To appear in Proceedings of the ICML-98, Madison, Wisconsin.

Solving a huge number of similar tasks: a combination of multi-task learning and a hierarchical Bayesian approach

Tom Heskes
Foundation for Neural Networks
Geert Grooteplein 21, 6525 EZ Nijmegen, The Netherlands
tom@mbfys.kun.nl

Abstract

In this paper, we propose a machine-learning solution to problems consisting of many similar prediction tasks. Each of the individual tasks has a high risk of overfitting. We combine two types of knowledge transfer between tasks to reduce this risk: multi-task learning and hierarchical Bayesian modeling. Multi-task learning is based on the assumption that there exist features typical to the task at hand. To find these features, we train a huge two-layered neural network. Each task has its own output, but shares the weights from the input to the hidden units with all other tasks. In this way a relatively large set of possible explanatory variables (the network inputs) is reduced to a smaller and easier to handle set of features (the hidden units). Given this set of features and after an appropriate scale transformation, we assume that the tasks are exchangeable. This assumption allows for a hierarchical Bayesian analysis in which the hyperparameters can be estimated from the data. Effectively, these hyperparameters act as regularizers and prevent overfitting. We describe how to make the system robust against nonstationarities in the time series and give directions for further improvement. We illustrate our ideas on a database regarding the prediction of newspaper sales.

1 INTRODUCTION

1.1 PROBLEM DESCRIPTION

In this paper, we focus on problems such as

- efficient distribution of newspapers and magazines;
- predicting gas consumption of different companies;
- analyzing sales figures of many company branches;
- optimizing stock selection and portfolio management.

The main characteristic of each of these problems is that they are in fact composed of many similar prediction tasks. These individual tasks usually have a low signal-to-noise ratio: in some cases one would be happy if one could explain 10 percent of the variance in the data. Because of the large amount of different tasks, any performance improvement is almost immediately significant, both financially and statistically. Furthermore, in most cases one can easily come up with quite a few (possibly) explanatory variables. For example, in predicting sales figures, one may want to include some of the recent sales figures, sales figures from the same period last year, sales figures from other companies, different kinds of weather information, and so on. Overfitting then becomes a major concern. The question addressed in this paper is therefore: how can we exploit the benefit of not having a single prediction task but a whole set of seemingly similar tasks, such that we can reduce the risk of overfitting in a computationally feasible way?

We propose to combine two approaches: multi-task learning, suggested in the neural-network and machine-learning community, and hierarchical Bayesian modeling, developed in the statistics community. Multi-task learning is treated in Section 2. The idea is that tasks can learn from each other by sharing the same features. The underlying assumption is that such features, typical to the task at hand, indeed exist. Hierarchical Bayesian modeling applies

when one can rely on the assumption that *a priori*, i.e., before taking into account the data itself, there is no information to distinguish the model parameters of any one task from those of any of the other tasks. We will describe hierarchical modeling in Section 3.

We will illustrate our ideas on a database concerning the prediction of newspaper sales. This database consists of several years of weekly sales figures for a set of 343 different points of sale. Each points of sale represents a different time-series prediction task. In Section 1.2 we first discuss how to make the tasks "sufficiently similar", i.e., such that we can apply the approach proposed in Section 2 and 3. Although our examples include collections of time-series tasks, our analysis in these sections in completely static. In Section 4.1 we therefore describe a first crude attempt to handle nonstationarities in the data. Section 4 further links the different components together, recapitulates the assumptions and discusses directions for further improvements.

1.2 MAKING TASKS SIMILAR

The underlying assumption of both the multi-task learning approach and the hierarchical Bayesian approach is that the different tasks can be considered similar. This is not always immediately obvious. As can be seen for example from Figure 1, where we plotted the averages sales of 343 newspaper points of sale versus their standard deviation, the typical number of single copies sold at each outlet ranges from just a few to a few hundred. Still we want to assume that the tasks are, in some sense, exchangeable. In Section 2 this implies that sales figures, when used as explanatory variables, should have more or less the same meaning: 20 newspapers may be quite a lot for a small outlet, but are well below average for a large outlet. Similar reasoning applies to the scaling of model parameters in our choice of prior distributions in Section 3. In the newspaper example, our working hypothesis will be that the points of sale are exchangeable, *after* correcting for their typical scale.

