THEORETICAL REPORT IFS-TR-029

THE USE OF BAYESIAN INFERENCE IN THE TRAINING AND INTERPRETATION OF NEURAL NETWORK PREDICTORS

ABSTRACT

The most common method used to train neural networks is the minimisation of the misfit error for a training set of examples using gradient descent based techniques. In the statistical sense, we can view this as a process by which the parameters of the neural network (the weights) are estimated from a sample of data (the training set) using the method of Maximum Likelihood.

However, as was discussed in Theoretical Reports IFS-TR-027 and IFS-TR-028, purely minimising the data misfit is not usually sufficient to ensure that the network is able to generalise well on unseen data. For this reason, regularisation constraints such as weight decay are usually added to the data misfit error during training to control the complexity of the mapping performed by the neural network.

In this report, we demonstrate how such regularisation techniques can be shown to have a Bayesian interpretation. In particular, we show how the method of weight decay (IFS-TR-028) is consistent with the assumption of a Gaussian prior on the weights of the network and a Bayesian approach to training. We also show how a Bayesian interpretation of the weight estimation process leads to the possibility of assigning error bars to the predictions made by a neural network.

We conclude this report with a discussion of some of the practicalities involved in the application of Bayesian approaches to neural network training and review the material contained in other Theoretical Reports that deal with such matters in more detail.

THE BAYESIAN APPROACH TO NEURAL NETWORKS

We shall consider the use of Bayesian techniques and their application to neural network predictions in two contexts. Firstly we shall examine the Bayesian interpretation of the training process of the neural networks. We shall show how minimisation of the data misfit error is in fact a **special case** of the Bayesian Framework.

Secondly, we shall examine the implication of the Bayesian model of neural network training with respect to the interpretation of the outputs of neural networks. In particular, we shall show how the Hessian of the error derivatives with respect to the weights may be used to assign uncertainties (error bars) on the predictions. We also show, how the Bayesian interpretation lends support to the idea of using committees of neural networks.

Framework

Consider the task of training a fixed neural network architecture, $\aleph$ on a set of training examples

$$D = \{\underline{x}_i, t_i, i = 1...N\} \tag{1}$$

where $\underline{x}_i$ is the i^{th} training input vector, t_i is the target output and N is the number of training patterns. We have assumed, for simplicity, that the target outputs are scalar, but they could equally well be multi-dimensional. In what follows, extension to multi dimensional targets is trivial.

In the standard method of training a neural network we try to find a set of weights $\underline{\omega}$, such that the network function $f(\underline{x}, \underline{\omega})$ minimises a least square error term E_D defined

$$E_D = \frac{1}{2}\sum_{i=1}^{N}\left(f(\underline{x}_i, \underline{\omega}_i) - t_i\right)^2 \tag{2}$$

If we assume that the target values t_i are in fact generated by a process with additive Gaussian noise then the set of weights that minimises the above will correspond to the Maximum Likelihood estimate of the weights, $\underline{\omega}_{ML}$.

In the Bayesian approach to network training, rather than finding a single *best* set of weights we attempt to find the probability distribution of the weights given the training data D, i.e. we wish to find

$$p(\underline{\omega} \mid D) = \frac{p(D \mid \underline{\omega})\, p(\underline{\omega})}{p(D)} \tag{3}$$

The denominator of the above is a weight independent normalisation factor and can be written in terms of the two densities contained in the numerator,

$$p(D) = \int p(D \mid \underline{\omega})\, p(\underline{\omega})\, d\underline{\omega} \tag{4}$$

So, in order to determine the density $p(\underline{\omega} \mid D)$, we need to obtain expressions for the densities $p(\underline{\omega})$ and $p(D \mid \underline{\omega})$.

Evaluating $p(D \mid \underline{\omega})$

Let us assume that the target functions are generated by an unknown process with additive, zero mean Gaussian noise with a variance of σ_n^2. We can then define the probability of observing a particular target t_i given an input vector $\underline{x}_i$ and particular set of network weights, $\underline{\omega}$ as

$$p(t_i \mid \underline{x}_i, \underline{\omega}) = \frac{1}{\sqrt{2\pi}\,\sigma_n} \exp\left(-\frac{\left(f(\underline{x}_i, \underline{\omega}) - t_i\right)^2}{2\sigma_n^2} \right) \tag{5}$$

We can therefore write the likelihood of the entire data set (assuming that the training examples are independent) as

$$\begin{aligned} p(D \mid \underline{\omega}) &= \prod_{i=1}^{N} p(t_i \mid \underline{x}_i, \underline{\omega}) \\ &= \frac{1}{\left(2\pi\sigma^2\right)^{\frac{N}{2}}} \exp\left(-\sum_{i=1}^{N} \frac{\left(f(\underline{x}_i, \underline{\omega}) - t_i\right)^2}{2\sigma_n^2} \right) \end{aligned} \tag{6}$$

