Approximate Algorithms for Neural-Bayesian Approaches.

Tom Heskes * Bart Bakker Bert Kappen

SNN, University of Nijmegen, The Netherlands

Abstract

We describe two specific examples of neural-Bayesian approaches for complex modeling tasks: survival analysis and multitask learning. In both cases, we can come up with reasonable priors on the parameters of the neural network. As a result, the Bayesian approaches improve their (maximum likelihood) frequentist counterparts dramatically. By illustrating their application on the models under study, we review and compare algorithms that can be used for Bayesian inference: Laplace approximation, variational algorithms, Monte Carlo sampling, and empirical Bayes.

Key words: neural networks, Bayesian inference, survival analysis, multitask learning, Monte Carlo sampling, variational approximation
PACS: 02.50.Ng, 02.70.Lg, 07.05.Mh

1 Introduction

Feedforward neural networks, also called multi-layered perceptrons, have become popular tools for solving complex prediction and classification problems.

* Corresponding author. Address: SNN Nijmegen, Geert Grooteplein 21, 6525 EZ, Nijmegen, The Netherlands.
Email address: tom@mbfys.kun.nl (Tom Heskes).
URL: http://www.mbfys.kun.nl/people/tom (Tom Heskes).

More and more people start to realize that there is nothing magical about neural networks: they are "just" nonlinear models, not principally different from many others. The so-called weights are the parameters of the model. The error function one tries to minimize can often be interpreted as (minus) the loglikelihood of the data given the parameters. The famous backpropagation rule is nothing but an efficient way to compute the gradient of this error function.

This insight has led to a welcome cross-fertilization between neural network research and advanced statistics. Neural networks are now standardly trained with better optimization procedures as Levenberg-Marquardt and conjugate gradient. Frequentist tools such as bootstrapping and cross-validation generate ensembles of neural networks with much better and more reliable performance than single ones. Algorithms have been developed for computing confidence and prediction intervals (errorbars) on the network outcomes.

The introduction of Bayesian methodology for neural networks has been another important advance. Training became Bayesian inference and popular strategies such as "weight decay" could be interpreted in terms of Bayesian priors. Now there seems to be a status-quo between the frequentist and Bayesian approaches for training standard multi-layered perceptrons. The Bayesian approach seems to be more principled and elegant, but the frequentist one more practical, yielding about the same performance.

In this article we describe two examples of multi-layered perceptrons solving a specific problem: survival analysis and multitask learning. We will see that in both cases the Bayesian approach is definitely better than its frequentist alternative. The reason is that in these specific cases we can come up with sensible priors. These priors specify reasonable assumptions about the weights of the neural network. The Bayesian inference machine can be used to infer the appropriate strength of these priors. In most cases, however, exact Bayesian inference is intractable and we have to resort to approximations. We highlight and compare the approximations that are currently in use.

In section 2 we describe our neural network model for survival analysis. Section 3 treats appropriate approximations for Bayesian inference of the parameters of this model: Monte Carlo sampling, the variational approach, and the Laplace approximation. In section 4 the different approaches are applied and compared on a large real-world database involving patients with ovarian cancer. Our model for multitask learning is introduced in section 5. Section 6 discusses empirical Bayes, also called "evidence framework", an approximation that seems particularly suited for multitask learning. Results obtained on a huge dataset involving magazine sales are described in section 7. We end with a discussion and conclusions in section 8. Detailed mathematics is treated in the Appendix.

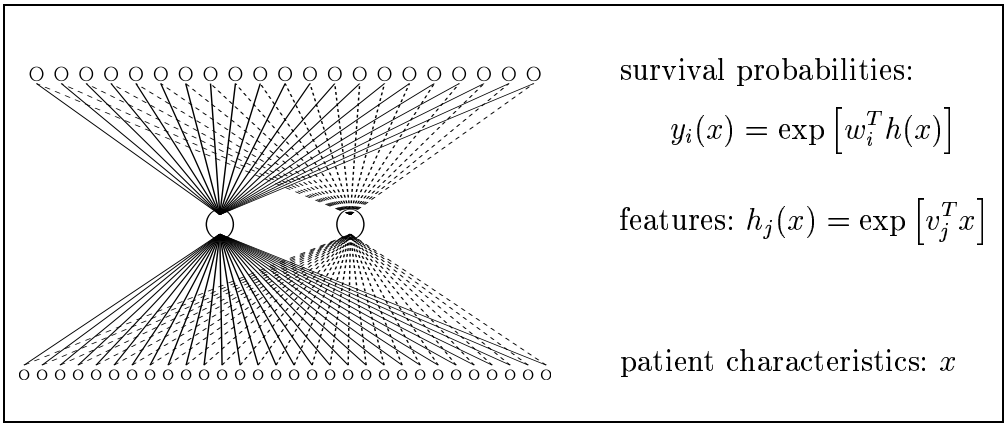

Fig. 1. Neural interpretation of survival analysis. See the text for an explanation.

2 Multi-layered perceptrons for survival analysis

2.1 The model and the likelihood

The purpose of survival analysis is to estimate a patient's chances of survival as a function of time, given the available medical information at the time the patient is admitted to the study. This information (e.g., age, given treatment, outcome of medical tests) is quantified by an n_{inp}-dimensional input vector x. Time is discretized into n equal time intervals of length Δt. The output y_i stands for the (estimated) probability to survive a time t_i after entering the study. It is modeled through a multi-layered perceptron with exponential transfer functions as sketched in Figure 1:

$$y_i(x) = \exp\left[\sum_j^{n_{\text{hid}}} w_{ji} h_j(x)\right] \quad \text{and} \quad h_j(x) = \exp\left[\sum_k^{n_{\text{inp}}} v_{kj} x_k\right] . \tag{1}$$

In [1] it is shown that the case of a single hidden unit corresponds to a discretized version of Cox proportional hazards model [2], which is the standard in the field of survival analysis. $h_j(x)$ is called the proportional hazard and only depends on the patient's characteristics; w_j is the baseline hazard, a function of time t, here discretized as t_i. By adding hidden units (the dashed lines in Figure 1), we can in principle extend our model beyond proportional hazards to allow for more complex input-output relationships.

Our database D consists of a set of N patients with characteristics x^μ and corresponding times t^μ, the time that the patient leaves the study after entering it, either because the patient dies (uncensored) or because the study ended when the patient was still alive (censored). In computing the likelihood of the data, we have to make this distinction between censored and uncensored

patients. We have

$$P(D|v,w) = \prod_{\nu \in \text{uncensored}} f_{i(\nu)}(x^\nu) \prod_{\mu \in \text{censored}} y_{i(\mu)}(x^\mu) \,, \tag{2}$$

with $i(\mu)$ such that $t_{i(\mu)-1} < t^\mu \leq t_{i(\mu)}$, and

$$f_i(x) = -\frac{1}{\Delta t} \sum_j [w_{ji} - w_{j,i-1}] h_j(x) y_i(x) \,,$$

the probability density for a patient with characteristics x to die between t_{i-1} and t_i ($f_i(x)$ is minus the derivative of $y_i(x)$ with respect to time).