Such a correction can be accomplished by normalizing the sales figures for each outlet separately. The strong correlation between the average sales and the noise level in Figure 1 ($R^2 = 0.90$ on the logarithmic scale) suggests that we can represent the typical scale of each individual outlet through just one parameter θ_i, denoting the average sales of outlet i. We can correct for this typical scale by normalizing all sales figures using this average and the fitted standard deviation as given by the dashed line in Figure 1.

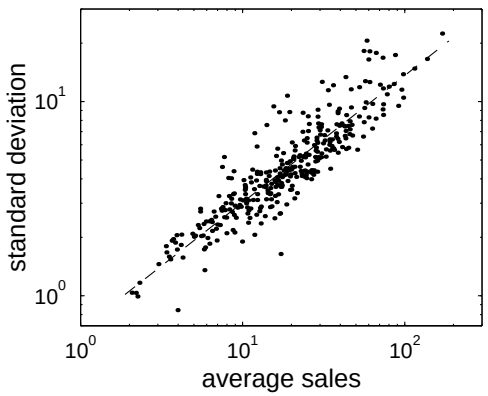

Figure 1: Average newspaper sales θ_i versus the corresponding standard deviation for 343 different points of sale. The dashed line is the least squares fit of the logarithm of the standard deviation as a function of the logarithm of the average sales.

2 MULTI-TASK LEARNING

2.1 ARGUMENTATION

We want to build and train a model relating a set of explanatory variables x to an output z. First we have to choose which explanatory variables to include in such a model. Typically, it is easy to come up with on the order of $n_{\text{inputs}} \approx 20$ input variables (see for example Table 1 where we describe the explanatory variables incorporated in our newspaper example). With on the order of a hundred training patterns per task and a low signal-to-noise ratio, any attempt to fit a direct model between the input variables and the targets corresponding to a single task, is doomed to lead to overfitting and thus lousy prediction performance.

We need some preprocessing stage transforming the n_{inputs} input variables x into a small set of say $n_{\text{features}} \approx 3$ features y, typical to the task at hand. In practice, one often tries to find these features through an iterative process of thinking and testing (see also Figure 4). For example, one tries several ways of combining the most recent sales figures into a single number, tests each of them, and takes the best. Here we propose to *learn* this transformation. We combine all tasks into one big network (see Figure 2). The input units are connected to the hidden (feature) units through a weight matrix B. The weight vector connecting the hidden units to the output unit corresponding to task i is denoted A_i. In other words, all

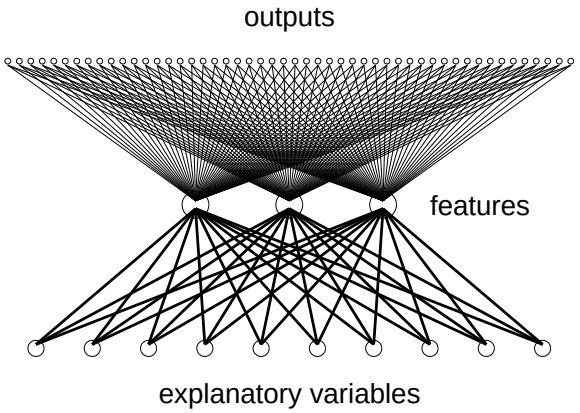

Figure 2: Typical network structure: a reasonably large number of input units, a small number of hidden units, and huge number of output units.

tasks share the weight matrix B, but have independent weight vectors A_i.

In this paper, we will consider the case of linear hidden units. Given an input vector x, the features and outputs are then computed through

$$y_j = \sum_k B_{jk} x_k \quad \text{and} \quad z_i = A_{i0} + \sum_j A_{ij} y_j \ , \qquad (1)$$

where z_i refers to the output corresponding to task i. We will use A_i to denote the set of all hidden-to-output weights specific to each task, i.e., $A_i \equiv \{A_{i0}, \ldots, A_{i,n_{\text{features}}}\}$. We refer to A_i and σ_i as the set of model parameters of task i.

The inputs x can be divided into two categories: those with equal input values across all tasks and those with input values specific to a particular task. Nonspecific inputs in the newspaper example (see Table 1) are e.g. seasonal variables and weather figures (we considered the "average" weather across The Netherlands instead of more local weather figures). The specific inputs should have more or less the same meaning across all tasks. This is accomplished by the transformation of the sales figures described and discussed in Section 1.2. We will use x_i to denote the set of inputs corresponding to task i.

