Techreport NCRG/97/025. Under review for Neural Computation

Computation with infinite neural networks

Christopher K. I. Williams
Neural Computing Research Group
Department of Computer Science and Applied Mathematics
Aston University, Birmingham B4 7ET, UK
c.k.i.williams@aston.ac.uk

September 1997

Abstract

For neural networks with a wide class of weight priors, it can be shown that in the limit of an infinite number of hidden units the prior over functions tends to a Gaussian process. In this paper analytic forms are derived for the covariance function of the Gaussian processes corresponding to networks with sigmoidal and Gaussian hidden units. This allows predictions to be made efficiently using networks with an infinite number of hidden units, and shows that, somewhat paradoxically, it may be easier to carry out Bayesian prediction with infinite networks rather than finite ones.

1 Introduction

To someone training a neural network by maximizing the likelihood of a finite amount of data it makes no sense to use a network with an infinite number of hidden units; the network will "overfit" the data and so will be expected to generalize poorly. However, the idea of selecting the network size depending on the amount of training data makes little sense to a Bayesian; a model should be chosen that reflects the understanding of the problem, and then application of Bayes' theorem allows inference to be carried out (at least in theory) after the data is observed.

In the Bayesian treatment of neural networks, a question immediately arises as to how many hidden units are believed to be appropriate for a task. Neal (1996) has argued compellingly that for real-world problems, there is no reason to believe that neural network models should be limited to nets containing only a "small" number of hidden units. He has shown that it is sensible to consider a limit where the number of hidden units in a net tends to infinity, and that good predictions can be obtained from such models using the Bayesian machinery. He has also shown that for fixed hyperparameters, a large class of neural network models will converge to a Gaussian process prior over functions in the limit of an infinite number of hidden units.

Neal's argument is an existence proof—it states that an infinite neural net will converge to a Gaussian process, but does not give the covariance function needed to actually specify the particular Gaussian process. In this paper (which is a revised and expanded version of Williams (1997)), I show that for certain weight priors and transfer functions in the neural network model, the covariance function which describes the behaviour of the corresponding Gaussian process can be calculated analytically. This allows predictions to be made using neural networks with an infinite number of hidden units in time $O(n^3)$, where n is the number of training examples[1]. The only alternative currently available is to use Markov Chain Monte Carlo (MCMC) methods (e.g. Neal, 1996) for networks with a large (but finite) number of hidden units. However, this is likely to be computationally expensive, and there are possible concerns over the time needed for the Markov chain to reach equilibrium. The availability of an analytic form for the covariance function also facilitates the comparison of the properties of neural networks with an infinite number of hidden units as compared to other Gaussian process priors that may be considered.

[1] For large n, various approximations to the exact solution which avoid the inversion of an $n \times n$ matrix are available; see e.g. Gibbs and MacKay (1997) and discussion in section 4.

The Gaussian process analysis applies for fixed hyperparameters $\boldsymbol{\theta}$. If it were desired to make predictions based on a hyperprior $P(\boldsymbol{\theta})$ then the necessary $\boldsymbol{\theta}$-space integration could be achieved by MCMC methods. (This is described in more detail in section 3.4.) In comparison, for networks with a large but finite number of hidden units, MCMC methods (e.g. Neal, 1996) require us to carry out integration over both the weight and hyperparameter spaces. Thus the great advantage of integrating out the weights analytically for infinite networks is that it yields much lower-dimensional integrals to be computed by MCMC methods than with finite nets, and should thus improve the computational efficiency of these techniques.

1.1 From priors on weights to priors on functions

Bayesian neural networks are usually specified in a hierarchical manner, so that the weights $\boldsymbol{w}$ are regarded as being drawn from a distribution $P(\boldsymbol{w}|\boldsymbol{\theta})$. For example, the weights might be drawn from a zero-mean Gaussian distribution, where $\boldsymbol{\theta}$ specifies the variance of groups of weights. A full description of the prior is given by specifying $P(\boldsymbol{\theta})$ as well as $P(\boldsymbol{w}|\boldsymbol{\theta})$. The hyperprior can be integrated out to give $P(\boldsymbol{w}) = \int P(\boldsymbol{w}|\boldsymbol{\theta})P(\boldsymbol{\theta})\, d\boldsymbol{\theta}$, but in our case it will be advantageous not to do this as it introduces weight correlations which prevent convergence to a Gaussian process.

In the Bayesian view of neural networks, predictions for the output value y_* corresponding to a new input value $\boldsymbol{x}_*$ are made by integrating over the posterior in weight space. Let $\mathcal{D} = ((\boldsymbol{x}_1, t_1), (\boldsymbol{x}_2, t_2), \ldots, (\boldsymbol{x}_n, t_n))$ denote the n training data pairs where t_i denotes the target corresponding to input $\boldsymbol{x}_i$, $\boldsymbol{t} = (t_1, \ldots, t_n)^T$ and let $f_*(\boldsymbol{w})$ denote the value output by the network for input $\boldsymbol{x}_*$ given weights $\boldsymbol{w}$. Each t_i is assumed to be a noise-corrupted version of an underlying y_i. $P(\boldsymbol{w}|\boldsymbol{t}, \boldsymbol{\theta})$ is the weight posterior given the training data[2]. Then the predictive distribution for y_* given the training data and hyperparameters $\boldsymbol{\theta}$ is

$$P(y_*|\boldsymbol{t}, \boldsymbol{\theta}) = \int \delta(y_* - f_*(\boldsymbol{w}))P(\boldsymbol{w}|\boldsymbol{t}, \boldsymbol{\theta})\, d\boldsymbol{w} \tag{1}$$

