ESANN'1997 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 16-17-18 April 1997, D-Facto public., ISBN 2-9600049-7-3, pp. 321-326

Extended Bayesian Learning

Steffen Gutjahr, Christian Nautze

University of Karlsruhe
Institute of Logic, Complexity and Deduction Systems
Am Fasanengarten 5, 76128 Karlsruhe
Email: {gutjahr,nautze}@ira.uka.de

Abstract. In Bayesian learning one represents the relative degree of believe in different values of the weight vector - including biases - by considering a probability distribution function over weight space. In general, this a priori probability is expected to come from a Gaussian with zero mean and flexible variance which is callded a hyperparameter. It can be optimized automatically during training by maximizing the evidence. The extended Bayesian learning (EBL) approach consists of considering a more general form of priors by using several weight classes and by considering the mean of the Gaussian distribution to be another hyperparameter. We propose an algorithm which determines automatically the optimal number of different weight classes and where the weights can change from one class to another. Our approach is applied in several benchmark problems and outperforms simple Bayesian learning as well as other optimization strategies.

1. Introduction

We begin by considering the problem of training a network in which the architecture is given. The task of learning a neural network is to find a set of connections $\vec{w}$ that minimizes a pre-defined error function. In the conventional maximum likelihood approach the mean squared error is used. In the Bayesian framework, however, we consider a probability distribution over weight values.

In the following we will give a brief introduction of the Bayesian learning of neural networks. Apart from the likelihood function we also introduce the new prior probability.

1.1. The Likelihood

The training set for the mapping to be learned is a set of N input-target pairs $D = \{x^m, t^m\}$. We assume that the target data is generated from a function with additive Gaussian noise, so that the probability of observing t^m would be

$$p(t^m|x^m, \vec{w}, \beta) \propto \exp(-\frac{\beta}{2}\{y(x^m; \vec{w}) - t^m\}^2)$$

where $y(x; \vec{w})$ represents the network function with weight vector $\vec{w}$ and $\frac{1}{\beta}$ is the presumed noise in the data.

ESANN'1997 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 16-17-18 April 1997, D-Facto public., ISBN 2-9600049-7-3, pp. 321-326

Provided the data points are drawn independently, the probability of the data, called the likelihood is

$$P(D|\vec{w},\beta) = \prod_{m=1}^{N} P(t^m|x^m,\vec{w}) = \frac{1}{Z_D(\beta)} \exp(-\frac{\beta}{2}\sum_{m=1}^{N}\{y(x^m;\vec{w}) - t^m\}^2) \quad (1)$$

where $Z_D(\beta) = (\frac{2\pi}{\beta})^{N/2}$ is the normalizing constant. The data error $E_D(\vec{w}) = \frac{1}{2}\sum_{m=1}^{N}\{y(x^m;\vec{w}) - t^m\}^2$ is used to express the likelihood function. Here $P(D|...)$ denotes the probability of the target. The input is part of the condition and is suppressed in the notation.

1.2. The Prior Probability

Up to now the prior was based on the believe that positive and negative weights are equally frequent and that smaller weights are more likely to occur than larger ones. In this context the Gaussian prior with zero mean is a natural description of such a compact distribution. For a general survey see [Tho93] and [Mac92].

Neural networks are often used when the relation between input vector and target is highly nonlinear. When using sigmoid activation functions large weights are needed to detect the nonlinearity in the data. Therefore it seems reasonable to drop down the assumption that larger weights are more unlikely than smaller ones. We rather believe that there must exist several large weights to adequately build up the given data.

Our approach consists of changing the a priori probability by keeping variable the mean of the Gaussian distribution and to consider it as a hyperparameter. Additionally, we split the W adjustable parameters $\vec{w}$ into G groups $\mathcal{W} = \{\mathcal{W}_g|g = 1,\ldots,G\}$ with W_g weights in group $\mathcal{W}_g$. By denoting $\alpha = (\alpha_1,\alpha_2,\ldots,\alpha_G)^T$ and $\mu = (\mu_1,\mu_2,\ldots,\mu_G)^T$ the prior is of the form

$$P(\vec{w}|\alpha,\mu) = \frac{1}{Z_{\mathcal{W}}(\alpha)} \exp(-\sum_{g=1}^{G}\frac{\alpha_g}{2}\sum_{w\in\mathcal{W}_g}(w-\mu_g)^2) \quad (2)$$