Hidden units do not have bias units: it is easy to see that these can be scaled away into the bias of the output units. We further assume a Gaussian noise model with standard deviation σ_i, which is different for each task, but independent of the inputs x_i. Assuming that the targets $D_i \equiv \{t_i^\mu\}$ are independently and identically distributed (iid) given the inputs $I_i \equiv \{x_i^\mu\}$, model parameters A_i and σ_i and feature matrix B, we can compute the probability of observing these targets through

$$P(D_i|I_i, A_i, \sigma_i, B) \propto \exp\left[-E(A_i, \sigma_i, B|D_i, I_i)\right] \ , \quad (2)$$

where we have defined the error

$$E(A_i, \sigma_i, B|D_i, I_i) = \frac{1}{2} \sum_\mu \left[\frac{(t_i^\mu - z_i^\mu)^2}{\sigma_i^2} - \log \sigma_i \right] \ , \qquad (3)$$

with the output z_i^μ computed as in (1). For notational convenience we will from now on leave out the explicit dependency on the inputs I_i. The iid assumption may be too strong for time-series prediction tasks. We will come back to that in Section 4.1.

We propose to find an appropriate feature matrix B through a maximum likelihood procedure: we minimize the error (3), averaged over all n_{tasks} tasks and obtain the maximum likelihood solutions B^{ML}, A_i^{ML} and σ_i^{ML}.

2.2 SIMILAR IDEAS

There has been quite a lot of interesting research in the area of inductive transfer, yielding both empirical and theoretical evidence that multi-task learning improves performance (see [10] for collections of papers on multi-task learning). In [1] the advantage of combining several tasks is investigated theoretically, under the assumption that a feature matrix B common to all tasks indeed exists.

In most approaches to multi-task learning (see e.g. [2] and references therein), all tasks receive the same input information, i.e., all inputs are nonspecific. As in our case, the different tasks are forced to share the same hidden unit representation. Often, but not always, this leads to a better generalization performance [2]. The problems considered in the literature are mostly artificial and combine on the order of 10 or less tasks. An exception is [7], where different tasks concerning stock selection and portfolio management are combined in various ways. This experimental study is probably closest in spirit to our multi-tasking approach, but its number of tasks (36) is still much smaller than the 343 real-world tasks that we use in our simulation.

Group	#	Type	B_1	B_2	B_3
last year sales	3	specific	0.8	54.1	2.9
last year sellouts	3	specific	0.6	1.0	3.6
recent sales	5	specific	93.5	15.4	0.4
recent sellouts	5	specific	1.9	2.9	10.6
weather figures	5	nonspec.	1.2	15.8	15.0
season variables	2	nonspec.	1.9	10.7	67.6

Table 1: List of input variables (see text for further explanation) on the lefthand side. Numbers on the righthand side give the percentage of variance of the features explained by a particular group of input variables.

2.3 FEATURES FOR THE PREDICTION OF NEWSPAPER SALES

The explanatory variables that we took into account are summarized in Table 1. We normalized all non-specific variables. Sales figures were rescaled for each outlet separately as described in Section 1.2. Sellout figures were not rescaled: a sellout is represented by 1, a non-sellout by 0. Recent figures start from 4 weeks ago (the time it takes to collect and administrate all sales figures) and end at 8 weeks ago. Figures from last year are from exactly the same week and the week just before and after that. Weather information includes temperature (relative to the average temperature at the time of year), wind velocity, percentage sunshine, and precipitation (both amount and duration). We slightly changed the definition of the probability model (2) and error (3) to incorporate sellouts (number of sold copies equal to the number of delivered copies) and to take into account that newspaper sales is always integer.

We trained networks with $n_{\text{features}} = 1$ to 8 hidden units. The percentages in Table 1 indicate what part of the variance in each of the features is explained by a particular group of input variables for $n_{\text{features}} = 3$. The features are ordered from most to least relevant. The first feature strongly focuses on the recent sales, the second mostly on the sales from last year, the third mostly on the seasonal variables. Sellouts and weather figures seem to play a minor role, although especially the weather figures explain some of the variance of the second and third feature.

We can also compute the variance in the outputs explained by each group of input variables. The circles in Figure 3 show these percentages for different numbers of hidden units. With any number of hidden units, the recent sales figures come out to be most relevant. There are, however, interesting differences: a remarkable increase in the relevance of seasonal variables when going from one to two hidden units, a similar increase in the relevance of the recent sellouts when going from two to three hidden units, and somewhat less dramatic increases in last year's figures and weather information.

