Empirical Bayes for Learning to Learn

Tom Heskes TOM@MBFYS.KUN.NL
SNN, University of Nijmegen, Geert Grooteplein 21, Nijmegen, 6525 EZ, The Netherlands

Abstract

We present a new model for studying multitask learning, linking theoretical results to practical simulations. In our model all tasks are combined in a single feedforward neural network. Learning is implemented in a Bayesian fashion. In this Bayesian framework the hidden-to-output weights, being specific to each task, play the role of model parameters. The input-to-hidden weights, which are shared between all tasks, are treated as hyperparameters. Other hyperparameters describe error variance and correlations and priors for the model parameters. An important feature of our model is that the probability of these hyperparameters given the data can be computed explicitely and only depends on a set of sufficient statistics. None of these statistics scales with the number of tasks or patterns, which makes empirical Bayes for multitask learning a relatively straightforward optimization problem. Simulations on real-world data sets on single-copy newspaper and magazine sales illustrate properties of multitask learning. Most notably we derive experimental curves for "learning to learn" that can be linked to theoretical results obtained elsewhere.

1. Introduction

Machine learning is the art of building machines that learn from data. Whereas the learning part is automatic, an expert has to take care of the building part. This often involves choosing the structure and learning algorithm, setting regularization parameters, and so on. In many cases the expert's capabilities are crucial for the success or failure of an application: he or she has to provide the right bias that makes the machine generalize well. Why not try to move the expert's responsibility to the next level and let the machine itself *learn* its own bias? This is the concept of "learning to learn" described in, among others, Baxter (1997) and Thrun and Pratt (1997).

Multitask learning constitutes an ideal setting for studying learning to learn. 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 neural architecture 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. Caruana (1997) and Pratt and Jennings (1996). Whether it works depends on whether the tasks are indeed sufficiently similar, which is often hard to tell in advance.

Baxter (1997) proposed hierarchical Bayesian inference as a model for studying multitask learning. Parameters that are shared between tasks are treated as hyperparameters at a higher level than the task-specific model parameters. Information-theoretic arguments show that multitask learning can be highly advantageous, especially when the number of hyperparameters is much larger than the number of model parameters per task. The necessary assumption to arrive at these theoretical results is that the tasks are indeed related, i.e., that they are drawn from the same (hyper)distribution.

In this article, we aim at a practical implementation of Baxter's framework. The essence is the ability to compute the probability of the hyperparameters given the data. In a full hierarchical Bayesian procedure, one should sample the hyperparameters from this distribution, and subsequently the model parameters from the distribution of model parameters given the data *and* the hyperparameters. The empirical Bayesian approach is the frequentist shortcut: rather than sampling the distribution of hyperparameters, we will only consider their most likely values.

We will study a specific model for solving many related regression tasks, which is similar to the one studied in Heskes (1998), except for all kinds of technical details. These details are necessary for building a robust and accurate prediction system, but a deathblow for clear methodology and extensive testing. In this article we therefore abstract from the exact implementation and describe a simpler model, easier to analyze theoretically but still applicable on real-world problems.

In Section 2 we work out our model of multitask learning. The empirical Bayesian approach is sketched in Section 3, with the crucial probability of the hyperparameters given the data outlined in the Appendix. The model is illustrated on real-world data involving newspaper and magazine sales in Section 4. Ideas for further research are discussed in Section 5.

2. A Model of Multitask Learning

2.1 Data Likelihood

We are given a set of N combinations of input vectors x^μ and n outputs y_i^μ, each corresponding to a different task. Our model assumption is that the output y_i^μ is a linear function of the input x^μ with additive noise ϵ_i^μ and δ^μ:

$$y_i^\mu = A_i^T B x^\mu + \epsilon_i^\mu + \delta^\mu \; . \tag{1}$$

We will assume that there is a single output for each task and write y^μ for the vector of all n outputs y_i^μ. The input vectors x^μ are the same for all tasks. This assumption is introduced to simplify the mathematical exposition, but will be relaxed later on.