Standard Cox now corresponds to a maximum likelihood (ML) approach: try and find the parameters v and w that maximize the likelihood of the data D. An advantage of Cox analysis is that the optimal parameters v of the proportional and w of the time-dependent hazard can be found sequentially. Disadvantages of this approach are the hazard's tendency to become highly non-smooth, and the danger of strongly overfitting the data. Here we suggest a Bayesian approach to overcome these weaknesses.

2.2 Sensible priors

In a Bayesian approach we seek to construct a probability distribution over all possible values of the parameters, here v and w. This distribution not only depends on the data D, but also on prior knowledge about the nature of the problem. This prior knowledge can be expressed as prior probabilities on the model parameters. Using Bayes' formula the priors and the data likelihood are combined into a posterior distribution.

For ease of notation, we consider the case of one hidden unit and make a transformation of variables by defining $q_i = \log(w_i - w_{i-1})$ and $q_1 = \log(w_1)$. The first prior

$$P(q|\gamma) \propto \exp\left[-\frac{\gamma}{2} \sum_{ij} g(|i-j|)[q_i - q_j]^2\right] \propto \exp\left[-\frac{\gamma}{2} q^T \Gamma q\right] \,,$$

with $\Gamma_{ij} = -g(|i-j|), \Gamma_{ii} = \sum_{j \neq i} g(|i-j|)$, and $g(x) = \mathrm{e}^{-x^2/\tau}$, prevents the hazard from becoming too sharp as a function of time. Since $P(q|\gamma)$ assigns the highest likelihood to a hazard function that is constant in time (independent of i), it smoothes out the hazard function, and introduces a preference for survivor functions which decay exponentially. Its effect is visualized in Figure 2. The hazard function in the ML Cox approach is a jagged function, due to the limited information in the database. After incorporating the prior $P(q|\gamma)$

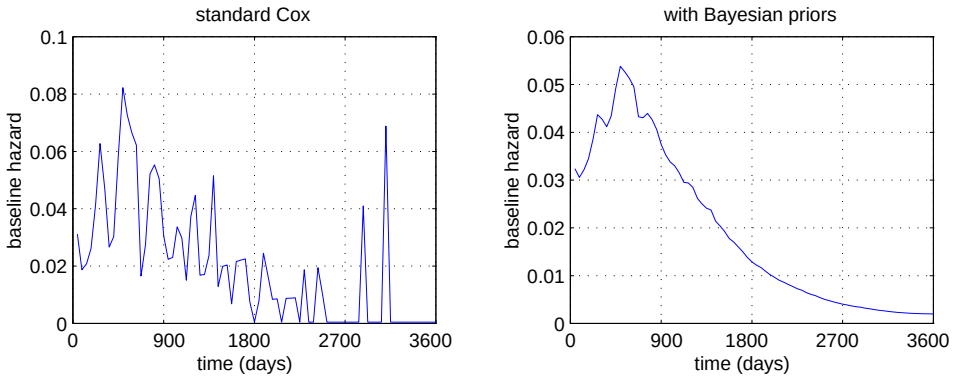

Fig. 2. The baseline hazard w_i as obtained in maximum likelihood Cox analysis (left), and the maximum *a posteriori* solutions resulting from the Bayesian approach incorporating the prior on q (right). In both cases w_i is optimized on a training set of 600 patterns, chosen randomly from our database. The effect of the prior is a considerable smoothing of the hazard function.

this function becomes much more smooth. This smoother function is not only more plausible *a priori*, but, as we will see below, also improves the model's predictive performance.

The second prior

$$P(v|\lambda) \propto \exp\left[-\frac{\lambda}{2} v^T \Lambda v\right] \quad , \quad \text{where} \quad \Lambda = \frac{1}{N} \sum_{\mu} x^{\mu} x^{\mu T} \; ,$$

prevents large activities of hidden units (high values for the proportional hazard), i.e., prefers small weights. This prior corresponds to a ridge-type estimator, as discussed in [3]. Incorporation of the input covariance matrix Λ makes this preference independent of a (linear) scaling of the inputs x. The effect of imposing this prior on the parameters v is illustrated in Figure 3. It can be seen that in the unrestrained case (left panel) it occurs quite often that the proportional hazard of one patient is upto ten times larger than it is for another patient, clearly indicating that the model is overfitting on the training data. After incorporating the prior on v, these differences become much more reasonable.

λ and γ are called hyperparameters. They express the confidence we have in the knowledge expressed in the two prior distributions. Since we do not want to specify the exact values of λ and γ, we introduce gamma distributions $P(\lambda|\sigma,\tau) \propto \lambda^{\sigma-1} \exp(-\tau\lambda)$ and a similar term for $P(\gamma)$ for the hyperparameters. Here the ratio $\frac{\sigma}{\tau}$ signifies the value for λ we deem most likely *a priori*. The ratio $\frac{\tau^2}{\sigma}$ measures the strength of this belief. In the rest of this article we assume the hyperhyperparameters $\{\sigma,\tau\}$ (one set for each of the two prior distributions) fixed and given and we do not incorporate them explicitly in our notation.

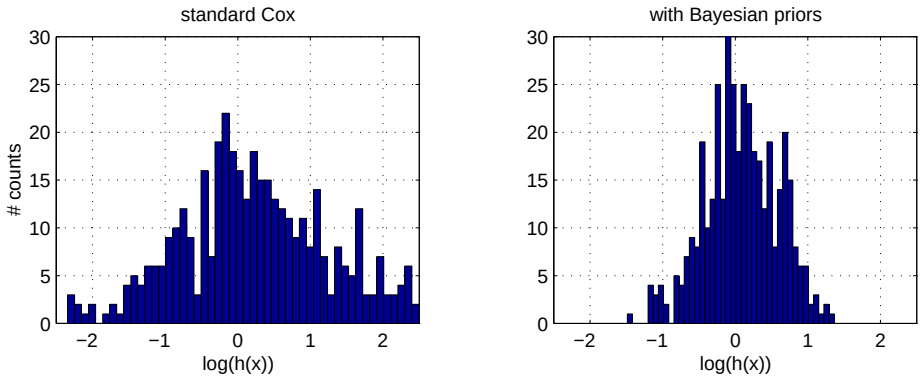

Fig. 3. Histograms of the (log of the) proportional hazard $h(x)$ as obtained in maximum likelihood Cox analysis (left), and for the maximum *a posteriori* solutions resulting from the Bayesian approach incorporating the prior on v. In both cases the parameters v of the proportional hazard are optimized on a training set of 600 patterns, chosen randomly from our database. The effect of the prior is a considerable "shrinking" of the proportional hazard.

2.3 Bayes' formula

The posterior distribution of the parameters $W = \{v, q\}$ and hyperparameters $\Omega = \{\lambda, \gamma\}$ given the data now follows from Bayes' formula:

$$P(W, \Omega | D) = \frac{P(D|W)P(W|\Omega)P(\Omega)}{P(D)}\,, \tag{3}$$

with $P(D|W)$ the likelihood as in (2) and $P(D)$ an irrelevant normalizing constant.