3 HIERARCHICAL BAYES

3.1 BAYESIAN MODELING

In this section, we replace the maximum likelihood approach of the previous section by a Bayesian approach. We will focus on a Bayesian inference of the model parameters A_{ij} and standard deviations σ_i, given the feature matrix B^{ML} obtained in the previous section. The underlying assumption is that, if there indeed exist features typical to the task of predicting newspaper sales, it should not matter too much whether we find these through an, in this context computationally unfeasible, Bayesian approach or through a much simpler maximum likelihood procedure. Furthermore, we are making lots of other assumptions: our choice of possible explanatory variables, the number of hidden units, the linear transfer function and thus restriction to find linear relationships, and so on. Each set of assumptions corresponds to a different model or hypothesis $\mathcal{H}$. We can simply include B^{ML} in our definition of $\mathcal{H}$. In the following, all probability distributions are conditioned on this $\mathcal{H}$. We will omit this explicit dependency from our notation.

Equation (2) gives the probability distribution of the data for a single task given its model parameters. The probability distribution of all data follows from

$$P(\mathcal{D}|\mathcal{A}) = \prod_i P(D_i|A_i) \; ,$$

where A_i now stands for all model parameters of task i (including the standard deviation σ_i), $\mathcal{A} \equiv \{A_1, \ldots, A_{n_{\text{tasks}}}\}$, and $\mathcal{D} \equiv \{D_1, \ldots, D_{n_{\text{tasks}}}\}$. In a Bayesian analysis, we infer the probability of the model parameters given the data using Bayes' rule:

$$P(\mathcal{A}|\mathcal{D}) = \frac{P(\mathcal{D}|\mathcal{A})P(\mathcal{A})}{P(\mathcal{D})} \; , \qquad (4)$$

where $P(\mathcal{D})$ is a normalization factor independent of the model parameters and $P(\mathcal{A})$ is a prior distribution of the model parameters.

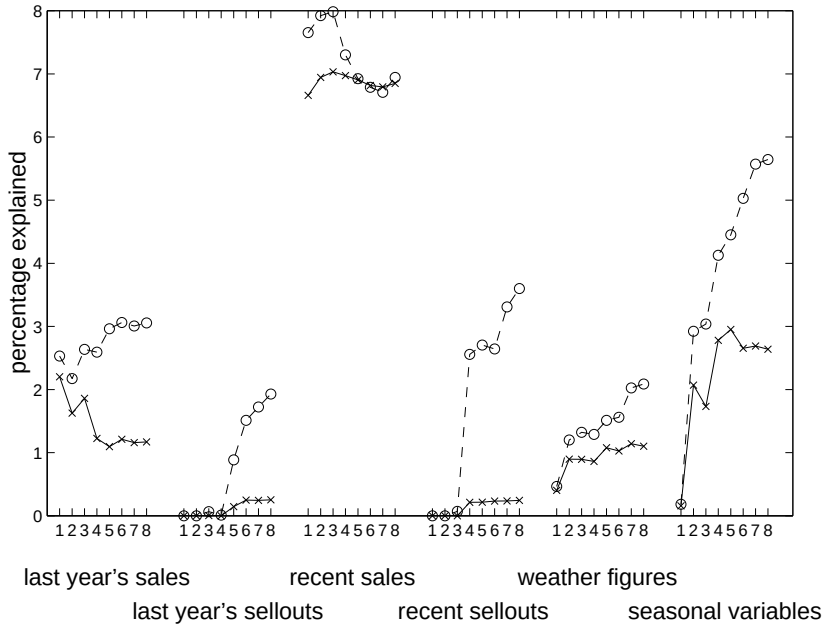

Figure 3: Percentage of variance explained by each group of input variables for various numbers of hidden units: maximum likelihood solution (circles, dashed lines) and most probable solutions (crosses, solid lines).

We take a Gaussian prior on the model parameters $A_i \equiv \{A_{i0}, \ldots, A_{i,n_{\text{features}}+1}\}$, with $A_{i,n_{\text{features}}+1} \equiv \log \sigma_i$:

$$P(A_i|\Lambda) \propto \exp\left[-\frac{1}{2}(A_i - m)^T \lambda (A_i - m)\right] ,$$

where $\Lambda = \{\lambda, m\}$ is called a set of hyperparameters with λ an $[(n_{\text{features}} + 2) \times (n_{\text{features}} + 2)]$-dimensional symmetric matrix and m an $(n_{\text{features}}+2)$-dimensional vector. The model parameters of each task are assumed to be exchangeable, i.e.,

$$P(\mathcal{A}|\Lambda) = \prod_i P(A_i|\Lambda) .$$

This exchangeability assumption can be compared with the iid assumption in (2). It implies that, prior to the arrival of data, the probability distribution of the model parameters is invariant under renumbering of the tasks. This is not directly obvious, but may be a reasonable assumption if the outputs for each of the tasks are appropriately rescaled, as discussed in Section 1.2. Another interpretation is that the parameters of the different tasks are penalized by the same set of hyperparameters.