We will now show how this can also be viewed as making the prediction using priors over functions rather than weights. Let $\boldsymbol{f}(\boldsymbol{w})$ denote the vector of outputs corresponding to inputs $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ given weights $\boldsymbol{w}$. Then, using Bayes' theorem we have $P(\boldsymbol{w}|\boldsymbol{t}, \boldsymbol{\theta}) = P(\boldsymbol{t}|\boldsymbol{w})P(\boldsymbol{w}|\boldsymbol{\theta})/P(\boldsymbol{t}|\boldsymbol{\theta})$, and $P(\boldsymbol{t}|\boldsymbol{w}) = \int P(\boldsymbol{t}|\boldsymbol{y})\, \delta(\boldsymbol{y} - \boldsymbol{f}(\boldsymbol{w}))\, d\boldsymbol{y}$. Hence equation 1 can be rewritten as

$$P(y_*|\boldsymbol{t}, \boldsymbol{\theta}) = \frac{1}{P(\boldsymbol{t}|\boldsymbol{\theta})} \int\!\!\int P(\boldsymbol{t}|\boldsymbol{y})\, \delta(y_* - f_*(\boldsymbol{w}))\delta(\boldsymbol{y} - \boldsymbol{f}(\boldsymbol{w}))\, P(\boldsymbol{w}|\boldsymbol{\theta})\, d\boldsymbol{w}\, d\boldsymbol{y} \tag{2}$$

However, the prior over $(y_*, y_1, \ldots, y_n)$ is given by $P(y_*, \boldsymbol{y}|\boldsymbol{\theta}) = P(y_*|\boldsymbol{y}, \boldsymbol{\theta})P(\boldsymbol{y}|\boldsymbol{\theta}) = \int \delta(y_* - f_*(\boldsymbol{w}))\, \delta(\boldsymbol{y} - \boldsymbol{f}(\boldsymbol{w}))P(\boldsymbol{w}|\boldsymbol{\theta})\, d\boldsymbol{w}$ and thus the predictive distribution can be written as

$$P(y_*|\boldsymbol{t}, \boldsymbol{\theta}) = \frac{1}{P(\boldsymbol{t}|\boldsymbol{\theta})} \int P(\boldsymbol{t}|\boldsymbol{y})P(y_*|\boldsymbol{y}, \boldsymbol{\theta})P(\boldsymbol{y}|\boldsymbol{\theta})\, d\boldsymbol{y} = \int P(y_*|\boldsymbol{y}, \boldsymbol{\theta})P(\boldsymbol{y}|\boldsymbol{t}, \boldsymbol{\theta})\, d\boldsymbol{y} \tag{3}$$

Hence in a Bayesian view it is the prior over function values $P(y_*, \boldsymbol{y}|\boldsymbol{\theta})$ which is important; specifying this prior by using weight distributions is one valid way to achieve this goal. In general we can use the weight space or function space view, which ever is more convenient, and for infinite neural networks the function space view is more useful.

2 Gaussian processes

A stochastic process is a collection of random variables $\{Y(\boldsymbol{x})|\boldsymbol{x} \in X\}$ indexed by a set X. In our case X will be $\mathbb{R}^d$, where d is the number of inputs. The stochastic process is specified by giving the probability distribution for every finite subset of variables $Y(\boldsymbol{x}_1), \ldots, Y(\boldsymbol{x}_k)$ in a consistent manner. A Gaussian process (GP) is a stochastic process which can be fully specified by its mean function $\mu(\boldsymbol{x}) = E[Y(\boldsymbol{x})]$ and its covariance function $C(\boldsymbol{x}, \boldsymbol{x}') = E[(Y(\boldsymbol{x}) - \mu(\boldsymbol{x}))(Y(\boldsymbol{x}') - \mu(\boldsymbol{x}'))]$; any finite set of Y-variables will have a joint multivariate Gaussian distribution. For a multidimensional input space a Gaussian process may also be called a Gaussian random field.

[2]For notational convenience we suppress the $\boldsymbol{x}$-dependence of the posterior.

Below we consider Gaussian processes which have $\mu(\boldsymbol{x}) \equiv 0$, as is the case for the neural network priors discussed in section 3. A non-zero $\mu(\boldsymbol{x})$ can be incorporated into the framework at the expense of a little extra complexity.

A widely used class of covariance functions is the stationary covariance functions, whereby $C(\boldsymbol{x}, \boldsymbol{x}') = C(\boldsymbol{x} - \boldsymbol{x}')$. These are related to the spectral density (or power spectrum) of the process by the Wiener-Khinchine theorem, and are particularly amenable to Fourier analysis as the eigenfunctions of a stationary covariance kernel are $\exp i\boldsymbol{k}.\boldsymbol{x}$. Many commonly used covariance functions are also isotropic, so that $C(\boldsymbol{h}) = C(h)$ where $\boldsymbol{h} = \boldsymbol{x} - \boldsymbol{x}'$ and $h = |\boldsymbol{h}|$. For example $C(h) = \exp(-(h/\sigma)^\nu)$ is a valid covariance function for all input dimensionalities $d \geq 1$ and for $0 < \nu \leq 2$ (Yaglom, 1987, pg. 137). Note that in this case σ sets the correlation length-scale of the random field, although other covariance functions (e.g. those corresponding to power-law spectral densities) may have no preferred length scale.

A simple example of a non-stationary covariance function is obtained by letting $Y(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x})$ for some set of basis functions $\{\phi_i(\boldsymbol{x})\}$. Letting $\boldsymbol{w} \sim N(0, \Sigma)$ we obtain $C_\phi(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{\phi}^T(\boldsymbol{x}) \Sigma \boldsymbol{\phi}(\boldsymbol{x}')$.

Gaussian random fields can have very different properties depending on the choice of covariance function. For example, in 1-d, the Ornstein-Uhlenbeck process (with covariance function e^{-h}) has sample paths which are not mean-square differentiable, implying that the sample paths are very "rough". By contrast, choosing $y = \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x})$ with $\boldsymbol{\phi}(x) = (1, x)^T$ leads to straight-line sample paths of the form $y = w_0 + w_1 x$.