The posterior $P(W|D)$ follows from integrating out the hyperparameters Ω. In theory, this posterior is all we need to make predictions for new patients. However, since the integral cannot be done analytically, we have to make approximations.

3 Approximations of the posterior

Fortunately, an ample arsenal of methods to approximate $P(W|D)$ is available. These methods have become popular tools in the neural network community. In this section we describe three such methods: Monte Carlo sampling, a variational approach and the Laplace approximation.

Since the posterior $P(W, \Omega|D)$ is not a simple analytic function of the model parameters, it is hard to draw samples from it. We can use a combination of Gibbs sampling and Hybrid Markov Chain Monte Carlo Sampling (HMCMC) to make it tractable.

Gibbs sampling is used to iterate between drawing samples for the model parameters W and for the hyperparameters Ω. As we will see, drawing hyperparameters is straightforward for two reasons.

- The probability of the hyperparameters Ω given a particular sample W and the data D is independent of the latter:

$$P(\Omega|W, D) = \frac{P(D, W|\Omega)P(\Omega)}{P(D, W)} = \frac{P(D|W)P(W|\Omega)P(\Omega)}{P(D|W)P(W)} = P(\Omega|W)\,.$$

- The resulting distributions $P(\lambda|v)$ and $P(\gamma|q)$ are again gamma distributions. In statistical terms, the gamma prior is the natural conjugate of the Gaussian. Standard tricks to sample from a gamma distribution can be found in any textbook on Bayesian analysis (see e.g. [4]).

Now, given a new set of hyperparameters Ω, we have to draw a new W from

$$P(W|\Omega, D) = \frac{P(D|W)P(W|\Omega)}{P(D|\Omega)} \propto P(D|W)P(W|\Omega)\,,$$

the product of the likelihood and the priors, with fixed values for the hyperparameters.

The data likelihood $P(D|W)$ and priors $P(W|\Omega)$ are easy to compute; the problem is in the normalization $P(D|\Omega)$. Monte Carlo sampling circumvents computations of this normalization term. A Monte Carlo sampling procedure works as follows.

(1) Starting from a sample W use the "jumping distribution" $J(W'|W)$ to generate a candidate sample W. In its simplest version, the jumping distribution is symmetric, i.e., $J(W'|W) = J(W|W')$.
(2) Compute the ratio
$$r = \frac{P(W'|\Omega, D)}{P(W|\Omega, D)}\,.$$
Note that computation of this ratio is doable, since the normalizations of the separate probabilities drop out.
(3) Accept the new state W' if $r \geq 1$ or with probability r if $r < 1$. Otherwise keep the previous state W. Return to the first step.

The essence of a good Monte Carlo sampling algorithm is in the "jumping distribution": you want to have a distribution that makes large steps in the space of your parameters, such that subsequent samples are more or less independent, with a high acceptance rate.

A very popular sampling algorithm is Hybrid Markov Chain Monte Carlo sampling (see e.g. [5]). The basic idea is to double the parameter space, introducing an extra canonical parameter for each original one. Jumps are defined on both original and canonical parameters, such that the joint probability of the candidate sample is, up to numerical errors, equal to that of the one started from: the candidate has a high acceptance rate. Since the canonical parameters and the original ones are by construction independent, sampling of the joint distribution also corresponds to sampling of the distribution of the parameters we are interested in: the values of the canonical ones are simply neglected.

A disadvantage of Monte Carlo sampling is that generating (independent) samples can be quite time-consuming. Even more important, it is difficult to tell when sufficient samples have been drawn. It might for example be that one is sampling in just one part of the parameter space, with a very small probability of jumping to another relevant part. Advantages are that Monte Carlo sampling algorithms, in the limit of an infinite number of samples and except for singular cases, converge to the exact probability distribution. Furthermore, they are easy to implement for many probability distributions.

3.2 Variational approach

The variational approach offers an alternative. It has been introduced under the term "ensemble learning" in [6] and has been applied to learning in multilayered perceptrons and radial basis function networks in [7,8].

Recall that we are interested in the joint posterior $P(W, \Omega|D)$ of model parameters and hyperparameters. Knowing that we cannot describe it in an analytical form, the best we can do is to try and approximate it with an analytical distribution. Here we take

$$P^*(W, \Omega) = Q(W)R(\Omega)\,,$$

with [1] $Q(W) = \phi(W|\hat{W}, C)$, a Gaussian distribution, and $R(\Omega)$ for the moment unspecified.

[1] $\phi(W|\hat{W}, C) \propto \exp\left[-\frac{1}{2}(W - \hat{W})^T C^{-1}(W - \hat{W})\right]$.

A natural distance between the probabilities $P(W, \Omega|D)$ and $P^*(W, \Omega)$ is the Kullback-Leibler divergence

$$KL[Q, R] = \int dW d\Omega Q(W) R(\Omega) \log \left[\frac{Q(W)R(\Omega)}{P(W, \Omega|D)} \right]$$
$$= \langle \log [Q(W)R(\Omega)] \rangle_{Q,R} - \langle \log [P(D|W)P(W|\Omega)P(\Omega)] \rangle_{Q,R} \ , \quad (4)$$

where in the second line we substituted (3) and neglected irrelevant constants. The goal of the variational approach is now to find the parameters of the distribution $Q(W)$ and the distribution $R(\Omega)$ that minimize this distance.

The steps to be taken are quite similar to those in the Gibbs algorithm explained in the previous section. Suppose first that we know the distribution $Q(W)$ and would like to find the optimal $R(\Omega)$. Collecting the terms in (4) that depend on $R(\Omega)$, we obtain

$$KL[R] = \langle \log R(\Omega) \rangle_R - \langle \log P(\Omega) \rangle_R - \langle \log P(W|\Omega) \rangle_{Q,R} \ . \quad (5)$$

Note that, as in the Gibss sampling procedure, none of the terms involves the data D. Furthermore, we have the same kind of "conjugacy": with a Gaussian prior $P(W|\Omega)$ and a gamma distribution for $P(\Omega)$, the optimal $R(\Omega)$ is also gamma distributed. Its average $\bar{\Omega}$ only depends on the parameters $\bar{W}$ and C of the distribution $Q(W)$ (see the Appendix for details).

Next we assume that $R(\Omega)$ is known and try to optimize the parameters of $Q(W)$. The terms in (4) depending on $Q(W)$ and thus on the variational parameters $\{\hat{W}, C\}$ are

$$KL[Q] = \langle \log Q(W) \rangle_Q - \langle \log P(D|W) \rangle_Q - \langle \log P(W|\Omega) \rangle_{Q,R} \ . \quad (6)$$

In the Appendix we show that, for our survival analysis model, all these terms can be computed analytically as a function of the data D and the average $\bar{\Omega}$ for the hyperparameters. Computing the new $\bar{W}$ and C boils down to a straightforward optimization procedure, to be solved e.g. using a conjugate gradient method.