In an exact Bayesian procedure, one should always integrate out the hyperparameters. In a hierarchical Bayesian procedure, we approximate (4) through

$$P(\mathcal{A}|\mathcal{D}) = \int d\Lambda P(\mathcal{A}|\Lambda, \mathcal{D}) P(\Lambda|\mathcal{D}) \approx P(\mathcal{A}|\Lambda^{\text{MP}}, \mathcal{D}),$$
$$\text{with } \Lambda^{\text{MP}} = \operatorname*{argmax}_{\Lambda} P(\Lambda|\mathcal{D}) .$$

The procedure is called hierarchical to indicate that the hyperparameters are inferred at a higher level than the model parameters. The idea behind this approximation is that the distribution $P(\Lambda|\mathcal{D})$ is sharply peaked around its most probable value Λ^{MP}. In our case, where we can use the data for all n_{tasks} tasks to infer the most probable Λ^{MP}, this approximation is extremely accurate and useful. We will simply take an (improper) flat prior for Λ, i.e., $P(\Lambda) \propto 1$, such that the most probable Λ^{MP} is in fact equivalent to the set of maximum likelihood hyperparameters Λ^{ML}.

3.2 RELEVANT LITERATURE

A nice overview of hierarchical (also called empirical) Bayesian modeling, with both a discussion of its underlying assumptions and lots of references to its applications in statistics, can be found in [6]. Our approach is quite similar in spirit to the use of empirical Bayesian techniques in law school validity studies, described and

discussed in [9]. James-Stein estimation can be viewed as the frequentists' equivalent of hierarchical Bayesian modeling. A nice link is provided in [4].

In the neural-network community, hierarchical Bayes is often referred to as the evidence framework [8]. The focus is on learning a single task, where the prior distribution of the weights (usually a diagonal matrix λ and m equal to zero) is chosen to reflect the belief that weights should be small. This yields the Bayesian justification for weight decay or ridge regression. Although from a technical point of view our analysis is at some points quite similar, the meaning of the prior distribution is different: our choice of priors has nothing to do with an *a priori* assumption of small weights, only with exchangeability under a Gaussian probability model.

3.3 INFERENCE OF THE HYPERPARAMETERS

To find the most probable set of hyperparameters $\Lambda^{\rm MP}$, we have to maximize the posterior distribution $P(\Lambda|\mathcal{D})$. One way of doing this is through an EM algorithm (see e.g. [6]). The multi-task situation allows for quite a lot of simplifications, which in the end lead to update equations for $\Lambda(n)$. Here we only state the result:

$$m(n+1) = \frac{1}{n_{\rm tasks}} \sum_i \bar{A}_i(n)$$

$$\lambda^{-1}(n+1) = \frac{1}{n_{\rm tasks}} \sum_i \Sigma_i^2(n) + \frac{1}{n_{\rm tasks}} \sum_i \left[\bar{A}_i(n) - m(n+1)\right] \left[\bar{A}_i(n) - m(n+1)\right]^T$$

where $\bar{A}_i(n)$ and $\Sigma_i^2(n)$ are the mean and variance of the distribution $P(A|D_i, \Lambda(n))$, respectively. The second term on the righthand side measures the variance between the most probable solutions [given $\Lambda(n)$] for the different tasks, the first term the variance of $P(A|D_i, \Lambda(n))$ around these most probable solutions, averaged over all tasks. We can use Laplace's method (see [6]), based on a quadratic Taylor expansion of $\log P(A|D_i, \Lambda(n))$ around its mode, to find approximations for $\bar{A}_i(n)$ and $\Sigma_i^2(n)$:

$$\bar{A}_i(n) \approx \operatorname*{argmax}_A \log P(A|D_i, \Lambda(n)) \quad \text{and} \quad \Sigma_i^2(n) \approx [H_i(n) + \lambda_n]^{-1} ,$$

where the Hessian matrix $H_i(n)$ of the error $E(A|D_i)$ has to be evaluated at $\bar{A}_i(n)$:

$$H_i(n) \equiv \left. \frac{\partial^2 E(A|D_i)}{\partial A \partial A^T} \right|_{A=\bar{A}_i(n)} .$$

Laplace's method becomes more and more accurate for large sample sizes p per task.