The noise part $\epsilon_i^\mu + \delta^\mu$ consists of two Gaussian noise terms with average zero and standard deviation σ and $\rho\sigma$, respectively. The noise ϵ_i^μ is specific to task i, the noise δ^μ the same for all tasks. This distinction between an individual noise term and a common noise term becomes relevant when the individual predictions are translated to an aggregate level. For example, fluctuations in the error averaged over all tasks will be constant and proportional to $\rho\sigma$ for nonzero ρ, yet scale with $1/\sqrt{n}$ for $\rho = 0$. Furthermore, substantial correlations might indicate that some important information is lacking. Both σ and ρ are the same for all tasks and thus treated as hyperparameters.

The model part $A_i^T B x^\mu$ can be interpreted as a linear neural network. The $n_{\rm hid} \times n_{\rm inp}$ matrix B contains the weights connecting the inputs to the hidden units, the $n_{\rm hid} \times n$ matrix A, with column vectors A_i, represents the weights from the hidden to the outputs units. Typically we have $n_{\rm hid} < n_{\rm inp} \ll n$, i.e., a bottleneck of hidden units. The hidden units will tend to find a low dimensional representation of the inputs that is useful for all tasks. The standard case, maximum likelihood for $\rho = 0$, has been treated in great detail in Baldi and Hornik (1989). The feature matrix B is shared by all tasks and will thus play the role of a hyperparameter. The vectors A_i, on the other hand, are task-specific and thus treated as model parameters.

Let D stand for the outputs y_i^μ for all tasks i and patterns μ. Assuming independently and identically distributed (iid) observations we have the likelihood

$$P(D|A, B, \rho, \sigma) = \prod_\mu \phi(y^\mu : A^T B x^\mu, \Sigma_\epsilon) \tag{2}$$

where $\phi(y : m, \Sigma)$ means that y is normally distributed with mean m and covariance matrix Σ. Σ_ϵ is the noise covariance matrix with components $[\Sigma_\epsilon]_{ii} = (1+\rho^2)\sigma^2$ and $[\Sigma_\epsilon]_{ij} = \rho^2\sigma^2$.

The linearity of (1) induces a symmetry: the data likelihood (2) cannot distinguish between $\{QA, Q^{-1}B\}$ and the original $\{A, B\}$ for any invertible $n_{\rm hid} \times n_{\rm hid}$ matrix Q. We use this freedom to constrain B such that the covariance matrix of hidden unit activities is the identity matrix:

$$\frac{1}{N} \sum_\mu [Bx^\mu][Bx^\mu]^T = B\Sigma_{xx}B^T = \mathbf{I} \tag{3}$$

with $\Sigma_{xx} = \left\langle xx^T \right\rangle_{\rm patterns}$ the covariance matrix of the inputs. This constraint greatly simplifies the analysis that follows. Going through this analysis it can be seen that we do not need to require that all tasks receive the same input vectors x^μ, but just that they have the same number of input vectors and the same input covariance matrix (and thus the same covariance matrix of hidden unit activities).

With the constraint (3) the maximum likelihood solution $A_{\rm ml}$ takes the simple form

$$A_{\rm ml} = {\rm argmax}_A P(D|A, B, \rho, \sigma) = B\Sigma_{xy} \; ,$$

with $\Sigma_{xy} = \left\langle xy^T \right\rangle_{\rm patterns}$ the input-output covariance matrix. Note that the data likelihood (2) can be rewritten as the exponent of a term quadratic in $A - A_{\rm ml}$ times a term independent of A.

2.2 Prior Information

The model parameters A_i determine the impact of the hidden units on output i. As a start we could consider the Gaussian prior

$$P(A_i|M, \Sigma_A) = \phi(A_i : M, \Sigma_A) \; ,$$

with M a vector of length $n_{\rm hid}$ and Σ_A an $n_{\rm hid} \times n_{\rm hid}$ covariance matrix. This corresponds to a so-called exchangeability assumption (the same M for all tasks) and introduces a tendency for similar model parameters across tasks. How similar is determined by the covariance matrix Σ_A.

The exchangeability assumption is rather strong. It can be generalized to include task-dependent averages, e.g., by making these (linearly) dependent on an $n_{\text{prior}} \times n$ matrix z with task characteristics:

$$P(A_i|M_i, \Sigma_A) = \phi(A_i : M_i, \Sigma_A) \quad \text{with} \quad M = \beta z \,.$$

β is an $n_{\text{hid}} \times n_{\text{prior}}$ matrix with regression parameters. The exchangeability prior is a special case: $n_{\text{prior}} = 1$ and z a vector of all ones. In the case of newspaper sales, task characteristics are properties of the particular outlet that are assumed to have an effect on sales patterns. An example is (a useful representation of) the outlet's geographical location.