The hyperparameter α_g is in fact the reciproque of the standard deviation of the Gaussian of $\mathcal{W}_g$. The term $E_{\mathcal{W}_g}(\vec{w}) = \frac{1}{2}\sum_{w\in\mathcal{W}_g}(w-\mu_g)^2$ is a new form of regularization term centered at the nonzero point μ_g. Setting $\mu_g = 0\ \forall g$ yields to the well known weight decay. The normalization coefficient is $Z_{\mathcal{W}}(\alpha) = \prod_g \left(\frac{2\pi}{\alpha_g}\right)^{W_g/2}$.

1.3. The A Posteriori Probability

Inserting (1) and (2) into the Bayes' rule we get the posterior weight distribution

$$P(\vec{w}|D,\beta,\alpha,\mu) = \frac{1}{Z_M(\alpha,\beta,\mu)} \exp\{-\beta E_D - \sum_{g=1}^{G}\alpha_g E_{\mathcal{W}_g}\} \quad (3)$$

ESANN'1997 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 16-17-18 April 1997, D-Facto public., ISBN 2-9600049-7-3, pp. 321-326

where $Z_M(\alpha, \beta, \mu)$ is the normalizing constant. In order to maximize the a posteriori probability we have to minimize the exponent. This is learning in the mean square error sense using a generalized version of weight decay terms.The minimization of the error function is carried out with the RPROP learning algorithm [RB93].

2. A Generalization of the Evidence

Up to now we developed the algorithm for optimizing the weights given the values of α, μ and β, which we call hyperparameters. The advantage of the Bayesian approach of learning is that the hyperparameters can be evaluated automatically. This is done by optimizing the evidence for (β, α, μ) which can be written as

$$P(D|\beta, \alpha, \mu) = \frac{Z_M(\alpha, \beta, \mu)}{Z_W(\alpha) Z_D(\beta)}.$$

For an excellent review see [Bis96]. We already evaluated $Z_W(\alpha)$ and $Z_D(\beta)$. The evaluation of Z_M is much more difficult analytically. Therefore we suggest that we trained the neural network to its most probable weight value $\vec{w}_{MP}$ with fixed hyperparameters. If we make a Gaussian approximation for the posterior distribution of the weights around $\vec{w}_{MP}$ we get an approximation of the form

$$Z_M(\beta, \alpha, \mu) \approx \exp\{-\beta E_D(\vec{w}_{MP}) - \sum_{g=1}^{G} \alpha_g E_{W_g}(\vec{w}_{MP})\} (2\pi)^{W/2} ||A||^{-1/2}$$

where $A = \beta \nabla\nabla E_D(\vec{w}_{MP}) + \sum_g \alpha_g I_g$ is the Hessian matrix of the total error function evaluated at $\vec{w}_{MP}$. Note that A does not depend on μ. I_k is a matrix whose elements are all zero, except for some elements on the leading diagonal I_{kk} where i corresponds to the weight from group W_k. We finally compute the log of the evidence to be

$$\begin{aligned} \log P(D|\beta, \alpha, \mu) &= -\beta E_D(\vec{w}_{MP}) - \sum_{g=1}^{G} \alpha_g E_{W_g}(\vec{w}_{MP}) - \frac{1}{2} \log ||A|| \\ &\quad + \sum_g \frac{W_g}{2} \log \alpha_g + \frac{N}{2} \log \beta - \frac{N}{2} \log(2\pi). \end{aligned} \tag{4}$$

We consider the problem of finding the maximum with respect to the hyperparameters. This is done by differentiating (4) and setting the resulting equations to zero.

We briefly describe the algorithm of finding the maximum with respect to α and β.