The EM algorithm is intuitive and computationally feasible with the approximation suggested by Laplace's method. A disadvantage of the EM algorithm is that its convergence can be rather slow. A more direct method can be obtained if we make a stronger assumption, namely that the error $E(A|D_i)$ is approximately quadratic in the model parameters A, i.e.,

$$E(A|D_i) \approx E(A_i^{\rm ML}|D_i) + \frac{1}{2}(A - A_i^{\rm ML})^T H_i (A - A_i^{\rm ML}) , \tag{5}$$

with $A_i^{\rm ML}$ the maximum likelihood solution minimizing $E(A|D_i)$ and H_i the Hessian evaluated at $A_i^{\rm ML}$. This is the approximation frequently applied in the evidence framework for neural networks (see e.g. [8]). Now all integrations needed to compute

$$P(\Lambda|\mathcal{D}) \propto \prod_i \int dA \, P(D_i|A) P(A|\Lambda) ,$$

are over Gaussian probability distributions, yielding

$$\log P(\Lambda|\mathcal{D}) = -\frac{1}{2} \sum_i (A_i^{\rm ML} - m)^T Z_i(\lambda) (A_i^{\rm ML} - m) + \frac{1}{2} \sum_i \log \left[\det Z_i(\lambda)\right] , \tag{6}$$

with $Z_i(\lambda) \equiv \left(H_i^{-1} + \lambda^{-1}\right)^{-1}$ and where we neglected irrelevant additive constants. The most probable $\Lambda^{\rm MP}$ maximizes (6) and can be found using e.g. a standard BFGS quasi-Newton algorithm.

3.4 SIMULATIONS

In our newspaper example, the approximation (5) appeared to be extremely accurate. $\Lambda^{\rm MP}$ was therefore obtained through direct optimization of (6). Given this $\Lambda^{\rm MP}$, we computed the most probable model parameters $A^{\rm MP}$ exactly, i.e., without making the approximation (5). The difference between the calculation of the maximum likelihood solutions and the most probable solutions is that the latter are regularized through the hyperparameters $\Lambda^{\rm MP}$.

In the previous section we noted dramatic changes in the relevance of groups of input variables with increasing number of hidden units. The relevances for

the most probable solutions, shown by the circles in Figure 3, are surprisingly constant across the different networks with $n_{\text{features}} > 1$: given the correct prior parameters Λ^{MP}, the most probable solutions are roughly the same. Especially the influence of the sell-outs, which seemed to be highly relevant according to the maximum likelihood solutions, almost completely vanishes.

4 DISCUSSION AND CONCLUSION

4.1 DEALING WITH NONSTATIONARITY

Until now, our analysis has been completely static. However, the typical examples given in Section 1 are mostly time-series prediction problems, for which the iid assumption (2) can be too strong. Suppose that we want to predict the output z at "time" μ given inputs x^μ (we leave out the index i for notational convenience). As in the previous sections, we fit the parameter set $\mathbf{A} \equiv \{\theta, A, \sigma\}$ on a training set containing the most recent p patterns. This is a kind of "sliding window approach": with the addition of every new pattern, the oldest pattern is deleted from the training set. With a delay of n_{delay} patterns between the most recently available pattern and the output to be predicted, the training set ends at $\mu - n_{\text{delay}}$. The naive sliding window approach now computes the output from the input x^μ and the scale and model parameters $\mathbf{A}^{\mu - n_{\text{delay}}}$, which in a way assumes stationarity of the scale and model parameters, i.e., $\mathbf{A}^\mu \approx \mathbf{A}^{\mu - n_{\text{delay}}}$. This naive approach may work fine for many prediction tasks, but leads to lousy predictions on some of them.