Iterating back and forth as in the Gibbs sampler, the variational procedure converges to an (at least locally) optimal distribution $P^*(W, \Omega)$. The advantage of the variational approach is that we arrive at a relatively simple distribution to work with. Furthermore, in the above application to survival analysis all computations to arrive at this distribution can be done analytically. In most cases however, numerical integrations are unavoidable (see e.g. [7] for the case of multi-layered perceptrons with sigmoidal transfer functions).

Also here, with more than one hidden unit, we have to resort to numerical integrations, scaling with the number of added hidden units.

3.3 Laplace approximation

The variational procedure can be simplified by replacing the minimization of the Kullback-Leibler divergence $KL[Q]$ by a Laplace approximation. That is, instead of fitting the parameters of $Q(W)$ against the distribution $P(W|\bar{\Omega}, D)$, we can take the Laplace approximation

$$\hat{W} = \operatorname*{argmax}_W P(W|\bar{\Omega}, D)$$

and C the Hessian of $-\log(P(W|\bar{\Omega}, D))$. Based on these new parameters $\hat{W}$ and C, new values for can be computed as in (10).

An advantage of the Laplace approximation over the full variational approach is that it does not require the evaluation of integrals, just minimization, and is therefore often easier to compute. One would expect the variational approach to be more accurate, since it not only considers the peak of the probability distribution, but might also take into account some of its mass. Note that what we call Laplace approximation is somewhat different from the "evidence framework" introduced by MacKay (see e.g. [9,10]). We discuss the evidence framework in more detail below.

4 Results for survival analysis

We test the standard ML approach and the three (approximate) Bayesian approaches on a database of 929 ovarian cancer patients (see [11] for details). The database is randomly divided in a training and test set. As an error criterion we take minus the loglikelihood on the test set, i.e., the logarithm of (2), with μ running over the test set. To get a clear indication of the relative strengths of the methods, we scale the errors relative to a "minimum error" $E_{\min}$ and the "Cox error" E_{cox}:

$$E_{\text{rel}} = \frac{E - E_{\min}}{E_{\text{cox}} - E_{\min}} \,. \tag{7}$$

The minimum error is defined as the test error of the maximum likelihood Cox solution when the ML parameters are optimized on this same test set. The Cox error is the test error of the ML solution computed by optimizing on the training set. So, a relative error of 1 means just as good as the ML Cox

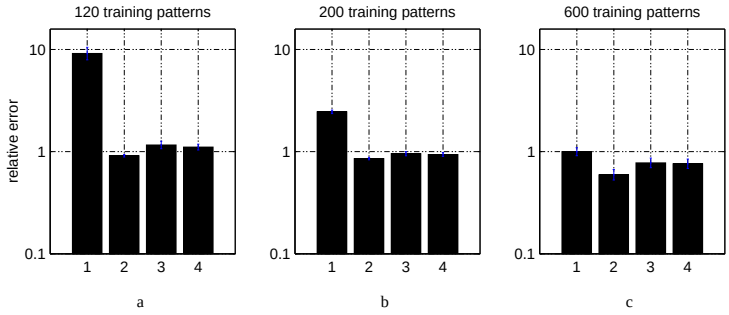

Fig. 4. Relative error after training on three partitions of the database (120 patients in the left panel, 200 patients in the middle panel and 600 patients in the right panel). In each panel, from left to right the bars represent the error in: the maximum likelihood Cox method (1), the HMCMC sampling approach (2), the variational approach (3), the Laplace approximation (4), all with one hidden unit. All differences are significant, except for the one between the variational approach and the Laplace approximation.

method. A relative error of 0.5 means twice as close to the error of the ML solution on the test set.

Let us first consider the results for a training set of 600 patients and testing on the remaining 329 (Figure 4c). The errors in any of the approximations to the Bayesian posterior are significantly ($p \approx 1 \times 10^{-5}$) smaller than the error in the ML Cox approach. Looking more closely at the difference between the three Bayesian approaches, the error in the variational approach happens to be slightly (but significantly) larger than the error in the sampling approach. The Laplace approximation, which takes about an equal amount of computation time as the variational approach, does not perform significantly better or worse than the variational approach.

The size of our database (929 patients) is much larger than common in survival analysis (typically 100-200 patients). Therefore, we also compared the four approaches when applied on smaller parts (120 and 200 patients), results of which are shown in Figures 4a and b. Note that we used the values $E_{\rm cox}$ and $E_{\rm min}$ obtained on the training set of 600 patterns to define the relative errors. For lower and lower numbers of training patterns, the error in the ML Cox method increases dramatically. The error in the Bayesian approaches also increases slightly, but much more gradually: the less data, the higher the impact of the priors, which here yield an enormous improvement.

We notice a similar effect when considering more complex models. After adding an extra hidden unit, which doubles the number of model parameters, the error corresponding to the ML solution increases dramatically, even with a training set of 600 patients. The Bayesian solution, obtained through HMCMC, is about

as accurate as the one for the model with a single hidden unit. Apparently a model with a single hidden unit is sufficiently complex for the database under study. The Bayesian approach again manages to avoid overfitting when extra complexity is introduced.

The comparison made here, with the complete model considering all inputs, may however not be a totally "fair" one. In the medical statistical community it is well-known that Cox analysis with a full set of inputs strongly suffers from overfitting. A standard procedure to decrease the overfitting problem (and thus the error) in Cox analysis, is to reduce the number of inputs to the model, e.g., through backward elimination.

With backward elimination, the least relevant input is eliminated iteratively. One way to compute the relevance of an input is by considering the Bayes factor, the ratio between the likelihood of the data with the model excluded divided by the likelihood of the model with the input included (see e.g. [9,12]). The Bayes factor can be easily estimated based on the approximate distributions that result from the variational and Laplace approach. We will not go into the exact procedures here, but simply show the results in Figure 5. It can be seen that this procedure indeed has a wholesome effect: in both the ML Cox method and the Bayesian approach the test error decreases when the least relevant inputs are removed. The decrease of test error in the ML Cox method is larger than in the variational approach, since in the latter most of the overfitting problem has already been eliminated by the Bayesian priors. However, even after reduction the Bayesian approach still yields significantly better results than ML Cox. The variational approach and the Laplace approximation yield similar results. Backward elimination in combination with a Monte Carlo sampling is much more involved and has not been tested.

5 A Bayesian approach to multitask learning

5.1 Multitask learning

A neural-Bayesian approach is ideal for multitask learning. In multitask learning we are dealing with many related tasks. The hope is that the tasks can "learn from each other", for example by sharing parameters. A typical example, also studied in this article, is the use of a feedforward neural network with part of the weights shared and others specific to each task. Training the network on all tasks, the risk of overfitting the shared part is reduced and a common set of features can be obtained. This idea has been studied and tested on practical problems in e.g. [13] and [14].

