Paper presented at the Mathematics of Neural Networks and Applications conference, Oxford, UK, July 1995. To appear in *Mathematics of Neural Networks: Models, Algorithms and Applications.* Eds. S W Ellacott, J C Mason and I J Anderson, Kluwer, 1997?

Regression with Gaussian Processes

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

July 1995

Abstract

The Bayesian analysis of neural networks is difficult because the prior over functions has a complex form, leading to implementations that either make approximations or use Monte Carlo integration techniques. In this paper I investigate the use of Gaussian process priors over functions, which permit the predictive Bayesian analysis to be carried out exactly using matrix operations. The method has been tested on two challenging problems and has produced excellent results.

1 Introduction

In the Bayesian approach to neural networks a prior distribution over the weights induces a prior distribution over functions. This prior is combined with a noise model, which specifies the probability of observing the targets $\boldsymbol{t}$ given function values $\boldsymbol{y}$, to yield a posterior over functions which can then be used for predictions. For neural networks the prior over functions has a complex form which means that implementations must either make approximations [4] or use Monte Carlo approaches to evaluating integrals [6].

As Neal [7] has argued, there is no reason to believe that, for real-world problems, 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[1]. He has also shown that 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.

Although infinite networks are one method of creating Gaussian processes, it is also possible (and computationally easier) to specify them directly using parametric forms for the mean and covariance functions. In this paper I investigate using Gaussian processes specified parametrically for regression problems[2], and demonstrate very good performance on the two test problems I have tried. The advantage of the Gaussian process formulation is that the integrations, which have to be approximated for neural nets, can be carried out exactly (using matrix operations) in this case. I also show that the parameters specifying the Gaussian process can be estimated from training data, and that this leads naturally to a form of "Automatic Relevance Determination" [4], [7].

[1]Large networks cannot be successfully used with maximum likelihood training because of the overfitting problem.

[2]By regression problems I mean those concerned with the prediction of one or more real-valued outputs, as compared to classification problems.

2 Prediction with Gaussian Processes

A stochastic process is a collection of random variables $\{Y(\boldsymbol{x}) | \boldsymbol{x} \in X\}$ indexed by a set X. Often X will be a space such as $\mathbb{R}$ for some dimension d, although it could be more general. 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 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 points will have a joint multivariate Gaussian distribution.

Below I consider Gaussian processes which have $\mu(\boldsymbol{x}) \equiv 0$. This is the case for many neural network priors [7], and otherwise assumes that any known offset or trend in the data has been removed. A non-zero $\mu(\boldsymbol{x})$ can be incorporated into the framework, but leads to extra notational complexity.

Given a prior covariance function $C_P(\boldsymbol{x}, \boldsymbol{x}')$, a noise process $C_N(\boldsymbol{x}, \boldsymbol{x}')$ (with $C_N(\boldsymbol{x}, \boldsymbol{x}') = 0$ for $\boldsymbol{x} \neq \boldsymbol{x}'$) and data $\mathcal{D} = ((\boldsymbol{x}_1, t_1), (\boldsymbol{x}_2, t_2), \ldots, (\boldsymbol{x}_n, t_n))$, the prediction for the distribution of Y corresponding to a test point $\boldsymbol{x}$ is obtained simply by marginalizing the $(n+1)$-dimensional joint distribution to obtain the mean and variance

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

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

where $[K_\alpha]_{ij} = C_\alpha(\boldsymbol{x}_i, \boldsymbol{x}_j)$ for $\alpha = P, N$, $\boldsymbol{k}_P(\boldsymbol{x}) = (C_P(\boldsymbol{x}, \boldsymbol{x}_1), \ldots, C_P(\boldsymbol{x}, \boldsymbol{x}_n))^T$ and $\boldsymbol{t} = (t_1, \ldots, t_n)^T$. $\sigma_{\hat{y}}^2(\boldsymbol{x})$ gives the "error bars" of the prediction. In the work below the noise process is assumed to have a variance σ_ν^2 independent of $\boldsymbol{x}$ so that $K_N = \sigma_\nu^2 I$.

The Gaussian process view provides a unifying framework for many regression methods. ARMA models used in time series analysis and spline smoothing (e.g. [10]) correspond to Gaussian process prediction with a particular choice of covariance function[3], as do generalized linear regression models ($y(\boldsymbol{x}) = \sum_i w_i \phi_i(\boldsymbol{x})$, with $\{\phi_i\}$ a fixed set of basis functions) for a Gaussian prior on the weights $\{w_i\}$. Gaussian processes have also been used in the geostatistics field (e.g. [3], [1]), and are known there as "kriging", but this literature has concentrated on the case where $\boldsymbol{x} \in \mathbb{R}^2$ or $\mathbb{R}^3$, rather than considering more general input spaces. Regularization networks (e.g. [8], [2]) provide a complementary view of Gaussian process prediction in terms of a Fourier space view, which shows how high-frequency components are damped out to obtain a smooth approximator.