Substituting Equation (2) into the above and using the standard notational conveniences of writing

$$\beta = \frac{1}{\sigma_n^2}, \tag{7}$$

and

$$Z_D(\beta) = \left(\frac{2\pi}{\beta} \right)^{\frac{N}{2}}, \tag{8}$$

we can re-write the Equation (6) as

$$p(D \mid \underline{\omega}) = \frac{1}{Z_D} \exp(-\beta E_D) \tag{9}$$

Clearly, for the above the maximum for the likelihood is obtained for a minimum value of E_D. Thus, standard neural network training techniques that minimise data misfit error, E_D are equivalent to maximum likelihood estimation.

Evaluating $p(\underline{\omega})$

This density represents a prior on the values of the weights in the network. It has been empirically shown by a number of researchers that the final weights of an unregularised, trained network tend to exhibit Gaussian distributions. Let us assume that the prior for the weights may be given by a zero-mean Gaussian with a standard deviation of σ_w^2. We can then write the density of the weights as

$$p(\underline{w}) = \frac{1}{\sqrt{2\pi}\sigma_w} \exp\left(-\sum_{i=1}^{W} \frac{\underline{\omega}_i^2}{2\sigma_w^2} \right) \tag{10}$$

where W is the total number of weights contained in the neural network and ω_i is the value of the i^{th} network weight. Using similar substitutions to those introduced above in the previous Section,

$$\alpha = \frac{1}{\sigma_w^2} \tag{11}$$

and

$$Z_W(\alpha) = \left(\frac{2\pi}{\alpha} \right)^{\frac{W}{2}} \tag{12}$$

we can re-write this as

$$p(\underline{\omega}) = \frac{1}{Z_W(\alpha)} \exp(-\alpha E_W) \tag{13}$$

where we have defined the *weight error*, E_W as the sum of the squares of the weights,

$$E_W = \frac{1}{2}\sum_{i=1}^{W} \omega_i^2 \qquad (14)$$

Combining the Densities

Combining expressions for the two densities above we can now write the posterior distribution for the weights,

$$p(\underline{\omega} \mid D) = \frac{\exp(-M(\underline{\omega}))}{\int \exp(-M(\underline{\omega}))d\underline{\omega}} \tag{15}$$

where $M(\underline{\omega})$ is defined

$$M(\underline{\omega}) = \beta E_D + \alpha E_W \tag{16}$$

Now that we have an expression for this posterior distribution, we can predict the target value, t of a new input vector, $\underline{x}$ by using the integral

$$p(t \mid \underline{x}) = \int p(t \mid \underline{\omega}, D)\, p(\underline{\omega} \mid D)\, d\underline{\omega} \tag{17}$$

The problem with the above expression is that it implies that we evaluate the target for **every possible set of weights**, $\underline{\omega}$. Networks will usually contain 100 plus weights, so clearly, it is impractical to evaluate the above integral, as it would involve performing a 100 plus dimensional integration!

What is usually done instead is to assume that the quantity $p(\underline{\omega} \mid D)$ is sharply peaked about some maximum value, $\underline{\omega}_{MAP}$. We can therefore approximate the above integral by evaluating it only about the immediate vicinity of the *maximum a posteriori* (MAP) estimate. We will discuss how to perform this integration in the next Section. First of all we shall discuss methods for finding the maximum *a posteriori* weights, $\underline{\omega}_{MAP}$.

Finding the MAP Weights

From Equation (15) we see that in order to find the maximum *a posteriori probability* $P(\underline{\omega} \mid D)$ *of the* weights $\underline{\omega}$, we must minimise the function

$$M(\underline{\omega}) = \beta E_D + \alpha E_W \tag{18}$$

with respect to the network weights, $\underline{\omega}$. Expanding the original terms of E_D and E_W we obtain

$$M(\underline{\omega}) = \frac{\beta}{2}\sum_{i=1}^{N}\left(f(\underline{x}_i, \underline{\omega}) - t_i\right)^2 + \frac{\alpha}{2}\sum_{i=1}^{W}\omega_i^2 \tag{19}$$

This function is familiar as the error function used to train neural networks with weight decay (IFS-TR-028). The weight decay parameter, λ is in this case given by

$$\lambda = \frac{\alpha}{\beta} \tag{20}$$

Thus, training a neural network using weight decay is, in fact, equivalent to attempting to find the MAP weights assuming a Gaussian prior on the weights and additive Gaussian noise on the target values. Furthermore, we are given an interpretation of the weight decay, λ as being the ratio of the expected variances of the noise on the targets of the training data and the prior of the weights.