By using $V^T A V = D$ with $V = (v_1, \ldots, v_W)$ be the orthogonal matrix of the eigenvectors of A and $D = \text{diag}\ (\lambda_1, \cdots, \lambda_W)$ be the matrix of the eigenvalues of A we can show that $\frac{\partial}{\partial \alpha} \log ||A|| = \text{Trace}\{A^{-1} \frac{\partial}{\partial \alpha} A\}$. Straightforward calculations yield for the following conditions which have to be satisfied when

ESANN'1997 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 16-17-18 April 1997, D-Facto public., ISBN 2-9600049-7-3, pp. 321-326

the log evidence is maximized with respect to β and α_k.

$$2\beta E_D(\vec{w}_{MP}) = N - \gamma \qquad 2\alpha_g E_{\mathcal{W}_g}(\vec{w}_{MP}) = \gamma_g \tag{5}$$

where $\gamma = \sum \gamma_g$ and $\gamma_g = \sum_j \{\frac{\lambda_j - \alpha_g}{\lambda_j}(V^T I_g V)_{jj}\}$.

Additionally, in our approach we have to maximize the evidence with respect to μ_g.

$$\frac{\partial}{\partial \mu_g} P(D|\beta, \alpha, \mu_g) = \alpha_g \frac{\partial}{\partial \mu_g} E_{\mathcal{W}_g}(\vec{w}_{MP}) = \alpha_g \sum_{w \in \mathcal{W}_g} (w - \mu_g) \stackrel{!}{=} 0$$

So the best value for μ_g is the mean of the weights of the corresponding weight group, meaning

$$\mu_g = \frac{1}{W_g} \sum_{w \in \mathcal{W}_g} w \tag{6}$$

The Hessian A is calculated exactly as in [Bis93] and the resulting matrix is used to determine the eigenvectors V and the eigenvalues λ_j, see also [PFTV88].

3. Optimization of the Weight Classes

So far we considered the membership of each weight to its weight group as fixed. This means that one has to determine the number and the weight components of the weight decay terms in advance which may be not optimal. For solving this problem we invented an algorithm that automatically determines the number of useful Gaussian priors and their members.

By fixing the hyperparameters α_g and $\mu_g, g = 1, \ldots, G$ we define G different prior probabilities. The weight vector is set to his optimal value $\vec{w}_{MP}$ depending of those priors. The main idea is that the value of the weight component is interpreted as the probability of membership to the current weight groups. If this probability is higher for another weight group than the actual weight group, the weight changes into this one.

Figure 1 clarifies the situation by considering 3 different Gaussian priors. We can see that $\mathcal{W}_1$ is near the classical weight decay, meaning that μ_1 is near zero. $\mathcal{W}_2$ prefers positive weights whereas $\mathcal{W}_3$ prefers negative ones. Weight w is assumed to be a member of $\mathcal{W}_1$ but has a very high value. It would be much more probable in this case to be a member of weight group $\mathcal{W}_2$. This can be done by simply changing the weight group and train the neural network with this changed a priori distribution. In the course of the algorithm the number of priors reduces when several weight groups get empty. By this the optimal number of weight groups is found automatically.

4. Simulations

The weight groups were formed by putting all weights exiting from the same input in one group, all weights from hidden to output units in one group and all biases in one group. The training process consists of the following steps:

ESANN'1997 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 16-17-18 April 1997, D-Facto public., ISBN 2-9600049-7-3, pp. 321-326

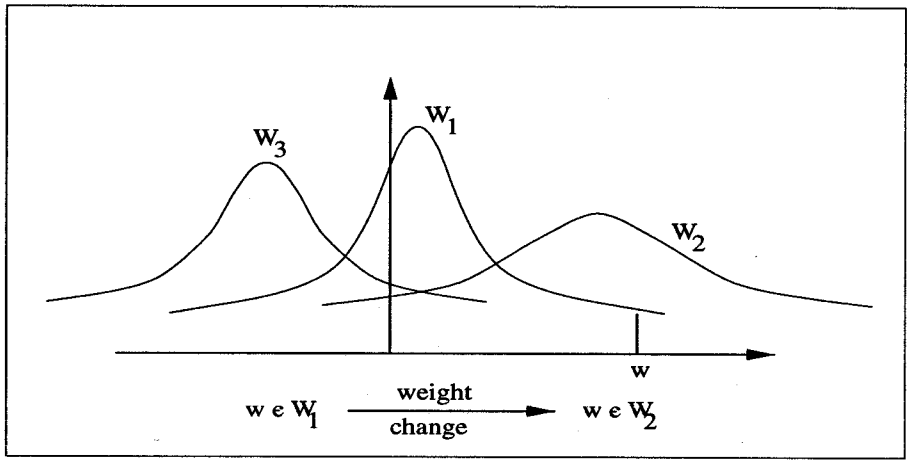

Figure 1: *Weight change algorithm. After learning the weight w changes from weight class $\mathcal{W}_1$ to weight class $\mathcal{W}_2$ because the prior is more likely with regard to w.*