2.1 Adapting covariance functions and ARD

Given a covariance function $C = C_P + C_N$, the log probability l of the training data is given by

$$l = -\frac{1}{2}\log\det K - \frac{1}{2}\boldsymbol{t}^T K^{-1}\boldsymbol{t} - \frac{n}{2}\log 2\pi \tag{3}$$

where $K = K_P + K_N$. If C has some adjustable parameters $\boldsymbol{\theta}$, then we can carry out a search in $\boldsymbol{\theta}$-space to maximize l; this is simply maximum likelihood estimation of $\boldsymbol{\theta}$ [4]. For example, in a d-dimensional input space we may choose

$$C(\boldsymbol{x}, \boldsymbol{x}') = v_0 \exp\left(-\sum_{i=1}^{d} \frac{w_i}{2}(x_i - x_i')^2\right) + v_1 \delta(\boldsymbol{x}, \boldsymbol{x}') \tag{4}$$

[3]Technically splines require generalized covariance functions.

[4]See section 4 for a discussion of the hierarchical Bayesian approach.

Method	No. of inputs	sum squared test error
Gaussian process	2	1.126
Gaussian process	6	1.138
MacKay	2	1.146
Neal	2	1.094
Neal	6	1.098

Table 1: Results on the robot arm task.

where v_0, v_1 and the $\{w_i\}$ are adjustable. In MacKay's terms [4] l is the log "evidence", with the parameter vector $\boldsymbol{\theta}$ roughly corresponding to his hyperparameters $\boldsymbol{\alpha}$ and β; in effect the weights have been exactly integrated out.

One reason for constructing a model with variable w's is to express the prior belief that some input variables might be irrelevant to the prediction task at hand, and we would expect that the w's corresponding to the irrelevant variables would tend to zero as the model is fitted to data. This is closely related to the Automatic Relevance Determination (ARD) idea of MacKay and Neal [5], [7].

3 Experiments with Gaussian Process prediction

Prediction with Gaussian processes and maximum likelihood training of the covariance function has been tested on two problems : (i) a modified version of MacKay's robot arm problem and (ii) the Boston housing data set.

For both datasets I used a covariance function of the form given in equation 4 and a gradient-based search algorithm for exploring $\boldsymbol{\theta}$-space; the derivative vector $\partial l/\partial \boldsymbol{\theta}$ was fed to a conjugate gradient routine with a line-search[5].

3.1 The robot arm problem

I consider a version of MacKay's robot arm problem introduced by Neal (1995). The standard robot arm problem is concerned with the mappings

$$y_1 = r_1 \cos x_1 + r_2 \cos(x_1 + x_2) \qquad y_2 = r_1 \sin x_1 + r_2 \sin(x_1 + x_2) \tag{5}$$

The data was generated by picking x_1 uniformly from [-1.932, -0.453] and [0.453, 1.932] and picking x_2 uniformly from [0.534, 3.142]. Neal added four further inputs, two of which were copies of x_1 and x_2 corrupted by additive Gaussian noise of standard deviation 0.02, and two further irrelevant Gaussian-noise inputs with zero mean and unit variance. Independent zero-mean Gaussian noise of variance 0.0025 was then added to the outputs y_1 and y_2. I used the same datasets as Neal and MacKay, with 200 examples in the training set and 200 in the test set.

The theory described in section 2 deals only with the prediction of a scalar quantity Y, so I constructed predictors for the two outputs separately, although a joint prediction is possible within the Gaussian process framework (see co-kriging, §3.2.3 in [1]). Two experiments were conducted, the first using only the two "true" inputs, and the second one using all six inputs.

[5]In fact the parameterization $\log \boldsymbol{\theta}$ was used in the search to ensure that the v's and w's stayed positive.

Procedure used	ave. squared test error
Guessing overall mean	84.4
Best result in Quinlan (1993)	10.9
Gaussian process	8.6
Neal (Bayesian network with 2 hidden layers)	6.5

Table 2: Results on the Boston housing data task.

For each experiment ten random starting positions were tried. The log(v)'s and log(w)'s were all chosen uniformly from [-3.0, 0.0], and were adapted separately for the prediction of y_1 and y_2. The conjugate gradient search algorithm was allowed to run for 100 iterations, by which time the likelihood was changing very slowly. Results are reported for the run which gave the highest probability of the training data, although in fact all runs performed very similarly. The results are shown in Table 1[6] and are encouraging, as they indicate that the Gaussian process approach is giving very similar performance to two well-respected techniques. All of the methods obtain a level of performance which is quite close to the theoretical minimum error level of 1.0. It is interesting to look at the values of the w's obtained after the optimization; for the y_2 task the values were 0.243, 0.237, 0.0650, 1.7×10^{-4}, 2.6×10^{-6}, 9.2×10^{-7}, and v_0 and v_1 were 7.920 and 0.0022 respectively. The w values show nicely that the first two inputs are the most important, followed by the corrupted inputs and then the irrelevant inputs.