Using the MAP Estimate to Make Predictions and Estimate Error-Bars

Suppose we have trained a neural network to minimise the function $M(\underset{\sim}{\omega})$ as discussed above. At the end of training we hope to have found the maximum a posteriori set of weights $\underset{\sim}{\omega}_{MAP}$. We now want to use this set of weights to provide predictions about a new input vector, $\underset{\sim}{x}$. Recall that, in order to do this we wish to evaluate the integral,

$$p(t \mid \underset{\sim}{x}, D) = \int p(t \mid \underset{\sim}{x}, \underset{\sim}{\omega}, D)\, p(\underset{\sim}{\omega} \mid D)\, d\underset{\sim}{\omega} \tag{21}$$

One option would be to assume that the value of $p(\underset{\sim}{\omega} \mid D)$ is 1.0 at the value $\underset{\sim}{\omega} = \underset{\sim}{\omega}_{MAP}$ and 0.0 for all other values of $\underset{\sim}{\omega}$ (i.e. it is a delta function). We would therefore simply use the output of the final trained network as an approximation to the above integral. This approach corresponds to the usual non-Bayesian approach to training neural networks using simple weight decay.

A more sophisticated approach is to assume that about some local neighbourhood of $\underset{\sim}{\omega}_{MAP}$ we can make a quadratic approximation to the function $M(\underset{\sim}{\omega})$, i.e.

$$M(\underset{\sim}{\omega}) = M(\underset{\sim}{\omega}_{MAP}) + \frac{1}{2}(\underset{\sim}{\omega} - \underset{\sim}{\omega}_{MAP})^T H (\underset{\sim}{\omega} - \underset{\sim}{\omega}_{MAP}) \tag{22}$$

where *H* is the Hessian of the regularised error function with respect to the weights. This can be written in terms of the Hessian of the unregularised error function,

$$H = \beta \nabla\nabla E_D + \alpha I \tag{23}$$

A number of techniques exist in the literature for the determination of the Hessian of the unregularised error function. For a summary, see IFS-TR-039.

Substituting the above expansion into Equation (21) and making a linear approximation for the output of the network about the MAP weights, i.e.

$$f(\underset{\sim}{x}, \underset{\sim}{\omega}_{MAP} + \Delta\underset{\sim}{\omega}) = f(\underset{\sim}{x}, \underset{\sim}{\omega}_{MAP}) + \underset{\sim}{g}^T \Delta\underset{\sim}{\omega} \tag{24}$$

where $\underset{\sim}{g}$ is the standard error gradient evaluated about $\underset{\sim}{\omega}_{MAP}$

$$\underset{\sim}{g} = \nabla_{\underset{\sim}{\omega}} y \Big|_{\underset{\sim}{\omega}_{MAP}} \tag{25}$$

we obtain the final expression for a prediction of the target , t for a new input vector $\underset{\sim}{x}$,

$$p(t \mid \underset{\sim}{x}, D) = \frac{1}{\sqrt{2\pi}\sigma_t} \exp\left(- \frac{(t - f(\underset{\sim}{x}, \underset{\sim}{\omega}_{MAP}))^2}{2\sigma_t^2} \right) \tag{26}$$

i.e. the probability of the predicted output is a Gaussian with expected mean equal to the output of the network with the MAP weights and variance given by

$$\sigma_t^2 = \frac{1}{\beta} + \underset{\sim}{g}^T H^{-1} \underset{\sim}{g} \tag{27}$$

So, in practical terms we use the network with the MAP weights to estimate the *expected* value of the target in the usual manner, but we can also associate with this estimate a measure of the uncertainty associated with this prediction.

Finding Suitable Values for α and β

So far, we have not discussed how to obtain suitable estimates for the so called *hyper parameters* α and β. We examine two possible techniques for this in Theoretical Report IFS-TR-031.

Summary

In this report, the following has been shown

- Training using the standard weight decay is consistent with training a neural network using Bayesian techniques with a Gaussian prior assumed for the distribution of the weights of the network.
- The Bayesian analysis allows us to understand more precisely the role of the regularisation constant, λ in the weight decay algorithm. Specifically it allows us to view λ as representing the ratio of the expected variances of the target outputs and the weight priors.

- It is possible, through the consideration of the Hessian matrix to calculate error bars or *uncertainties* in the output responses of the neural networks. This is particularly valuable in the case of financial time series prediction as it allows us to estimate the volatility of the predicted time series.

Author: Darren Toulson
Revision: v 1.00 7th March, 1997
See ALSO: Theoretical Reports IFS-TR-027, IFS-TR-028, IFS-TR-031, IFS-TR-039