2.1 Prediction with Gaussian processes

The model for the observed data is that it was generated from the *prior* stochastic process, and that independent Gaussian noise (of variance σ_ν^2) was then added. Given a prior covariance function $C_P(\boldsymbol{x}_i, \boldsymbol{x}_j)$, a noise process $C_N(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sigma_\nu^2 \delta_{ij}$ (i.e. independent noise of variance σ_ν^2 at each data point) and the training data, the prediction for the distribution of y_* corresponding to a test point $\boldsymbol{x}_*$ is obtained simply by applying equation 3. As the prior and noise model are both Gaussian the integral can be calculated analytically and $P(y_*|\boldsymbol{t}, \boldsymbol{\theta})$ is Gaussian with mean and variance

$$\hat{y}(\boldsymbol{x}_*) = \boldsymbol{k}_P^T(\boldsymbol{x}_*)(K_P + K_N)^{-1}\boldsymbol{t} \tag{4}$$

$$\sigma_{\hat{y}}^2(\boldsymbol{x}_*) = C_P(\boldsymbol{x}_*, \boldsymbol{x}_*) - \boldsymbol{k}_P^T(\boldsymbol{x}_*)(K_P + K_N)^{-1}\boldsymbol{k}_P(\boldsymbol{x}_*) \tag{5}$$

where $[K_P]_{ij} = C_P(\boldsymbol{x}_i, \boldsymbol{x}_j)$, $[K_N]_{ij} = C_N(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and $\boldsymbol{k}_P(\boldsymbol{x}_*) = (C_P(\boldsymbol{x}_*, \boldsymbol{x}_1), \ldots, C_P(\boldsymbol{x}_*, \boldsymbol{x}_n))^T$.

It is interesting to understand the form of the predictor $\hat{y}(\boldsymbol{x}_*)$ (as given by equation 4). For any given dataset $(K_P + K_N)^{-1}\boldsymbol{t}$ is a fixed vector. If the covariance function is isotropic, $\hat{y}(\boldsymbol{x}_*)$ is a linear combination of *radial basis functions*, as observed by Poggio and Girosi (1990), as each element in $\boldsymbol{k}_P(\boldsymbol{x}_*)$ is a radial function. (Poggio and Girosi derived radial basis functions via regularization theory, which is equivalent to the Gaussian process view; see e.g. Girosi *et* al (1995) and Wahba (1990) for more details.)

Equations 4 and 5 are the analogue for spatial processes of Wiener-Kolmogorov prediction theory. They have appeared in a wide variety of contexts including geostatistics where the method is known as "kriging" (Journel and Huijbregts, 1978; Cressie 1993), multidimensional spline smoothing (Wahba, 1990), and in the work of Whittle (1963).

It is also possible to view regression with support vector machines (Vapnik, 1995) in terms of Gaussian processes. The prior over functions is taken to be a Gaussian process, but instead of using a squared error loss function (which corresponds to Gaussian noise), a modified version of the l_1 error metric $|t_i - y_i|$ is used, called the ϵ-insensitive loss function. Finding the *maximum a posteriori* y-values for the training and test points can now be achieved using quadratic programming (see Vapnik, 1995 for details).

3 Covariance functions for Neural Networks

Consider a network which takes an input $\boldsymbol{x}$, has one hidden layer with H units and then linearly combines the outputs of the hidden units with a bias b to obtain $f(\boldsymbol{x})$. The mapping can be written

$$f(\boldsymbol{x}) = b + \sum_{j=1}^{H} v_j h(\boldsymbol{x}; \boldsymbol{u}_j) \tag{6}$$

where $h(\boldsymbol{x};\boldsymbol{u})$ is the hidden unit transfer function (which we shall assume is bounded) which depends on the input-to-hidden weights $\boldsymbol{u}$. This architecture is important because it has been shown by Hornik (1993) that networks with one hidden layer are universal approximators as the number of hidden units tends to infinity, for a wide class of transfer functions (but excluding polynomials). Let b and the v's have independent zero-mean distributions of variance σ_b^2 and σ_v^2 respectively, and let the weights $\boldsymbol{u}_j$ for each hidden unit be independently and identically distributed. Denoting all weights by $\boldsymbol{w}$, we obtain (following Neal, 1996)

$$
\begin{aligned}
E_{\boldsymbol{w}}[f(\boldsymbol{x})] &= 0 && (7)\\
E_{\boldsymbol{w}}[f(\boldsymbol{x})f(\boldsymbol{x}')] &= \sigma_b^2 + \sum_j \sigma_v^2 E_{\boldsymbol{u}}[h_j(\boldsymbol{x};\boldsymbol{u})h_j(\boldsymbol{x}';\boldsymbol{u})] && (8)\\
&= \sigma_b^2 + H\sigma_v^2 E_{\boldsymbol{u}}[h(\boldsymbol{x};\boldsymbol{u})h(\boldsymbol{x}';\boldsymbol{u})] && (9)
\end{aligned}
$$

where equation 9 follows because all of the hidden units are identically distributed. The final term in equation 9 becomes $\omega^2 E_{\boldsymbol{u}}[h(\boldsymbol{x};\boldsymbol{u})h(\boldsymbol{x}';\boldsymbol{u})]$ by letting σ_v^2 scale as ω^2/H.

The sum in equation 8 is over H identically and independently distributed random variables. As the transfer function is bounded, all moments of the distribution will be bounded and hence the Central Limit Theorem can be applied. Hence if we assume that b has a Gaussian distribution, the stochastic process will converge to a Gaussian process in the limit as $H \to \infty$. From general Central Limit Theorem results one would expect a rate of convergence of $H^{-1/2}$ towards a Gaussian process prior.

By evaluating $E_{\boldsymbol{u}}[h(\boldsymbol{x})h(\boldsymbol{x}')]$ for all $\boldsymbol{x}$ and $\boldsymbol{x}'$ in the training and testing sets we can obtain the covariance function needed to describe the neural network as a Gaussian process. These expectations are, of course, integrals over the relevant probability distributions of the biases and input weights. In the following sections two specific choices for the transfer functions are considered, (1) a sigmoidal function and (2) a Gaussian. Gaussian weight priors are used in both cases.

It is interesting to note why this analysis cannot be taken a stage further to integrate out any hyperparameters as well. For example, the variance σ_v^2 of the v weights might be drawn from an inverse Gamma distribution. In this case the distribution $P(\boldsymbol{v}) = \int P(\boldsymbol{v}|\sigma_v^2)P(\sigma_v^2)\, d\sigma_v^2$ is no longer the product of the marginal distributions for each v weight (in fact it will be a multivariate t-distribution). A similar analysis can be applied to the $\boldsymbol{u}$ weights with a hyperprior. The effect is to make the hidden units non-independent, so that the Central Limit Theorem can no longer be applied.