1. Initialize α_g with some small starting value. Set $\mu_g = 0$ and $\beta = 1$ and determine the weights according to their prior distribution.
2. Train the network for some epochs by maximizing (3).
3. Update parameters α_g, μ_g and β using (5) and (6)
4. IF the hyperparameters have not converged GOTO 2
   ELSE make a weight change
5. IF no weight change their group STOP algorithm
   ELSE GOTO 2

Negative Eigenvalues were omitted while updating the parameters, see [Bis96]. If there is a group $\mathcal{W}_g$ with all $\lambda_j < 0$, then $\alpha_g = W_g/(\sum_{w \in \mathcal{W}_g} w^2)$.

5. Results

We test the framework on two problems. The first task is the speaker independent recognition of 11 vowels. [Rob89]) did a comparative study of different classifiers where the nearest neighbour was best with a hit rate of 56 %.

We take 528 pattern for training and 462 test patterns, no validation data is used. The fully connected network topology consists of 10 input units, 3 hidden units and 11 output units. By using classical Bayesian learning we get a hit rate of 52.3 %. This is slightly better than the feedforward network in [Rob89]) but does not outperform the nearest neighbour classifier. The EBL approach yields a hit rate of 58.5 %. This is better than the best result of [Rob89]) and is an improvement to classical Bayesian learning of more than 6 %.

The next table shows the distribution of the weights. Finally we get 2 weight groups. $\mathcal{W}_1$ consists of almost 95 % of the parameters and the mean is close to zero. However the 12 weights in group $\mathcal{W}_2$ have a very small mean value of -1186.75 and the variance is immense, i.e. α_2 is near zero.

	$\mathcal{W}_g$	γ_g	$E_{\mathcal{W}_g}$	α_g	μ_g
g=1	217	162.826	3575.65	0.0227688	-0.314492
g=2	12	0.0229098	9.15892e+07	1.25068e-10	-1186.75

This is a typical result of our EBL approach. We end up in two or three different weight groups where one of them represents the classical weight decay group, meaning the mean is near zero. Generally more than 75 % of the weights are included in this group. The remaining groups comprise very big weights which obviously represent the nonlinearity of the problem. The algorithm shows stable results, i.e. by using different initial weights the resulting neural network perform quite similar and the performance is crucially better than comparable approaches.

The second task is the classification of thyroid disease with 3772 training and 3428 test patterns. Every pattern comprise 21 continues inputs and 3 binary targets.

	MLP	BL - MLP	EBL - MLP
hit rate (training)	100.0	99.9	100.0
hit rate (testing)	98.1	98.9	99.2

	W_g	γ_g	$E_{\mathcal{W}_g}$	α_g	μ_g
g=1	5	5.0077e-06	1.20123e+08	2.0844e-14	-4327.38
g=2	36	7.35848	322327	1.14146e-05	16.2066
g=3	87	1.53297	0.135045	5.67576	0.0109744

References

[Bis93] Christopher M. Bishop. Exact calculation of the hessian matrix for the multilayer perceptron. *Neural Computation*, 4(4):494–501, 1993.

[Bis96] Christopher M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, 1996.

[Mac92] D. J. C. MacKay. Bayesian interpolation. *Neural Computation*, 4(3):415–417, 1992.

[PFTV88] William H. Press, Brian P. FLannery, Saul A. Teukolsky, and William T. Vetteriling. *Numerical Recipes in C. The Art of Scientific Computing.* University Press, Cambridge, 1988.

[RB93] Martin Riedmiller and Heinrich Braun. A direct adaptiv method for faster backpropagation learning: The RPROP algorithm. In *Proceedings of the IEEE International Conference on Neural Networks*, 1993.

[Rob89] A. J. Robinson. *Dynamic Error Propagation Networks.* PhD thesis, Cambridge University Engineering Department, 1989.

[Tho93] H. H. Thodberg. Ace of bayes: application of neural networks with pruning. Technical Report 1132E, The Danish Meat Research Institut, 1993.