3. Empirical Bayes

3.1 Integrating out the Model Parameters

Let $\Lambda = \{B, \rho, \sigma, \beta, \Sigma_A\}$ denote the set of all hyperparameters. In empirical Bayes the idea is to try and find the hyperparameters that maximize the probability $P(\Lambda|D)$, which are then fixed when computing statistics over the model parameters (see e.g. Robert, 1994). Using Bayes' formula we obtain

$$\begin{aligned} P(\Lambda|D) &= \int dA P(A, \Lambda|D) = \int dA \frac{P(A, \Lambda, D)}{P(D)} \\ &= \frac{P(\Lambda)}{P(D)} \int dA P(D|A, B, \rho, \sigma) P(A|\beta, \Sigma_A)\,. \end{aligned}$$

Taking a flat prior for $P(\Lambda)$, the goal is to minimize the loss defined as

$$L(\Lambda) = -\log \int dA P(D|A, B, \rho, \sigma) P(A|\beta, \Sigma_A).$$

This integral over the model parameters is highly dimensional, but perfectly doable since all terms in the exponent are at most quadratic in A. The result is given in the Appendix.

3.2 Properties of the Loss Function

Here we highlight the main features of the loss function $L(\Lambda)$. First of all we note that $L(\Lambda)$ scales with Nn, i.e., with the total number of examples in the data set. For the conditions considered in this article (both N and n a few hundred) and except for symmetries, the probability $P(\Lambda|D)$ is sharply peaked around its optimal value. This validates the empirical Bayesian approach, which is asymptotically equivalent to a full hierarchical Bayesian approach that would also require sampling over hyperparameters (see e.g. Robert, 1994).

The loss $L(\Lambda)$ only depends on the data through a set of "sufficient statistics". An example of such a statistic is the $n_{\text{inp}} \times n_{\text{inp}}$ matrix $R_{xx} = \Sigma_{xy}\Sigma_{xy}^T/n$. The other statistics are given in the Appendix. This property of the loss function has important practical and theoretical consequences. In practice, the procedure for obtaining the most likely hyperparameters no longer scales with the number of tasks or patterns (after initial computation of the statistics). This speed-up enables extensive testing of all kinds of ideas and options. In the rest of this article we will exploit these practical possibilities and illustrate properties of multitask learning on real-world databases, leaving precise theoretical analyses for later.

3.3 Link with Multilevel Analysis

The empirical Bayesian procedure proposed here can be viewed as a special case of a hierarchical Bayesian approach to multilevel analysis, a statistical method for analyzing nested data sets (see e.g. Bryk and Raudenbusch, 1992). Hierarchical because of the different stages of inference, multilevel because of the different levels of noise. Important differences with standard multilevel analysis are the inference of the feature matrix B and the incorporation of correlated errors.

4. Illustrations on Real-World Data

4.1 Preprocessing the Data Sets

For illustration we use three different data sets. Two of them (data sets I and II), involve newspaper sales, the other magazine sales (III). Each of the sets contains two to three years of weekly sales figures, i.e., N between 100 and 150, for $n = 343$ (data set I) to 1000 (data set III) outlets. Explanatory variables taken into account differ between data sets, with the number of inputs ranging from $n_{\text{inp}} = 9$ to 20, but always contain a few weeks of recent sales figures (specific to the outlet involved), two inputs coding season, and one constant input. The sales figures to be estimated are all rescaled per outlet to have mean zero and unit standard deviation. Outliers are detected and removed, the sales figures are corrected for sellouts, and missing input values are filled in by averaging over the inputs of nearby editions. The original input vectors are transformed such that the input covariance matrices built from the training inputs of each task are indeed the same. We neglect the minor differences in the number of training patterns for each task and take for N the number of training patterns averaged over all tasks.