3.1 Sigmoidal transfer function

A sigmoidal transfer function is a very common choice in neural networks research; nets with this architecture are usually called multi-layer perceptrons.

Below we consider the transfer function $h(\boldsymbol{x};\boldsymbol{u}) = \mathrm{erf}(u_0 + \sum_{i=1}^d u_j x_i)$, where $\mathrm{erf}(z) = 2/\sqrt{\pi}\int_0^z e^{-t^2} dt$ is the error function, closely related to the cumulative distribution function for the Gaussian distribution. Appropriately scaled, the graph of this function is very similar to the tanh function which is more commonly used in the neural networks literature.

In calculating $V(\boldsymbol{x},\boldsymbol{x}') \overset{def}{=} E_{\boldsymbol{u}}[h(\boldsymbol{x};\boldsymbol{u})h(\boldsymbol{x}';\boldsymbol{u})]$ we make the usual assumptions (e.g. MacKay, 1992) that $\boldsymbol{u}$ is drawn from a zero-mean Gaussian distribution with covariance matrix Σ, i.e. $\boldsymbol{u} \sim N(0,\Sigma)$. Let $\tilde{\boldsymbol{x}} = (1, x_1, \ldots, x_d)^T$ be an augmented input vector whose first entry corresponds to the bias. Then $V_{\mathrm{erf}}(\boldsymbol{x},\boldsymbol{x}')$ can be written as

$$V_{\mathrm{erf}}(\boldsymbol{x},\boldsymbol{x}') = \frac{1}{(2\pi)^{\frac{d+1}{2}}|\Sigma|^{1/2}} \int \mathrm{erf}(\boldsymbol{u}^T\tilde{\boldsymbol{x}})\mathrm{erf}(\boldsymbol{u}^T\tilde{\boldsymbol{x}}')\exp(-\frac{1}{2}\boldsymbol{u}^T\Sigma^{-1}\boldsymbol{u})\, d\boldsymbol{u}. \quad (10)$$

Let C be the square root of Σ^{-1}, so that $C^T = C$ and $\Sigma^{-1} = C^2$. Let $\boldsymbol{u}' = C\boldsymbol{u}$. Then equation 10 becomes

$$V_{\mathrm{erf}}(\boldsymbol{x},\boldsymbol{x}') = \frac{1}{(2\pi)^{\frac{d+1}{2}}} \int \mathrm{erf}(\boldsymbol{u}'^T C^{-1}\tilde{\boldsymbol{x}})\mathrm{erf}(\boldsymbol{u}'^T C^{-1}\tilde{\boldsymbol{x}}')\exp(-\frac{1}{2}\boldsymbol{u}'^T\boldsymbol{u}')\, d\boldsymbol{u}' \quad (11)$$

which can be evaluated analytically (see Appendix A) to give

$$V_{\mathrm{erf}}(\boldsymbol{x},\boldsymbol{x}') = \frac{2}{\pi}\sin^{-1}\frac{2\tilde{\boldsymbol{x}}^T\Sigma\tilde{\boldsymbol{x}}'}{\sqrt{(1+2\tilde{\boldsymbol{x}}^T\Sigma\tilde{\boldsymbol{x}})(1+2\tilde{\boldsymbol{x}}'^T\Sigma\tilde{\boldsymbol{x}}')}}. \quad (12)$$

We observe that this covariance function is not stationary, which makes sense as the distributions for the weights are centered about zero, and hence translational symmetry is not present. Note that the range of $V_{\mathrm{erf}}(\boldsymbol{x},\boldsymbol{x}')$ is $[-1,1]$, and thus negative values of the covariance function are allowed[3].

To examine this covariance function in a little more detail we now consider a diagonal weight prior $\Sigma = \mathrm{diag}(\sigma_0^2, \sigma_I^2, \ldots, \sigma_I^2)$, so that the inputs $i = 1, \ldots, d$ have a different weight variance to the bias σ_0^2. Then for $|\boldsymbol{x}|^2$ and $|\boldsymbol{x}'|^2 \gg (1+2\sigma_0^2)/2\sigma_I^2$, we find that $V_{\mathrm{erf}}(\boldsymbol{x},\boldsymbol{x}') \simeq 1 - 2\theta/\pi$, where θ is the angle between $\boldsymbol{x}$ and $\boldsymbol{x}'$. Again this makes sense intuitively; if the model is made up of a large number of sigmoidal functions in random directions (in $\boldsymbol{x}$ space), then we would expect points that lie diametrically opposite (i.e. at $\boldsymbol{x}$ and $-\boldsymbol{x}$) to be anti-correlated, because they will lie in the $+1$ and -1 regions of the sigmoid function for most directions. In the other extreme, $\sigma_0^2 \gg \max(\sigma_I^2|\boldsymbol{x}|^2,\ \sigma_I^2|\boldsymbol{x}'|^2,\ 1)$, the zero-crossings of most of the sigmoids will be further away from the origin than $\boldsymbol{x}$ and $\boldsymbol{x}'$, thus $\boldsymbol{x}$ and $\boldsymbol{x}'$ will lie in the same tail of each sigmoid and hence $V_{\mathrm{erf}}(\boldsymbol{x},\boldsymbol{x}') \simeq 1$, meaning that any prediction in this region is just the average of all of the t's.

It is also worth noting that for "projective" basis functions (i.e. those for which $h(\boldsymbol{x};\boldsymbol{u}) = \psi(\boldsymbol{u}^T\tilde{\boldsymbol{x}})$ for some function $\psi(\cdot)$), the covariance structure obtained by passing the Gaussian process $Y(\boldsymbol{x}) = \boldsymbol{u}^T\tilde{\boldsymbol{x}}$ with $\boldsymbol{u} \sim N(\boldsymbol{0},\Sigma)$ through the function ψ is the same as the covariance term $E_{\boldsymbol{u}}[h(\boldsymbol{x};\boldsymbol{u})h(\boldsymbol{x}';\boldsymbol{u})]$ as found in equation 9.

