0^2}+\frac{n}{\sigma^2\|\boldsymbol{v}\|^2}\right)\right] = \frac{\sigma^{1-n}\|\boldsymbol{v}\|^{1-n}}{\sqrt{\sigma^2\|\boldsymbol{v}\|^2+n\sigma_0^2}}. \qquad \text{(B:13)}$$

Since $\|\boldsymbol{v}\|^2 \propto D$, it is likely that $\sigma_0 \ll \sigma\|\boldsymbol{v}\|$, why the variance due to bias weights will supposedly be very small as well. Furthermore, the variance will decrease as D^{-1}.

In summary, the local smoothing regularizer for networks with projective basis functions is an acceptable approximation. This is mainly due to the fact that $\|v_j\|$ has essentially the same value for all hidden units. The variation between units caused by different bias weight values will also, at least in reasonable cases, be small.

References

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

Bishop, C. (1991), 'Improving the generalization properties of radial basis function neural networks', *Neural Computation* **3**, 579–588.

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

Buntine, W. L. & Weigend, A. S. (1991), 'Bayesian back-propagation', *Complex Systems* **5**, 603–643.

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.

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

Friedman, J. H. & Stuetzle, W. (1981), 'Projection pursuit regression', *J. Amer. Stat. Assoc.* **76**(376), 817–823.

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.

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

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.

Ishikawa, M. (1996), 'Structural learning with forgetting', *Neural Networks* **9**(3), 509–521.

Kendall, M. G. & Stuart, A. (1972), *The Advanced Theory of Statistics*, third edn, Hafner Publishing Co., New York.

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

Lindley, D. V. & Smith, A. F. M. (1972), 'Bayes estimates for the linear model', *Journal of the Royal Statistical Society* **B 34**(302), 1–41.

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. (1996), Smoothing regularizers for radial basis function networks, Manuscript in preparation.

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

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.

Riedmiller, M. & Braun, H. (1993), A direct adaptive method for faster backpropagation learning: The RPROP algorithm, *in* H. Ruspini, ed., 'Proc. of the IEEE Intl. Conference on Neural Networks', San Fransisco, California, pp. 586–591.

Rumelhart, D. E. (1988), Learning and generalization, *in* 'Proc. of the IEEE Intl. Conference on Neural Networks', San Diego. (Plenary address).

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.

Smith, G. & Campbell, F. (1980), 'A critique of some ridge regression methods', *Journal of the American Statistical Association* **75**(369), 74–103.

van Vuuren, S. H. J. (1994), Neural network correlates with generalization, Master's thesis, University of Pretoria. http://www.cse.ogi.edu/∼sarelv.

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.

Williams, P. M. (1995), 'Bayesian regularization and pruning using a Laplace prior', *Neural Computation* **7**, 117–143.

Wu, L. & Moody, J. (1996), 'A smoothing regularizer for feedforward and recurrent networks', *Neural Computation* **8**(2).