4.2 Simulation Paradigm and Error Measures

The simulation paradigm for multitask learning is somewhat more involved than the one for a single task, since now we can sample both over tasks and over patterns. As performance measure we take the mean-squared error, denoted

$$E(D_{\text{tested}}|D_{\text{model}}, D_{\text{hyper}}) = \left\langle (y - A_{\text{mp}}^T B_{\text{ml}} x)^2 \right\rangle \ .$$

Here the average is over all combinations of inputs and outputs belonging to data set D_{tested}. The hyperparameters $\Lambda_{\text{ml}} = \text{argmax}_\Lambda P(D_{\text{hyper}}|\Lambda)$, including the feature matrix B_{ml}, are used to derive the most probable solutions $A_{\text{mp}} = \text{argmax}_A P(A|D_{\text{model}}, \Lambda_{\text{ml}})$.

Let us consider the general layout for a set of simulations in which we would like to evaluate the effect of varying one parameter, for example the number of hidden units, on a particular data set. Before each run, which involves testing all alternatives, we randomly split up the total set of tasks in two equal parts, D and its complement $\bar{D}$. The data sets are further randomly subdivided into 80% training set, denoted D_{tr}, and 20% test set, denoted D_{te}. Note that for practical purposes we are here considering in-sample test error rather than out-of-sample test error as in Heskes (1998). Following the above notation, we can now distinguish four different types of errors:

	intra	inter
training	$E(D_{\text{tr}}\|D_{\text{tr}}, D_{\text{tr}})$	$E(D_{\text{tr}}\|D_{\text{tr}}, \bar{D}_{\text{tr}})$
test	$E(D_{\text{te}}\|D_{\text{tr}}, D_{\text{tr}})$	$E(D_{\text{te}}\|D_{\text{tr}}, \bar{D}_{\text{tr}})$

Training and test refer to patterns, intra and inter to tasks. From the perspective of "learning to learn", the inter-task test error is the most interesting one: it measures to what extent the hyperparameters obtained on one set of tasks generalize to another set. By interchanging the role of D and $\bar{D}$, we compute these four errors twice for each run and each alternative, and average over the two options. For example, we consider in fact the statistics of

$$E_{\text{test}}^{\text{inter}} = \frac{1}{2}\left[E(D_{\text{te}}|D_{\text{tr}}, \bar{D}_{\text{tr}}) + E(\bar{D}_{\text{te}}|\bar{D}_{\text{tr}}, D_{\text{tr}})\right] \ ,$$

and similarly for the other types of errors. This strategy reduces the fluctuations induced by the accidental subdivision of tasks. A similar strategy for the subdivision in training and test patterns would leave us too few training patterns.

4.3 Qualitative Findings

Being a nonlinear function of the hyperparameters, $L(\Lambda)$ might well have local minima. However, with proper parameterizations (e.g., square root of the prior covariance Σ_A) and initial conditions (random, small β, large $\sqrt{\Sigma_A}$), a standard conjugate gradient algorithm happens to be extremely robust and leads to reproducible results. In the first stages of learning the feature matrix B changes a lot, with the other parameters slowly tracking. The second stages of learning, usually after roughly 100 conjugate gradient iterations, show a joint fine-tuning of all parameters involved. Apparently the loss function is dominated by the dependency of the standard mean-squared error (the term $L_{11}(B)$ in the Appendix) on the feature matrix B. As shown in Baldi and Hornik (1989), this term has no local minima, only saddle points.

The obtained feature matrices B make a lot of sense. One of the dominant features for all data sets is a weighted summation over recent sales figures. The other features are more specific to the characteristics of the different data sets, but always reasonable and reproducible. Features obtained for a smaller network reappear with only minor changes in a network with more hidden units, which is consistent with the principal component analysis of Baldi and Hornik (1989).

The noise parameters σ and ρ are on the order of 0.9 and 0.3, respectively. This implies that somewhere between 10 and 20% of the variance can be explained. Because of the large number of tasks and the daily need for accurate predictions, performance improvements of much less than 1% can be highly significant, both statistically and financially. The error correlations are substantial, which is not unexpected since there are many factors involving newspaper and magazine sales that affect all outlets but cannot (yet) be taken into account. An example is news content such as sports results. These are not only difficult to model, but are often also available only after circulation decisions have been made. It should further be noted that all three data sets have a relatively low average number of copies sold per outlet. In general, the higher this number, the higher the signal-to-noise ratio.