To take nonstationarity into account, we add a correction term to the uncorrected prediction:

$$z^{\mu+n_{\text{delay}}}_{\text{corrected}} = z^{\mu+n_{\text{delay}}}_{\text{uncorrected}} + \Delta^\mu \ .$$

The parameter set $\mathbf{A}$ used to compute the uncorrected $z^\mu_{\text{uncorrected}}$ is determined as before and we still make the assumption that this parameter set is roughly stationary on a time scale of a few patterns. Any nonstationarity should be corrected through Δ^μ. A simple, but efficient procedure for updating Δ^μ is through an exponential smoothing procedure:

$$\bar{\Delta}^\mu = \alpha e^\mu_{\text{uncorrected}} + (1-\alpha)\bar{\Delta}^{\mu-1} = \alpha e^\mu_{\text{corrected}} + \bar{\Delta}^{\mu-1},$$

with $e^\mu_{\text{uncorrected}} \equiv t^\mu - z^\mu_{\text{uncorrected}}$ and $e^\mu_{\text{corrected}} \equiv t^\mu - z^\mu_{\text{corrected}}$ the difference between the target and the uncorrected and corrected prediction, respectively. α is a so-called smoothing parameter and $1/\alpha$ corresponds to a typical time scale. It seems reasonable to choose the same α for all tasks. Furthermore, it is well-known (see e.g. [3]) that the precise setting of the smoothing parameter in exponential smoothing hardly affects the prediction performance (see also Figure 4). Perfectly stationary tasks hardly suffer from the extra correction, since their errors $e^\mu_{\text{uncorrected}}$ and $e^\mu_{\text{corrected}}$ tend to average out anyways.

4.2 TEST PERFORMANCE

Some results are displayed in Figure 4. All ideas presented in this paper have been implemented and tested on the prediction of newspaper sales for 343 points of sale. The test set consists of 85 weeks after the training set that has been used for computation of the feature matrix, hyperparameters and most probable model parameters. The model parameters are updated weekly using the sliding windows approach described above. The hyperparameters and feature matrix have been kept constant. The test error is minus the loglikelihood, averaged over both patterns and points of sale.

The network with two hidden units appears to be the best. The regularization through the Bayesian approach cannot completely avoid the risk of overfitting. On the other hand, the best solution without regularization (not shown) is the one with one hidden unit, with a test error of about 2.7, increasing rapidly for more hidden units. The star shows the test performance for a fixed choice of the feature matrix B, made before the start of this project after quite a lot of iterations of thinking, trying and testing. The solution obtained through the multi-tasking approach is significantly better. The righthand side shows the sensitivity to the choice of the smoothing parameter. Taking $\alpha = 0$ is suboptimal: at least for some points of sale, the time series are clearly nonstationary. Any choice of a typical smoothing time between half a year and a year leads to about the same performance.

4.3 STATIONARITY AND SPECIFICITY

A summary of the most important parameters in the complete system is given in Table 2. At the highest level, we have global parameters as the number of features n_{features} and the smoothing parameter α. The choice of these scalar parameters is not extremely critical (see Figure 4) and can be based either on experience with similar databases or by testing a few different alternatives. This is much less the case for the next

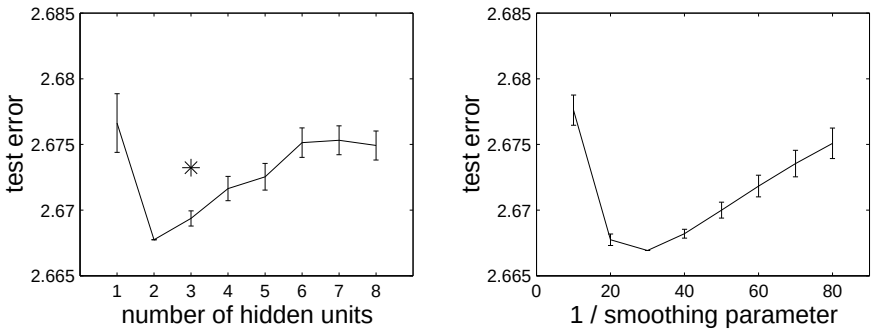

Figure 4: Test error (minus loglikelihood) averaged over 85 weeks and 343 points of sale. Error bars indicate the significance of the difference with the best solution. Lefthand side: as a function of the number of hidden units for smoothing parameter $\alpha = 0.05$. The star corresponds to the performance with a choice of three features obtained after extensive trial and error. Righthand side: as a function of the typical smoothing time (number of weeks) for the network with two hidden units.

level of parameters: the input-to-hidden weights and the hyperparameters of the prior distribution. These parameters are typical to the task at hand. For example, in predicting newspaper sales, they may be quite different for different days of the week. They can be optimized on a representative set of tasks and kept fixed afterwards. The model and scale parameters as well as the correction terms are obviously specific to each task. We assume that the model parameters are roughly stationary over the length of the training set and can thus be determined through a sliding window approach. The correction terms can be interpreted as corrections to the scale parameters. These may be much less stationary and should be updated with the addition of every single pattern.