3.2 Boston housing data

The Boston Housing data has been used by several authors as a real-world regression problem (the data is available from `ftp://lib.stat.cmu.edu/datasets`). For each of the 506 census tracts within the Boston metropolitan area (in 1970) the data gives 13 input variables, including per capita crime rate and nitric oxides concentration, and one output, the median housing price for that tract.

A ten-fold cross-validation method was used to evaluate the performance, as detailed in [9]). The dataset was divided into ten blocks of near-equal size and distribution of class values (I used the same partitions as in [9]). For each block in turn the parameters of the Gaussian process were trained on the remaining blocks and then used to make predictions for the hold-out block. For each of the ten experiments the input variables and targets were linearly transformed to have zero mean and unit variance, and five random start positions used, choosing the log(v)'s and log(w)'s uniformly from [-3.0,0.0]. In each case the search algorithm was run for 100 iterations. In each experiment the run with the highest evidence was used for prediction, and the test results were then averaged to give the entry in Table 2.

The fact that the Gaussian process result beats the best result obtained by Quinlan (who made a reasonably sophisticated application of existing techniques) is very encouraging. It was observed that different solutions were obtained from the random starting points, and this suggests that an hierarchical Bayesian approach, as used in Neal's neural net implementation and described in section 4, may be useful in further increasing performance.

[6]The bottom three lines of the table were obtained from [7]. The MacKay result is the test error for the net with highest "evidence".

4 Discussion

I have presented a Gaussian process framework for regression problems and have shown that it produces excellent results on the two test problems tried.

In section 2 I have described maximum likelihood training of the parameter vector $\boldsymbol{\theta}$. Obviously a hierarchical Bayesian analysis could be carried out for a model M using a prior $P(\boldsymbol{\theta}|M)$ to obtain a posterior $P(\boldsymbol{\theta}|\mathcal{D}, M)$. The predictive distribution for a test point and the "model evidence" $P(\mathcal{D}|M)$ are then obtained by averaging the conditional quantities over the posterior. Although these integrals would have to be performed numerically, there are typically far fewer parameters in $\boldsymbol{\theta}$ than weights and hyperparameters in a neural net, so that these integrations should be easier to carry out. Preliminary experiments in this direction with the Hybrid Monte Carlo method [7] are promising.

I have also conducted some experiments on the approximation of neural nets (with a finite number of hidden units) by Gaussian processes, although space limitations do not allow me to describe these here. Other directions currently under investigation include (i) the use of Gaussian processes for classification problems by softmaxing the outputs of k regression surfaces (for a k-class classification problem), and (ii) using non-stationary covariance functions, so that $C(\boldsymbol{x}, \boldsymbol{x}') \neq C(|\boldsymbol{x} - \boldsymbol{x}'|)$.

Acknowledgements

I thank Radford Neal and David MacKay for many useful discussions and for generously proving data used in this paper, Chris Bishop, Peter Dayan, Radford Neal and Huaiyu Zhu for comments on earlier drafts, George Lindfield for making his implementation of the scaled conjugate gradient search routine available and Carl Rasmussen for his C implementation which runs considerably faster than my MATLAB version. This work was supported by EPSRC grant GR/J75425.

References

[1] N. A. C. Cressie. *Statistics for Spatial Data.* Wiley, 1993.

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

[3] A. G. Journel and Ch. J. Huijbregts. *Mining Geostatistics.* Academic Press, 1978.

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

[5] D. J. C. MacKay. Bayesian Methods for Backpropagation Networks. In J. L. van Hemmen, E. Domany, and K. Schulten, editors, *Models of Neural Networks II.* Springer, 1993.

[6] R. M. Neal. Bayesian Learning via Stochastic Dynamics. In S. J. Hanson, J. D. Cowan, and C. L. Giles, editors, *Neural Information Processing Systems, Vol. 5*, pages 475–482. Morgan Kaufmann, San Mateo, CA, 1993.

[7] R. M. Neal. *Bayesian Learning for Neural Networks.* PhD thesis, Dept. of Computer Science, University of Toronto, 1995.

[8] T. Poggio and F. Girosi. Networks for approximation and learning. *Proceedings of IEEE*, 78:1481–1497, 1990.

[9] J. R. Quinlan. Combining Instance-Based and Model-Based Learning. In P. E. Utgoff, editor, *Proc. ML'93.* Morgan Kaufmann, San Mateo, CA, 1993.

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