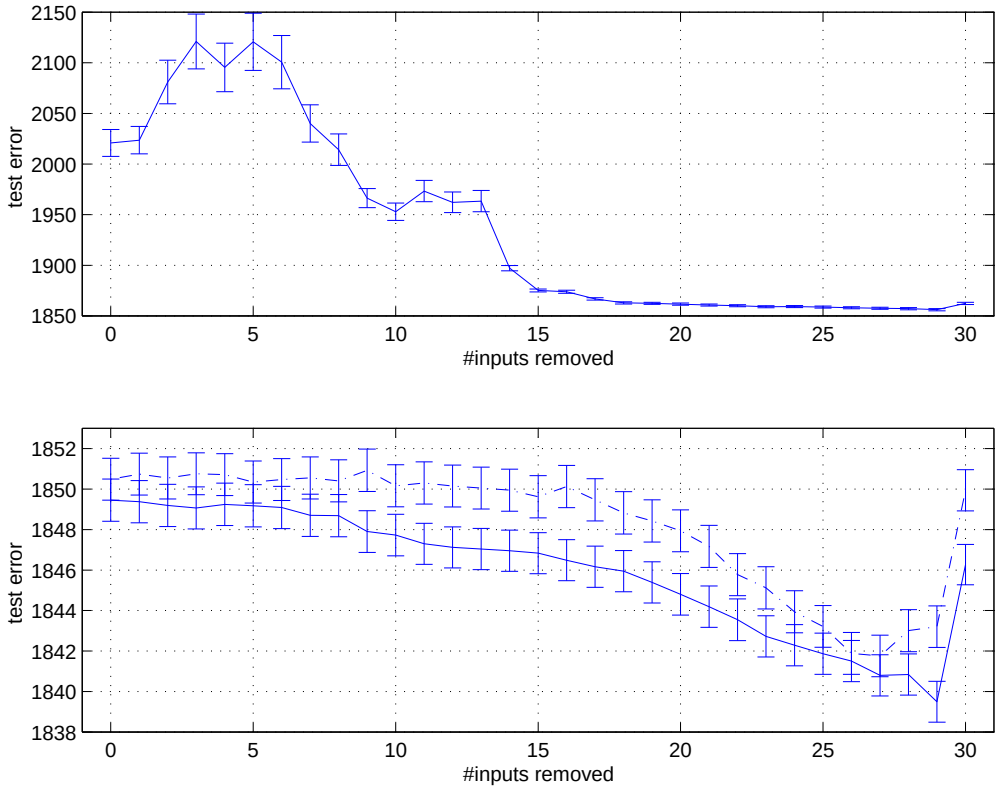

Fig. 5. The test error (minus the log-likelihood of the data in the test set under the current model) as a function of the number of removed input parameters, for the maximum likelihood Cox method (upper), the variational approach (lower, solid line) and the Laplace approximation (lower, dotted line). At zero, thirty one inputs are left in the model, at thirty, just one. The test error is the average error over 25 runs, conducted on parameters obtained from random choices of training sets, each containing 600 patterns.

Baxter [15] proposed hierarchical Bayesian inference as a model for studying multitask learning. Parameters that are shared between tasks are treated at a higher level than the task-specific parameters. Here we will give a concrete implementation of this idea: a huge neural network for solving many related

regression tasks.

5.2 The network architecture

The model that we study is sketched in Figure 6. It has n outputs, each corresponding to a different task i. Given the input vector x, the output $y_i(x)$ follows from [compare with (1)]

$$y_i(x) = \sum_j^{n_{\rm hid}} w_{ji} h_j(x) \quad \text{and} \quad h_j(x) = g\left(\sum_k^{n_{\rm inp}} v_{kj} x_k\right) \ ,$$

with $g(\cdot)$ the hidden unit's transfer function. The $n_{\rm inp} \times n_{\rm hid}$ matrix v with the weights from the input to the hidden units is shared by all tasks. The $n_{\rm hid} \times n$ matrix w, with column vectors w_i, contains the weights from the hidden to the output units specific to each task. Typically we have $n_{\rm hid} < n_{\rm inp} \ll n$, i.e., a bottleneck of hidden units. The idea behind multitask learning is that after training the input-to-hidden weights v implement a low dimensional (feature) representation of the inputs that is useful for all tasks. The risk of overfitting v is small since the data for all tasks can be used to infer this part of the model.

Our total database D consists of n sets D_i, one for each task, each containing N combinations of input vectors $x^\mu(i)$ and observed outputs y_i^μ. We assume that the observed outputs y_i^μ are given by the outputs $y_i(x^\mu(i))$ of the neural network, corrupted with Gaussian noise of zero mean and standard deviation σ, independent of i. The likelihood of all data D_i for task i given the weights v and w_i thus reads

$$P(D_i|w_i, v, \sigma) = \prod_\mu \phi(y_i^\mu | w_i^T g(v^T x^\mu(i)), \sigma^2) \ ,$$

with $\phi(\cdot|\cdot,\cdot)$ the normal distribution defined in section 3.2. The likelihood of all data is the product

$$P(D|w, v, \sigma) = \prod_i P(D_i|w_i, v, \sigma) \ .$$

5.3 Priors

The weights w_i determine the impact of the hidden units on output i. We consider the Gaussian prior

$$P(w_i|m, \Sigma) = \phi(w_i|m, \Sigma^2) \ ,$$

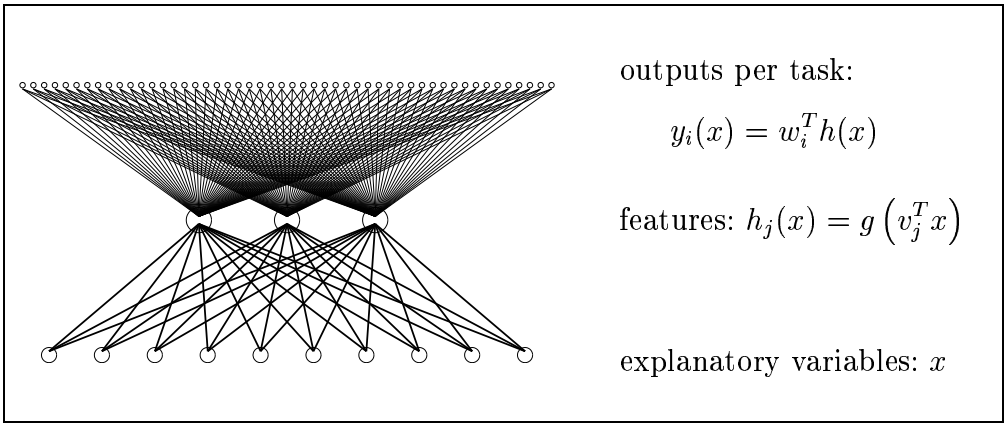

Fig. 6. Sketch of the architecture of the neural network for multitask learning. Each output corresponds to a different task. Input-to-hidden weights are shared; hidden-to-output weights specific.

with m a vector of length n_{hid} and Σ^2 an $n_{\text{hid}} \times n_{\text{hid}}$ covariance matrix. This corresponds to a so-called exchangeability assumption (the same m for all tasks) and introduces a tendency for similar weights across tasks. How similar is determined by the covariance matrix Σ^2.