In this section we have derived the covariance function for an erf network with an infinite number of hidden units. In the reverse direction, Girosi *et* al (1995) and Wahba (1996) have discussed how the prior covariance function may be chosen so as to obtain a predictor $\hat{y}(\boldsymbol{x})$ which is a linear combination of sigmoidal basis functions.

3.2 Gaussian transfer function

One other very common transfer function used in neural networks research is the Gaussian, so that $h(\boldsymbol{x};\boldsymbol{u}) = \exp[-(\boldsymbol{x}-\boldsymbol{u})^T(\boldsymbol{x}-\boldsymbol{u})/2\sigma_g^2]$, where σ_g^2 is the width parameter of the Gaussian. Gaussian basis functions are often used in Radial Basis Function (RBF) networks (e.g. Poggio and Girosi, 1990; Broomhead and Lowe, 1988).

For a Gaussian prior over the distribution of $\boldsymbol{u}$ so that $\boldsymbol{u} \sim N(0, \sigma_u^2 I)$,

$$V_G(\boldsymbol{x},\boldsymbol{x}') = \frac{1}{(2\pi\sigma_u^2)^{d/2}} \int \exp -\frac{(\boldsymbol{x}-\boldsymbol{u})^T(\boldsymbol{x}-\boldsymbol{u})}{2\sigma_g^2} \exp -\frac{(\boldsymbol{x}'-\boldsymbol{u})^T(\boldsymbol{x}'-\boldsymbol{u})}{2\sigma_g^2} \exp -\frac{\boldsymbol{u}^T\boldsymbol{u}}{2\sigma_u^2} d\boldsymbol{u} \tag{13}$$

By completing the square and integrating out $\boldsymbol{u}$ we obtain

$$V_G(\boldsymbol{x},\boldsymbol{x}') = \left(\frac{\sigma_e}{\sigma_u}\right)^d \exp\{-\frac{\boldsymbol{x}^T\boldsymbol{x}}{2\sigma_m^2}\} \exp\{-\frac{(\boldsymbol{x}-\boldsymbol{x}')^T(\boldsymbol{x}-\boldsymbol{x}')}{2\sigma_s^2}\} \exp\{-\frac{\boldsymbol{x}'^T\boldsymbol{x}'}{2\sigma_m^2}\} \tag{14}$$

where $1/\sigma_e^2 = 2/\sigma_g^2 + 1/\sigma_u^2$, $\sigma_s^2 = 2\sigma_g^2 + \sigma_g^4/\sigma_u^2$ and $\sigma_m^2 = 2\sigma_u^2 + \sigma_g^2$. This formula can be generalized by allowing covariance matrices Σ_b and Σ_u in place of $\sigma_g^2 I$ and $\sigma_u^2 I$; rescaling each input variable x_i independently is a simple example.

Again this is a non-stationary covariance function, although it is interesting to note that if $\sigma_u^2 \to \infty$ (while scaling ω^2 appropriately) we find that $V_G(\boldsymbol{x},\boldsymbol{x}') \propto \exp\{-(\boldsymbol{x}-\boldsymbol{x}')^T(\boldsymbol{x}-\boldsymbol{x}')/4\sigma_g^2\}$. For a finite value of σ_u^2, $V_G(\boldsymbol{x},\boldsymbol{x}')$ is a stationary covariance function "modulated" by the Gaussian decay function $\exp(-\boldsymbol{x}^T\boldsymbol{x}/2\sigma_m^2)\exp(-\boldsymbol{x}'^T\boldsymbol{x}'/2\sigma_m^2)$. Clearly if σ_m^2 is much larger than the largest distance in $\boldsymbol{x}$-space then the predictions made with V_G and a Gaussian process with only the stationary part of V_G will be very similar.

It is also possible to view a network with Gaussian transfer functions as an example of a shot-noise process based on an inhomogeneous Poisson process. A realization of a shot-noise process is obtained by placing a kernel (or shape-function) at each location defined by a Poisson process in $\mathbb{R}^d$. It is also possible to allow the kernel to have some random component such as a magnitude. In this case we can write

$$Y(\boldsymbol{x}) = \sum_n V_n h(\boldsymbol{x},\boldsymbol{x}_n) \tag{15}$$

[3]Although many commonly-used covariance functions are non-negative, there are examples such as the exponentially damped cosine $C(x, x+\tau) = \exp(-\alpha\tau)\cos\omega\tau$ which obtain negative values; see Yaglom (1987), pp. 366-367.

where the $\boldsymbol{x}_n$'s are the Poisson distributed, $h(\cdot,\cdot)$ is the kernel (which will often be a function of $\boldsymbol{x}-\boldsymbol{x}_i$) and the V_n's are iid random variables. It can be shown that the mean and covariance functions of this shot-noise process are given by (see Parzen (1962) §4.5 for details)

$$
\begin{aligned}
E[Y(\boldsymbol{x})] &= E[V]\int h(\boldsymbol{x},\boldsymbol{\xi})\nu(\boldsymbol{\xi})\,d\boldsymbol{\xi} && (16)\\
\operatorname{cov}[Y(\boldsymbol{x})Y(\boldsymbol{x}')] &= E[V^2]\int h(\boldsymbol{x},\boldsymbol{\xi})h(\boldsymbol{x}',\boldsymbol{\xi})\nu(\boldsymbol{\xi})\,d\boldsymbol{\xi} && (17)
\end{aligned}
$$

where $\nu(\boldsymbol{\xi})$ is the rate of the underlying Poisson process. These results are known as Campbell's theorem. By choosing the kernel function to be a Gaussian and the rate function of the form $\nu(\boldsymbol{x}) = \nu_0 \exp(-\boldsymbol{x}^T\boldsymbol{x}/2\sigma_u^2)$, we obtain a Gaussian process in the limit of $\nu_0 \to \infty$ with $E[V^2]$ scaled such that $\nu_0 E[V^2]$ remains finite. This gives a geometrical interpretation of a realization of the network as being comprised of kernels centered at random points $\boldsymbol{u}_j$ in $\boldsymbol{x}$-space and with each kernel scaled by a random variable corresponding to the hidden-to-output weight v_j.

For the case that the Poisson process has constant rate, Matérn (1980, Table 1, p. 30) gives some examples of other kernel functions and their corresponding covariance functions.