4.4 IMPROVEMENTS AND FURTHER DIRECTIONS

Let us recapitulate our approach and underlying assumptions. We started with the observation that we needed some transformation from the possibly quite high-dimensional input space to a much lower dimensional feature space. We proposed to learn this transformation through a maximum likelihood procedure on the weights of a huge network containing all tasks. In this we did not incorporate any prior information, nor did we worry about nonstationarity of the time series involved. Keeping the weights from input to hidden units fixed, we then performed a hierarchical Bayesian analysis to compute hyperparameters, which, roughly speaking, gave us the proper regularization of the model parameters specific to each task. Again, we disregarded any of the nonstationarity in the data. Finally, keeping both the hyperparameters and input-to-hidden weights fixed, we proposed an exponential smoothing procedure to correct for nonstationarities.

We might try and think of ways to integrate the parts of our approach, instead of applying them sequentially. For example, it may be possible to treat the input to hidden weights as hyperparameters, i.e., at the same level as the hyperparameters Λ for the mean and variance of our prior distribution. The problem here is that it is much more difficult to compute how a change in the hyperparameters of the prior distribution affects the input-to-hidden weights than vice versa. Our treatment follows from the assumption that this effect is negligible for practical purposes. It is not easy to see how to go beyond this simplification, without having to rely on procedures that are computationally unfeasible for any reasonable number of tasks.

About integrating nonstationarity and the Bayesian hierarchical analysis, we may be somewhat more positive. There has been some recent work, which can be viewed as a first attempt to combine Kalman filtering and the Bayesian evidence framework [5]. In this approach the hyperparameters are recomputed every time step. Similar ideas may be applicable to our multi-task situation, although also here we have to worry about the computational feasibility.

Another improvement could be to work with a more complicated prior for the model parameters of the different tasks than the Gaussian considered in this paper. One suggestion is to take another functional form,

Symbol	Description	Time	Tasks	Procedure
n_{features}	number of hidden units	constant	same	experience/test performance
α	smoothing parameter	constant	same	experience/test performance
B	input-to-hidden weights	constant	same	multi-task learning
Λ	hyperparameters	constant	same	Bayesian inference
θ	scale parameters	sliding window	specific	maximum likelihood
A	model parameters	sliding window	specific	MAP estimation
Δ	correction terms	single pattern	specific	exponential smoothing

Table 2: Characteristics of the most important parameters.

for example, a cluster of Gaussians or a prior which forces each task to focus on a subset of the available features. An even more appealing approach would be to make the prior distribution dependent on (known) characteristics of the particular task. In our newspaper case, the width and mean of the distribution could be functions of the distance from the point of sale to the beach, the population density in the vicinity of the point of sale, and so on. The hyperparameters to be inferred from the data would be the parameters in this functional dependency.

Acknowledgement

This research was supported by the Technology Foundation STW, applied science division of NWO and the technology programme of the Ministry of Economic Affairs.

References

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

[2] R. Caruana. Multitask learning. *Machine Learning*, 28:41–75, 1997.

[3] C. Chatfield. *The Analysis of Time Series: an Introduction*. Chapman & Hall, London, fourth edition, 1989.

[4] B. Efron and C. Morris. Data analysis using Stein's estimator and its generalizations. *Journal of the American Statistical Association*, 70:311–319, 1975.

[5] J. de Freitas, M. Niranjan, and A. Gee. Regularisation in sequential learning algorithms. In *Advances in Neural Information Processing Systems 10*, Cambridge, 1998. MIT Press.

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

[7] J. Ghosn and Y. Bengio. Multi-task learning for stock selection. In M. Mozer, M. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9*, pages 946–952, Cambridge, 1997. MIT Press.

[8] D. MacKay. Bayesian interpolation. *Neural Computation*, 4:415–447, 1992.

[9] D. Rubin. Using empirical Bayesian techniques in the law school validity studies (with discussion). *Journal of the American Statistical Association*, 75:801–827, 1980.

[10] S. Thrun and L. Pratt, editors. *Machine Learning. Second Special Issue on Inductive Transfer*, Dordrecht, 1997. Kluwer Academic.