In a full hierarchical Bayesian analysis, we should also specify priors for the parameters σ and v and even "hyperpriors" for the parameters m and Σ. In the empirical Bayesian approach, to be discussed next, we simply take an improper "flat" prior, giving equal *a priori* weight to all possible values.

6 Empirical Bayes for multitask learning

6.1 Task-specific parameters and shared parameters

In the (single-task) survival analysis case, we divided our total set of parameters into two sets: hyperparameters Ω and model parameters W. Hyperparameters Ω specified the prior distribution for the model parameters W, but had no direct effect on the data. The resulting conditional independence $P(\Omega|W, D) = P(\Omega|W)$ simplified both the Gibbs sampling and the variational approach considerably.

We could make the same subdivision here, but the resulting procedures would be extremely time-consuming. The empirical Bayesian approach introduced below distinguishes between parameters $W = \{w\}$ that are task-specific and parameters $\Omega = \{v, \sigma, m, \Sigma\}$ that are shared between tasks. With this subdivision we lose the conditional independence in $P(\Omega|W, D)$: $P(\Omega|W, D) \neq P(\Omega|W)$. Instead we will make the assumption that the posterior probability of the shared parameters is sharply peaked around its maximum *a posteriori*

(MAP) solution. This approximation is particularly suited for the multitask learning situation (see also e.g. [16] for similar approximations in the field of multilevel analysis), but can also be used in other contexts.

6.2 Empirical Bayes

We are interested in the probability $P(W, \Omega|D)$ of all parameters given all data and write

$$P(W, \Omega|D) = P(W|\Omega, D)P(\Omega|D) \,.$$

Since all data can be used to infer the probability of the shared parameters Ω, we make the assumption that $P(\Omega|D)$ is very sharply peaked around its maximum *a posteriori* (MAP) value Ω^*:

$$P(\Omega|D) \approx \delta(\Omega - \Omega^*) \quad \text{with} \quad \Omega^* = \operatorname*{argmax}_{\Omega} P(\Omega|D) \,, \tag{8}$$

The probability for the task-specific parameters W then reads

$$P(W|D) = \int d\Omega \; P(W|\Omega, D)P(\Omega|D) \approx P(W|\Omega^*, D) \,.$$

This is called the empirical Bayesian approach (see e.g. [17]). We have a two-stage procedure: first compute the MAP value of the shared parameters, then the probability of the task-specific parameters given this MAP solution.

In the multitask learning case, (8) seems to be a reasonable assumption. It can be shown that the distribution $P(\Omega|D)$ scales like $\exp\left[-NnE(\Omega)\right]$, with n the number of tasks, N the number of patterns per task, and $E(\Omega)$ an "error function" of order 1. In single-task learning, on the other hand, the scaling is more like $\exp\left[-NE(\Omega)\right]$, similar to the scaling of $P(W|D)$. However, for a hyperparameter specifying the prior distribution of m model parameters, the term $E(\Omega)$ is of the order m. So, with m sufficiently large, the assumption that $P(\Omega|D)$ is sharply peaked around its MAP solution may not be too far off. This is the reasoning behind the "evidence framework" [9]. The variational approach outlined in section 3.2, if necessary with the additional Laplace approximation discussed in section 3.3, seems simpler and more elegant, yielding roughly equivalent procedures [10].

6.3 Direct computation of the posterior

Empirical Bayes in combination with a flat prior $P(\Omega) \propto 1$ is in fact a maximum likelihood approach at the level of the shared parameters: the optimal

parameters Ω^* follow from

$$\Omega^* = \operatorname*{argmax}_{\Omega} P(\Omega|D) = \operatorname*{argmax}_{\Omega} P(D|\Omega)P(\Omega) = \operatorname*{argmax}_{\Omega} P(D|\Omega) \,.$$

The likelihood of the data given the shared parameters follows by integrating out the task-specific parameters

$$P(D|\Omega) = \int dW \; P(D, W|\Omega) = \int dW \; P(D|W, \Omega)P(W|\Omega) \,.$$

In the multitask learning situation introduced above, where the data for each task is considered independent given the shared parameters and the task-specific parameters, this further simplifies to

$$P(D|\Omega) = \prod_i \int dW_i \; P(D_i|W_i, \Omega)P(W_i|\Omega) \,, \tag{9}$$

with D_i the data corresponding to task i and W_i the corresponding task-specific parameters. In the above case, both the likelihood and the prior term are (unnormalized) Gaussians in W_i. The integral can be computed analytically, see (11) in the Appendix. Computing the ML parameters Ω^* becomes a nasty optimization problem.

6.4 EM algorithm

For slightly more complex models (e.g., non-Gaussian noise or nonlinear transfer function for the outputs), the integrals in (9) can no longer be computed analytically. But even if they can, as in our case, direct optimization of (9) may not be the smartest thing to do. The variational approach, outlined in section 3.2, provides an alternative. Here we take as an approximating distribution

$$P^*(W, \Omega) = Q(W)R(\Omega) \quad \text{with} \quad R(\Omega) = \delta(\Omega - \Omega^*) \,.$$

The (parameters specifying the) distribution $Q(W)$ and Ω^* have to be optimized such that the KL-divergence between the approximation and the true posterior is $P(W, \Omega|D)$ is minimized.

Minimization of the part $KL[R]$ in (5) given the current distribution $Q(W)$ now amounts to finding

$$\begin{aligned} \Omega^* &= \operatorname*{argmax}_{\Omega} \int dW \; Q(W) \log P(W, \Omega|D) \\ &= \operatorname*{argmax}_{\Omega} \int dW \; Q(W) \log P(D|W, \Omega)P(W|\Omega) \,. \end{aligned}$$

Substituting $R(\Omega) = \delta(\Omega - \Omega^*)$, the term $KL[Q]$ of (6) reads

$$KL[Q] = \int dW\, Q(W) \log \left[\frac{P(W|\Omega^*, D)}{Q(W)} \right] .$$

If we leave $Q(W)$ completely free, we directly obtain $Q(W) = P(W|\Omega^*, D)$. This corresponds to a standard EM algorithm for optimizing (9): in the E-step we compute the expectation of the "hidden variables" W given the current parameters Ω^*; in the M-step we use this probability to optimize for the next set of shared parameters. As can be seen in the Appendix, the two separate expectation and optimization steps are much simpler and easier to interpret than the direct integration of (9). Furthermore, by restricting $Q(W)$ to be of a specific form, we can obtain useful approximations when the exact integral is undoable. More on the link between variational approaches and EM algorithms can be found in [18].

7 Results for multitask learning

We illustrate multitask learning in combination with a Bayesian approach on a huge data set involving the sales of weekly magazines. Each task corresponds to a different outlet. The nine inputs taken into account include recent sales figures, holiday and price information, and season. Sales figures are normalized per task to have zero mean and unit variance. Magazine and newspapers sales is extremely noisy and thus difficult to predict. Since the same problem reoccurs week after week on a huge number of outlets, any performance improvement is significant. The technical difficulty is to avoid the risk of overfitting, while still being able to consider several input variables.