3.3 Comparing covariance functions

The priors over functions specified by sigmoidal and Gaussian neural networks differ from the stationary covariance functions that are usually employed when using Gaussian processes for prediction. The complex exponential $\exp i\boldsymbol{k}.\boldsymbol{x}$ is an eigenfunction of a stationary and isotropic covariance function, and hence the spectral density (or power spectrum) $S(k)$ ($k = |\boldsymbol{k}|$) nicely characterizes the corresponding stochastic process. Roughly speaking the spectral density describes the "power" at a given spatial frequency k; for example, splines have $S(k) \propto k^{-\beta}$. Non-stationary processes cannot be analyzed in exactly this fashion because the complex exponentials are not (in general) eigenfunctions of a non-stationary kernel. Instead, we must consider the eigenfunctions defined by $\int C(\boldsymbol{x},\boldsymbol{x}')\phi(\boldsymbol{x}')d\boldsymbol{x}' = \lambda\phi(\boldsymbol{x})$. This difference in eigenfunctions makes direct comparison of stationary and non-stationary covariance functions difficult.

The distinction between stationary and non-stationary covariance functions can be reduced to some extent if we consider Gaussian-process prediction in the presence of a non-negative weight function $w(\boldsymbol{x})$. In this case it turns out that the covariance function that we need to consider is $K(\boldsymbol{x},\boldsymbol{x}') = w^{1/2}(\boldsymbol{x})C(\boldsymbol{x},\boldsymbol{x}')w^{1/2}(\boldsymbol{x}')$ (see, e.g. Zhu *et al* (1997), Castro *et al*, (1986)), where $C(\boldsymbol{x},\boldsymbol{x}')$ is the original covariance function which applies under uniform weighting. For example, there is a close connection between the eigenfunctions of the covariance function $V_G(\boldsymbol{x},\boldsymbol{x}')$ obtained in the Gaussian transfer function case and $K(\boldsymbol{x},\boldsymbol{x}')$ derived from a squared exponential covariance function $C(\boldsymbol{x},\boldsymbol{x}') \propto \exp -(\boldsymbol{x}-\boldsymbol{x}')^T(\boldsymbol{x}-\boldsymbol{x}')/2\sigma_g^2$ under a Gaussian weight function. The eigenvalues and functions for this $K(\boldsymbol{x},\boldsymbol{x}')$ are given in Zhu *et al* (1997)[4].

3.4 Integration over the hyperparameters

In the derivations above the hyperparameters $\boldsymbol{\theta}$ were fixed. However, in a real data analysis problem it would be unlikely that appropriate values of these parameters would be known. Given a prior distribution $P(\boldsymbol{\theta})$ predictions should be made by integrating over the posterior distribution $P(\boldsymbol{\theta}|\boldsymbol{t}) \propto P(\boldsymbol{\theta})P(\boldsymbol{t}|\boldsymbol{\theta})$, where $P(\boldsymbol{t}|\boldsymbol{\theta})$ is the likelihood of the training data $\boldsymbol{t}$ under the model; $P(\boldsymbol{t}|\boldsymbol{\theta})$ is easily computed for a Gaussian process. The prediction $\overline{y}(\boldsymbol{x})$ for test input $\boldsymbol{x}$ is then given by

$$
\overline{y}(\boldsymbol{x}) = \int \hat{y}_{\boldsymbol{\theta}}(\boldsymbol{x})P(\boldsymbol{\theta}|D)d\boldsymbol{\theta} \qquad (18)
$$

where $\hat{y}_{\boldsymbol{\theta}}(\boldsymbol{x})$ is the predicted mean (as given by equation 4) for a particular value of $\boldsymbol{\theta}$. This integration is not tractable analytically but Markov Chain Monte Carlo methods such as Hybrid Monte Carlo can be used to approximate it. This strategy was used in Williams and Rasmussen (1996) and Rasmussen (1996), but for stationary covariance functions, not ones derived from neural networks.

[4]For the one-dimensional problem the eigenfunctions are an appropriately scaled Gaussian multiplied by the Hermite polynomials. These are also the eigenfunctions of Schrodinger's equation for the simple harmonic oscillator. For the d-dimensional case the eigenfunctions are just the tensor product of the one-dimensional eigenfunctions.

4 Discussion

The work described above shows how to calculate the covariance function for sigmoidal and Gaussian basis functions networks. It is probable similar techniques will allow covariance functions to be derived analytically for networks with other kinds of basis functions as well; these may turn out to be similar in form to covariance functions already used in the Gaussian process literature.

This paper extends the work of Neal (1996) in examining what kinds of priors over functions one obtains by specifying particular weight priors. This is important as it is necessary to understand our prior assumptions when carrying out Bayesian modelling. The existence of analytic expressions for the covariance functions allows predictions to be made efficiently from neural networks that have an infinite number of hidden units.

It is also interesting to consider the differences between finite and infinite neural network priors. One difference is that in functions generated from finite neural networks, the effects of individual basis functions can be seen. For example, with sigmoidal units a steep step may be observed where one basis function (with large weights) comes into play. A judgement about whether this type of behaviour is appropriate or not should depend on prior beliefs about the problem at hand. Of course it is also be possible to compare finite neural networks and Gaussian process predictions empirically. This has been done by Rasmussen (1996), where GP predictions (using MCMC for the hyperparameters) were compared to those from Neal's MCMC Bayesian neural networks. Rasmussen's results show that for a number of problems, the predictive performance of GPs and Bayesian neural networks is similar. This similarity may be due to the fact that Rasmussen used networks with relatively large numbers of hidden units, so that results similar to a Gaussian process predictor are not unexpected.

Finite and infinite neural networks can also be compared on the basis of the computational requirements for prediction. With small data sets, as we have argued above, it should be more effective to use GPs as this dramatically reduces the dimensionality of the space over which MCMC integrations are done. However, for large problems the matrix inversion required in equation 4 can be expensive. Computational approximations can be used e.g. Gibbs and MacKay (1997) but it may also be interesting to investigate the use of finite neural networks as approximations to infinite ones. This would need to be contrasted with standard eigenfunction decompositions of the covariance, as described for example in Zhu *et al*, 1997.