The prior mean M and covariance matrix Σ_A are also sensible. For example, the mean for the model parameters connected to the feature representing recent sales figures is always clearly positive. The leading eigenvalues of the covariance matrix Σ_A are such that the prior information is worth about half a year of sales figures. With higher numbers of hidden units, more and more eigenvalues become zero, basically shunting off the corresponding hidden unit. Prior means depending on task characteristics have been considered as well, and sometimes yield slightly better results. For example, the mean connected to a feature represent-

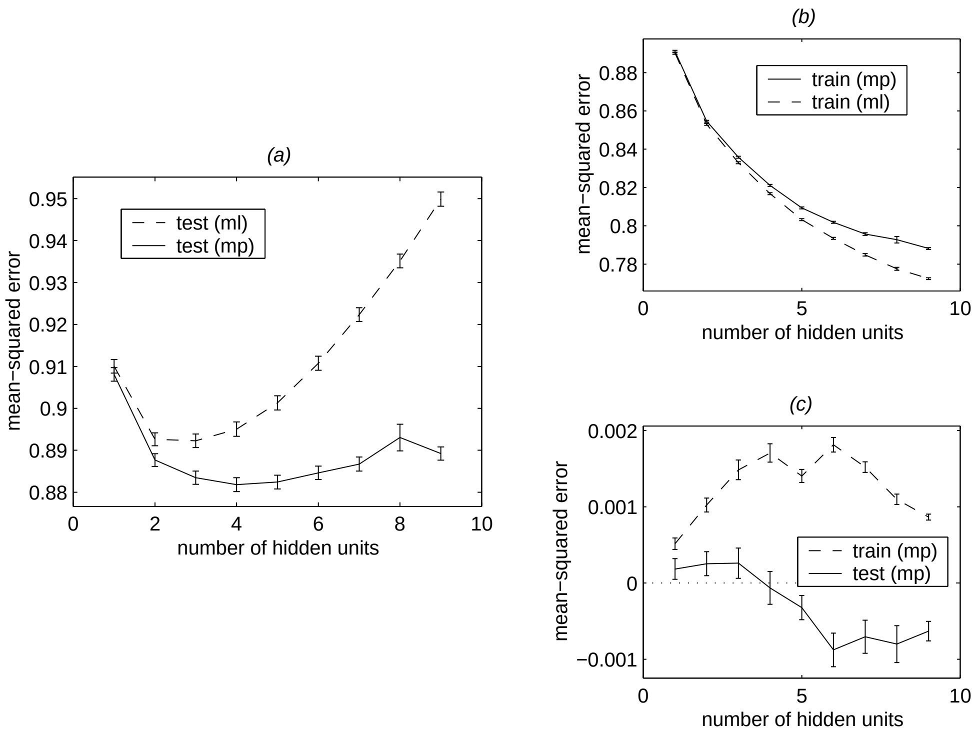

Figure 1. Mean-squared errors as a function of the number of hidden units. (a) Inter-task test error for most probable (solid) and maximum likelihood (dashed) solutions. (b) The same for the inter-task training error. (c) Difference between inter- and intra-task test (solid) and training (dashed) errors. The lines serve to guide the eye. Error bars give the standard deviation of the mean. Averages over 20 runs. See the text for further details.

ing seasonal patterns comes out significantly higher for outlets in touristic areas than for outlets in urban areas. More testing is needed, especially in situations with less training patterns per outlet where more accurate priors really start to pay off.

4.4 Performance with Increasing Hidden Units

In our model we have both a bottleneck of hidden units, reducing the inputs to a smaller set of features, and, on top of that, hyperparameters specifying a prior for the model parameters. Do we really need both? To check this we applied the simulation paradigm sketched above to data set III, varying the number of features and computing error measures not only for the most probable solutions $A_{\rm mp}$, but also for the maximum likelihood solutions $A_{\rm ml}$. The results for these maximum likelihood solutions indicate the performance that can be obtained in a frequentist multitask learning approach without taking into account prior information following e.g. Caruana (1997) and Pratt and Jennings (1996). On purpose we considered a data set with a relatively small number of inputs ($n_{\rm inp} = 9$). This makes it feasible to go all the way up to $n_{\rm hid} = n_{\rm inp}$, which is equivalent to the case of no bottleneck. Loosely speaking, the maximum likelihood test error for $n_{\rm hid} = n_{\rm inp}$ is the test error for single-task learning. The results are shown in Figure 1. In each run $n = 500$ tasks were available for fitting the hyperparameters and another 500 for evaluation, with $N = 92$ training patterns.