In our multitask learning model we have implemented two mechanisms for avoiding the risk of overfitting: a bottleneck of hidden units reducing the inputs to a smaller set of typical features, and, on top of that, regularization of the hidden-output weights through the specification of a prior. The obvious question is whether we really need both.

We test this on our dataset containing about 3 years of sales data for 1000 outlets. In each run, we subdivide the outlets into two sets of $n = 500$ outlets each. The available data for each outlet is further subdivided into a training and a test set, both containing $N = 92$ patterns. The ML parameters Ω^* are obtained by maximizing the probability of the data $P(D|\Omega)$ as in (11) [2], with

[2] The actual model is slightly different from the one explained here, with an extra shared parameter ρ that accounts for correlations between the tasks. The transfer function of the hidden units is linear and the inputs for each outlet are scaled such that the input covariances matrices are the same for all tasks. This makes

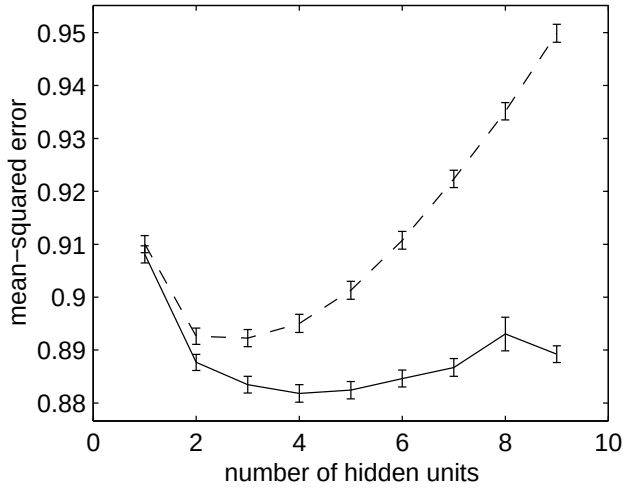

Fig. 7. Mean-squared test errors as a function of the number of hidden units for the maximum *a posteriori* (solid lines) and maximum likelihood (dashed) solutions. Error bars give the standard deviation of the mean. Averages are over 20 runs. See the text for further details.

D the training data for the first set of outlets. Based on these ML parameters Ω^*, we compute the MAP weights w_i maximizing $P(w_i|D_i, \Omega^*)$, with D_i the training set of outlet i in the second set of outlets. Performance is measured by the mean-squared error between the predictions based on these MAP weights and the actual sales figures, averaged over all test patterns. The results for different numbers of hidden units, with average and error bars computed over 20 runs, are given by the solid line in Figure 7. The dashed lines are obtained through exactly the same procedure, but then considering the ML weights, i.e., the weights that result from a flat prior instead of the prior $P(w_i|m, \Sigma)$ that is used to compute the MAP weights.

The results for the MAP solutions are clearly better than for the ML solutions, especially with increasing number of hidden units. With n_{hid} equal to n_{inp} (here 9), there is effectively no bottleneck. Loosely speaking, the ML test error for $n_{\text{hid}} = n_{\text{inp}}$ is the test error for single-task learning: the test error that we would obtain without making an attempt to "let the tasks learn from each other". A bottleneck of hidden units clearly helps, also for the MAP solutions. We conclude that on this dataset both incorporating prior information and trying to extract common features help a lot to gain a better performance.

the optimization procedures orders of magnitude faster, facilitating the extensive simulations that are needed to produce the results in Figure 7. See [19] for details.

8 Conclusion

The two examples treated in this article proof the usefulness of Bayesian analysis for inference in neural networks. In both cases, we could come up with priors that make sense, leaving the exact setting of the corresponding hyperparameters to the Bayesian machinery. We would argue that it is in these specific cases, where the nature of the problem suggests relevant prior information, that the advantage of more involved Bayesian technologies over "standard" frequentist approaches is most prominent. If no such *a priori* information is available, there may still be a principled or pragmatic preference for either a Bayesian or a frequentist approach, but the resulting performance should hardly matter.

Throughout this article, we discussed and compared different approximations when exact Bayesian inference is intractable. Which one is most appropriate depends on the problem at hand. Monte Carlo sampling is very general but might take too long. The approximate posterior obtained with the variational approach is much easier to work with, but may not be sufficiently accurate. Furthermore, the required calculations are not always (analytically) doable, in which case the Laplace approximation can be of help. The empirical Bayesian approach is similar to the "evidence framework" and is appropriate when there is reason to assume that the posterior distribution of some parameters (in multitask learning the ones that are shared between all tasks) is sharply peaked.

Appendix

The variational approach for survival analysis

In this Appendix we give the expressions for applying the variational approach to our model for survival analysis. We try to approximate the exact posterior $P(W, \Omega|D)$ with a distribution of the form $P^*(W, \Omega) = Q(W)R(\Omega)$. For ease of notation, we further assume that the covariance matrix C of the Gaussian $Q(W)$ is block-diagonal, i.e.,

$$C = \begin{pmatrix} C_{vv} & \emptyset \\ \emptyset & C_{qq} \end{pmatrix},$$

and similarly, that $R(\Omega) = R(\lambda)R(\gamma)$.

The optimization of the function $R(\Omega)$ given the current parameters $\bar{W}$ and

C is given in [7]. The resulting distributions $R(\lambda)$ and $R(\gamma)$ both happen to be gamma distributions. The average $\bar{\lambda}$ reads

$$\bar{\lambda} = \langle\lambda\rangle_R = \left(\frac{n_{\rm inp}}{2} + \sigma\right)\left[\frac{1}{2}\bar{v}^T\Lambda\bar{v} + \frac{1}{2}{\rm Tr}\ (C_{vv}\Lambda) + \tau\right]^{-1} , \qquad (10)$$

and similarly, with n instead of $n_{\rm inp}$ and Γ instead of Λ, for γ.

Computation of the functional (6) is more involved. We treat the terms one by one and neglect irrelevant constants. Straightforward manipulations yield ($|C|$ stands for the determinant of the matrix C)

$$\begin{aligned} -\langle\log Q(W)\rangle_Q &= \frac{1}{2}\log|C| = \frac{1}{2}\log|C_{vv}| + \frac{1}{2}\log|C_{qq}| \\ -\langle\log P(v|\lambda)\rangle_{Q,R} &= -\left\langle\log P(v|\bar{\lambda})\right\rangle_Q = \frac{\bar{\lambda}}{2}\left[\hat{v}^T\Lambda\hat{v} + {\rm Tr}\ (C_{vv}\Lambda)\right] , \end{aligned}$$

and a similar expression for $\langle\log P(w|\gamma)\rangle_{Q,R}$. The term $\langle\log P(D|W)\rangle_Q$ for the data loglikelihood can again be decomposed into two terms, see (2): a term involving only uncensored patients and a term to which all patients contribute. Both contributions can be computed analytically, starting from (2) and the transformation

$$w_i = \sum_{i'=1}^{i} {\rm e}^{q_{i'}} .$$

The uncensored patients ν contribute terms of the form

$$\left\langle\log\left[{\rm e}^{q_i}{\rm e}^{v^T x^\nu}\right]\right\rangle_Q = \hat{q}_i + \hat{v}^T x^\nu \text{ for } t_{i-1} < t^\nu \leq t_i .$$

For both censored and uncensored patients we get contributions

$$-\left\langle{\rm e}^{q_i}{\rm e}^{v^T x^\mu}\right\rangle_Q = -\exp\left[\hat{v}^T x^\mu + \hat{q}_i + \frac{1}{2}x^{\mu T}C_{vv}x^\mu + \frac{1}{2}C_{q_iq_i}\right] .$$

Here we used the equality

$$\int dy\ \phi(y|m,\Sigma^2){\rm e}^{y^T z} = \exp\left[m^T z + \frac{1}{2}z^T\Sigma^2 z\right] ,$$

where we substitute $\{v,q\}$ for y and the vector $\{x,[\ldots,0,0,1,0,0,\ldots]\}$, with 1 at the position of i, for z.