In this paper we have studied the covariance function derived from a given weight prior. It is also interesting to ask how one might try to specify a weight prior to approximate a given covariance function. There is some recent work by Moody and Rögnvaldsson (1997) in this direction, where correlations between v_j and $\boldsymbol{u}_j$ in the prior are discussed.

Appendix A

We wish to evaluate

$$V_{\mathrm{erf}}(\boldsymbol{z},\boldsymbol{z}') = \frac{1}{(2\pi)^{\frac{d+1}{2}}} \int \mathrm{erf}(\boldsymbol{u}^T\tilde{\boldsymbol{z}})\mathrm{erf}(\boldsymbol{u}^T\tilde{\boldsymbol{z}}')\exp(-\frac{1}{2}\boldsymbol{u}^T\boldsymbol{u})d\boldsymbol{u}, \tag{19}$$

where $\tilde{\boldsymbol{z}}$ is the concatenation of 1 and $\boldsymbol{z}$.

The first step is to realize that the $d+1$ dimensional integral in equation 19 can be reduced to a two-dimensional integral by rotating the basis so that two basis vectors ($\boldsymbol{e}_1$ and $\boldsymbol{e}_2$) lie in the plane defined by $\tilde{\boldsymbol{z}}$ and $\tilde{\boldsymbol{z}}'$ and the rest are orthogonal to this plane, to give $\tilde{\boldsymbol{z}} = a_{11}\boldsymbol{e}_1 + a_{12}\boldsymbol{e}_2$ and $\tilde{\boldsymbol{z}}' = a_{21}\boldsymbol{e}_1 + a_{22}\boldsymbol{e}_2$. Hence $\boldsymbol{u}^T\tilde{\boldsymbol{z}} = a_{11}u_1 + a_{12}u_2$, and similarly for $\boldsymbol{u}^T\tilde{\boldsymbol{z}}'$. Thus equation 19 reduces to

$$V_{\mathrm{erf}}(\boldsymbol{z},\boldsymbol{z}') = \frac{1}{2\pi}\int\int \mathrm{erf}(\boldsymbol{a}_1^T\boldsymbol{u})\mathrm{erf}(\boldsymbol{a}_2^T\boldsymbol{u})\exp(-\frac{u_1^2+u_2^2}{2})du_1du_2 \tag{20}$$

where $\boldsymbol{a}_1 = (a_{11}, a_{12})^T$ and $\boldsymbol{a}_2 = (a_{21}, a_{22})^T$. Consider

$$I(\lambda) = \frac{1}{2\pi}\int \mathrm{erf}(\lambda\boldsymbol{a}_1^T\boldsymbol{u})\mathrm{erf}(\boldsymbol{a}_2^T\boldsymbol{u})e^{-u^2/2}d\boldsymbol{u} \tag{21}$$

where u^2 denotes $\boldsymbol{u}^T\boldsymbol{u}$. Clearly $I(1) = V_{\text{erf}}(\boldsymbol{z}, \boldsymbol{z}')$. Differentiating with respect to λ

$$I'(\lambda) = \frac{dI(\lambda)}{d\lambda} = \frac{1}{2\pi}\frac{2}{\sqrt{\pi}}\int \boldsymbol{a}_1^T\boldsymbol{u}\,\text{erf}(\boldsymbol{a}_2^T\boldsymbol{u})\exp -\frac{1}{2}\boldsymbol{u}^T(I + 2\lambda^2\boldsymbol{a}_1\boldsymbol{a}_1^T)\boldsymbol{u}\,d\boldsymbol{u}. \tag{22}$$

Putting $B = I + 2\lambda^2\boldsymbol{a}_1\boldsymbol{a}_1^T$, $\tilde{\boldsymbol{u}} = B^{1/2}\boldsymbol{u}$, $\boldsymbol{c}_1 = B^{-1/2}\boldsymbol{a}_1$ and $\boldsymbol{c}_2 = B^{-1/2}\boldsymbol{a}_2$ we obtain

$$I'(\lambda) = \frac{1}{\pi^{3/2}|B|^{1/2}}\int \boldsymbol{c}_1^T\tilde{\boldsymbol{u}}\,\text{erf}(\boldsymbol{c}_2^T\tilde{\boldsymbol{u}})e^{-\tilde{u}^2/2}\,d\tilde{\boldsymbol{u}}. \tag{23}$$

Writing $\boldsymbol{c}_1^T\tilde{\boldsymbol{u}} = c_{11}\tilde{u}_1 + c_{12}\tilde{u}_2$ and integrating each of the two terms by parts yields

$$I'(\lambda) = \frac{2\boldsymbol{c}_1^T\boldsymbol{c}_2}{\pi^2|B|^{1/2}}\int \exp -\frac{1}{2}\tilde{\boldsymbol{u}}^T(I + 2\boldsymbol{c}_2\boldsymbol{c}_2^T)\tilde{\boldsymbol{u}}\,d\tilde{\boldsymbol{u}} \tag{24}$$

$$= \frac{4\boldsymbol{c}_1^T\boldsymbol{c}_2}{\pi}\frac{1}{|B|^{1/2}|I + 2\boldsymbol{c}_2\boldsymbol{c}_2^T|^{1/2}} \tag{25}$$

$$= \frac{4}{\pi}\frac{\boldsymbol{a}_1^T B^{-1}\boldsymbol{a}_2}{|I + 2\lambda^2\boldsymbol{a}_1\boldsymbol{a}_1^T + 2\boldsymbol{a}_2\boldsymbol{a}_2^T|^{1/2}}. \tag{26}$$

Using $(I + \boldsymbol{g}\boldsymbol{g}^T)^{-1} = I - \boldsymbol{g}\boldsymbol{g}^T/(1 + g^2)$ we find that

$$\boldsymbol{a}_1^T B^{-1}\boldsymbol{a}_2 = \frac{\boldsymbol{a}_1^T\boldsymbol{a}_2}{1 + 2\lambda^2 a_1^2} \tag{27}$$