Looking at the inter-task test error on the lefthand side, it can be seen that both the feature matrix and the priors for the model parameters improve performance. The test error based on the maximum likelihood solutions neglecting the prior rapidly grows with increasing number of features, yielding by far the worst performance with all tasks treated separately. The optimum for the most probable solutions is obtained for $n_{\rm hid} = 4$. Even although the priors on the model parameters control the risk of overfitting rather well, the test error for $n_{\rm hid} = 4$ is significantly lower than the one for $n_{\rm hid} = n_{\rm inp}$. In short, the main aspects of the model, feature reduction and exchangeability of model parameters, work well on this set of regression tasks and validate the concept of “learning to learn”.

The upper right graph shows that the inter-task train-

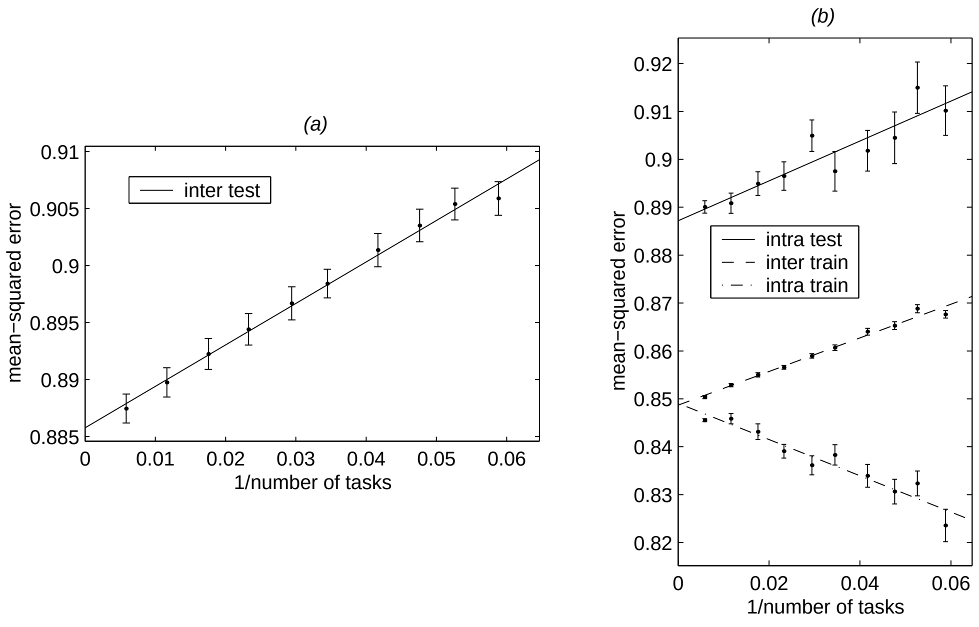

Figure 2. Mean-squared errors as a function of the number of tasks used for fitting the hyperparameters. (a) Learning-to-learning curve: the inter-task test error as a function of the number of tasks. (b) Other training curves: intra-task test error, inter-task training error, and intra-task training error (from top to bottom). The lines are the best linear fits. Error bars give the standard deviation of the mean. Averages over 75 runs. See the text for further details.

ing error indeed decreases as a function of the number of features, for the maximum likelihood solutions a little faster than for the most probable solutions. The lower right graph gives the difference between the inter- and intra-task training and test errors for the most probable solutions. These differences, although mostly significant, are very small on the scale of the other two graphs. Roughly speaking, they measure the impact of a single task on the hyperparameters optimized on a set of n tasks. This impact scales with $1/n$, yielding very small differences for the number of tasks $n = 500$ in this simulation. The intra-task training error is consistently lower than the inter-task training error, as could be expected. The same difference for the test error changes sign around the optimal number of features: with less hidden units the inter-task test error is higher than the intra-task test error, and vice versa with more hidden units. In other words, when overfitting it helps to optimize the hyperparameters on an independent set of tasks, *excluding* the task under consideration. We have observed similar behavior in other simulations, but do not yet fully understand it.