Mathematics of multitask learning

Computation of the likelihood $P(D_i|\Omega)$ boils down to the evaluation of Gaussian integrals, since both $P(D_i|W_i,\Omega)$ and $P(W_i|\Omega)$ are, up to normalization

constants, Gaussians in W_i. With definitions

$$C_i = \left\langle hh^T \right\rangle_i = \frac{1}{N} \sum_\mu h(x^\mu(i)) h^T(x^\mu(i)) \; ,$$

the covariance matrix of the hidden unit activities, and

$$\hat{w}_i = C_i^{-1} \left\langle hy \right\rangle_i = C_i^{-1} \frac{1}{N} \sum_\mu h(x^\mu(i)) y_i^\mu \; ,$$

the maximum likelihood solutions of the weights, we arrive at

$$\begin{aligned} -\log P(D_i|\Omega) &= \frac{N - n_{\rm hid}}{2} \log \sigma^2 + \frac{1}{2} \log |C_i| + \frac{1}{2} \log \left| \Sigma^2 + \left(\frac{NC_i}{\sigma^2} \right)^{-1} \right| \\ &+ \frac{N}{2\sigma^2} \left[\left\langle y^2 \right\rangle_i - \hat{w}_i^T C_i \hat{w}_i \right] + \frac{1}{2} (\hat{w}_i - m)^T \left[\Sigma^2 + \left(\frac{NC_i}{\sigma^2} \right)^{-1} \right]^{-1} (\hat{w}_i - m). \quad (11) \end{aligned}$$

Note that both $\hat{w}_i$ and C_i are in fact functions of the input-to-hidden weights v. Direct optimization of (11) happens to be extremely unstable.

In the optimization step of the variational (EM) approach, we have to compute terms of the form $\langle \log P(D_i|w_i, \Omega) \rangle_Q$ and $\langle \log P(w_i|\Omega) \rangle_Q$. We easily obtain, up to irrelevant constants,

$$\begin{aligned} -\left\langle \log P(D_i|w_i, \sigma, v) \right\rangle_Q &= \frac{1}{2\sigma^2} \sum_\mu \left\langle [y^\mu - w_i^T h(x^\mu(i))]^2 \right\rangle_Q + \frac{N}{2} \log \sigma^2 \\ -\left\langle \log P(w_i|m, \Sigma^2) \right\rangle_Q &= \frac{1}{2} \left\langle (w_i - m)^T \Sigma^{-2} (w_i - m) \right\rangle_Q + \frac{1}{2} \log \left| \Sigma^2 \right| . \quad (12) \end{aligned}$$

Both terms only depend on the first and second moments of $Q(w_i)$. Furthermore, optimization of each shared parameter can be done separately: σ after v and Σ after m. In the expectation step, we have to compute

$$P(w_i|D_i, \Omega^*) \propto P(D_i|w_i, \Omega^*) P(w_i|\Omega^*) \; .$$

The expressions for $P(D_i|w_i, \Omega^*)$ and $P(w_i|\Omega^*)$ are similar to those in (12), but then without the average over $Q(w_i)$. Combining both we get a simple Gaussian distribution $P(w_i|D_i, \Omega^*)$.

References

[1] B. Bakker, T. Heskes, A neural-Bayesian approach to survival analysis, in: Proceedings of ICANN99, 1999, pp. 832–837.

[2] D. Cox, D. Oakes, Analysis of Survival Data, Chapman Hall, London, 1984.

[3] M. Goldstein, A. Smith, Ridge-type estimators for regression analysis, Journal of the Royal Statistical Society 36 (1974) 284–291.

[4] A. Gelman, J. Carlin, H. Stern, D. Rubin, Bayesian data analysis, Chapman & Hall, London, 1995.

[5] R. Neal, Bayesian Learning for Neural Networks, Springer-Verlag, New York, 1996.

[6] G. Hinton, D. van Camp, Keeping neural networks simple by minimizing the description length of the weights, in: Proceedings of the 6th Annual Workshop on Computational Learning Theory, ACM Press, New York, 1993, pp. 5–13.

[7] D. Barber, C. Bishop, Ensemble learning for multi-layer networks, in: Advances in Neural Information Processing Systems 10, MIT Press, Cambridge, 1997, pp. 395–401.

[8] D. Barber, B. Schottky, Radial Basis Functions: a Bayesian treatment, in: Advances in Neural Information Processing Systems 10, MIT Press, Cambridge, 1997, pp. 402–408.

[9] D. MacKay, Probable networks and plausible predictions – a review of practical Bayesian methods for supervised neural networks, Network 6 (1995) 469–505.

[10] D. MacKay, Comparison of approximate methods for handling hyperparameters, Neural Computation 11 (1999) 1035–1068.

[11] H. Kappen, J. Neijt, Neural network analysis to predict treatment outcome, The Annals of Oncology 4 (1993) S31–S34.

[12] J. Berger, M. Delampady, Testing precise hypotheses, Statistical Science 2 (1987) 317–352.

[13] R. Caruana, Multitask learning, Machine Learning 28 (1997) 41–75.

[14] L. Pratt, B. Jennings, A survey of transfer between connectionist networks, Connection Science 8 (1996) 163–184.

[15] J. Baxter, A Bayesian/information theoretic model of learning to learn via multiple task sampling, Machine Learning 28 (1997) 7–39.

[16] A. Bryk, S. Raudenbusch, Hierarchical Linear Models, Sage, Newbury Park, 1992.

[17] C. Robert, The Bayesian Choice: A Decision-Theoretic Motivation, Springer, New York, 1994.

[18] R. Neal, G. Hinton, A view of the EM algorithm that justifies incremental, sparse, and other variants, in: M. Jordan (Ed.), Learning in Graphical Models, Kluwer Academic Publishers, Dordrecht, 1998, pp. 355–368.

[19] T. Heskes, Empirical Bayes for learning to learn, in: P. Langley (Ed.), Proceedings of the Seventeenth International Conference on Machine Learning, Morgan Kaufmann, San Francisco, CA, 2000, pp. 367–374.