By explicit computation of the 2×2 determinant it can be shown that

$$\Delta = |I + 2\lambda^2\boldsymbol{a}_1\boldsymbol{a}_1^T + 2\boldsymbol{a}_2\boldsymbol{a}_2^T| \tag{28}$$

$$= 1 + 2\lambda^2 a_1^2 + 2a_2^2 + 4\lambda^2[a_1^2 a_2^2 - (\boldsymbol{a}_1^T\boldsymbol{a}_2)^2] \tag{29}$$

Thus

$$I'(\lambda) = \frac{4}{\pi}\frac{\boldsymbol{a}_1^T\boldsymbol{a}_2}{(1 + 2\lambda^2 a_1^2)\Delta^{1/2}}. \tag{30}$$

Using the substitution

$$\theta = \frac{2\lambda\boldsymbol{a}_1^T\boldsymbol{a}_2}{\sqrt{1 + 2\lambda^2 a_1^2}\sqrt{1 + 2a_2^2}} \tag{31}$$

we find

$$I'(\lambda) = \frac{2}{\pi}\frac{1}{\sqrt{1 - \theta^2}}\frac{d\theta}{d\lambda}. \tag{32}$$

Thus $I(\lambda) = 2/\pi \sin^{-1}\theta$ and

$$V_{\text{erf}}(\boldsymbol{z}, \boldsymbol{z}') = I(1) = \frac{2}{\pi}\sin^{-1}\frac{2\boldsymbol{a}_1^T\boldsymbol{a}_2}{\sqrt{1 + 2a_1^2}\sqrt{1 + 2a_2^2}}. \tag{33}$$

Finally, using $\tilde{\boldsymbol{z}}^T\tilde{\boldsymbol{z}} = a_1^2$, $\tilde{\boldsymbol{z}}^T\tilde{\boldsymbol{z}}' = \boldsymbol{a}_1^T\boldsymbol{a}_2$, $\tilde{\boldsymbol{z}}'^T\tilde{\boldsymbol{z}}' = a_2^2$, $\tilde{\boldsymbol{z}} = C^{-1}\tilde{\boldsymbol{x}}$ and $\tilde{\boldsymbol{z}}' = C^{-1}\tilde{\boldsymbol{x}}'$ we obtain equation 12.

Acknowledgements

I thank David Saad and David Barber for help in obtaining the result in equation 12, and David Barber, Chris Bishop, Peter Dayan, Ian Nabney, Radford Neal, David Saad and Huaiyu Zhu for comments on various drafts of the paper. I would also like to thank the anonymous referees for improvements to the paper as suggested by their comments. This work was partially supported by EPSRC grant GR/J75425, "Novel Developments in Learning Theory for Neural Networks".

References

Broomhead, D. and D. Lowe (1988). Multivariable functional interpolation and adaptive networks. *Complex Systems* **2**, 321–355.

Castro, P. E., W. H. Lawton, and E. A. Sylvestre (1986). Principal Modes of Variation for Processes With Continuous Sample Curves. *Technometrics* **28(4)**, 329–337.

Cressie, N. A. C. (1993). *Statistics for Spatial Data.* New York: Wiley.

Gibbs, M. and D. J. C. MacKay (1997). Efficient Implementation of Gaussian Processes. Draft manuscript, available from
`http://wol.ra.phy.cam.ac.uk/mackay/homepage.html`.

Girosi, F., M. Jones, and T. Poggio (1995). Regularization Theory and Neural Networks Architectures. *Neural Computation* **7(2)**, 219–269.

Hornik, K. (1993). Some new results on neural network approximation. *Neural Networks* **6** (8), 1069–1072.

Journel, A. G. and C. J. Huijbregts (1978). *Mining Geostatistics.* Academic Press.

MacKay, D. J. C. (1992). A Practical Bayesian Framework for Backpropagation Networks. *Neural Computation* **4(3)**, 448–472.

Matérn, B. (1980). *Spatial Variation* (second ed.). Springer. Lecture Notes in Statistics 36.

Moody, J. E. and T. S. Rögnvaldsson (1997). Smoothing regularizers for projective basis functions. In M. C. Mozer, M. I. Jordan, and T. Petsche (Eds.), *Neural Information Processing Systems, Vol. 9.* MIT Press.

Neal, R. M. (1996). *Bayesian Learning for Neural Networks.* New York: Springer. Lecture Notes in Statistics 118.

Parzen, E. (1962). *Stochastic Processes.* Holden-Day.

Poggio, T. and F. Girosi (1990). Networks for approximation and learning. *Proceedings of IEEE* **78**, 1481–1497.

Rasmussen, C. E. (1996). *Evaluation of Gaussian Processes and Other Methods for Non-linear Regression.* Ph.D. thesis, Dept. of Computer Science, University of Toronto. Available from `http://www.cs.utoronto.ca/~carl/`.

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory.* New York: Springer Verlag.

Wahba, G. (1990). *Spline Models for Observational Data.* Society for Industrial and Applied Mathematics. CBMS-NSF Regional Conference series in applied mathematics.

Wahba, G. (1996). RBF's, SBF's, TreeBF's, Smoothing Spline ANOVA: Representers and pseudo-representers for a dictionary of basis functions for penalized likelihood estimates. Slides presented at the NIPS workshop on Model Complexity, December 1996. Available from `http://www.stat.wisc.edu/~wahba/wahba.html`.

Whittle, P. (1963). *Prediction and regulation by linear least-square methods.* English Universities Press.

Williams, C. K. I. (1997). Computing with infinite networks. In M. C. Mozer, M. I. Jordan, and T. Petsche (Eds.), *Advances in Neural Information Processing Systems 9.* MIT Press.

Williams, C. K. I. and C. E. Rasmussen (1996). Gaussian processes for regression. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo (Eds.), *Advances in Neural Information Processing Systems 8*, pp. 514–520. MIT Press.

Yaglom, A. M. (1987). *Correlation Theory of Stationary and Related Random Functions Volume I:Basic Results.* Springer Verlag.

Zhu, H., C. K. I. Williams, R. J. Rohwer, and M. Morciniec (1997). Gaussian Regression and Optimal Finite Dimensional Linear Models. Technical Report NCRG/97/011, Aston University, UK. Available from `http://www.ncrg.aston.ac.uk/Papers/` .