4.5 Learning-to-Learning Curves

In another set of simulations on data set I, we derived the learning curves of Figure 2. Within each run, tasks were split up randomly in 50% tasks for training and 50% for testing, as before. But now we varied the number of (training) tasks used for optimizing the hyperparameters from $n = 17$ to the maximal $n = 171$. We considered a network with $n_{\text{inp}} = 20$, $n_{\text{hid}} = 3$, and $n_{\text{prior}} = 1$ with $N = 122$ training patterns per task.

The graph on the lefthand side yields a "learning-to-learning" curve: the inter-task test error as a function of the number of tasks used to optimize the hyperparameters. On the right are the learning curves for the intra-task test, inter-task training, and intra-task training error. The simulations strongly suggest a linear relationship between these errors and the inverse of the number of tasks. Although the exact conditions and error measures differ, this relationship is well in line with the theoretical findings of Baxter (1997).

Here we will sketch a simpler alternative argument, to be worked out more precisely in a separate article. First we consider the difference between the intra-task training error and the inter-task training error. Call E_∞ the training error that would be obtained with an infinite number of tasks available for fitting the hyperparameters. With a finite number of tasks, overfitting will reduce the intra-task training error, yet increase the inter-task training error. Assuming that the training error is the dominant term in the loss function $L(\Lambda)$, loose application of Akaike's (1974) information criterion yields

$$E_\infty - E^{\text{intra}}_{\text{train}} \approx E^{\text{inter}}_{\text{train}} - E_\infty \approx \frac{\sigma^2|\Lambda|}{Nn} ,$$

with $|\Lambda|$ the (effective) dimension of the hyperparameters Λ. Based on this expression, we would predict an absolute slope of 0.4, fairly close to the fitted -0.38 for the intra-task training error and 0.42 for the inter-task training error. Furthermore, the intra- and inter-task training error indeed seem to have the same intercept.

Next we compare the inter-task training and inter-task test error. They are measured based on the same set of hyperparameters, obtained on an independent set of tasks. Therefore we can again apply Akaike's information criterion, but now considering the effect of optimizing the model parameters on the training data. Neglecting the effect of the prior, we get

$$E^{\text{inter}}_{\text{test}} - E^{\text{inter}}_{\text{train}} \approx \frac{2\sigma^2|A|}{N} ,$$

with $|A|$ the dimension of model parameters. This crude estimate yields a difference of 0.36 to be compared with 0.37 for the experimental fit. Furthermore, it suggests that the slope for the inter-task test error is the same as for the inter-task training error, which is also experimentally verified. There is no significant difference between the experimental fits for the intra-task and inter-task test errors, but again, we do not know how to link these theoretically.

5. Discussion and Outlook

In this article we have presented a new model for multitask learning, analyzed within an empirical Bayesian framework. Model parameters specific to each task and hyperparameters shared between tasks are treated at a different level, as advocated in Baxter (1997). Compared with the multitask learning approaches of Caruana (1997) and Pratt and Jennings (1996), where input-to-hidden weights B and hidden-to-output weights A are treated at the same frequentist level, the Bayesian approach has several advantages.

First of all, the Bayesian approach facilitates the incorporation of prior knowledge, the parameters of which, β and Σ_A, are learned in the same manner as the feature matrix B. This turns the maximum likelihood parameters A_{ml} into most probable parameters A_{mp}, significantly reducing the risk of overfitting. Secondly, the Bayesian approach naturally takes into account the variability of the model parameters A around their most probable values A_{mp}. In the model considered in this article, the model parameters can be integrated out explicitely, yielding a nonlinear optimization problem for all hyperparameters. This is in contrast with Heskes (1998), where first the feature matrix B was learned in a standard frequentist manner, after which the hyperparameters for the prior were obtained in an (empirical) Bayesian fashion using an EM algorithm.

The empirical Bayesian approach shifts the machine learning problem from learning model parameters to learning hyperparameters. From a technical point of view the basic learning problem stays the same, but now appears at a higher level. At this higher level, we can apply many of the algorithms originally designed for model parameters, such as e.g. pruning techniques.

The inference of the hyperparameters is in fact handled in a frequentist rather than a full Bayesian manner. This seems fine for the large amount of tasks in our simulations, but for smaller problems it would be better to take the full Bayesian approach and sample over the hyperparameters. Another alternative is to approximate their distribution using a standard Laplace approximation (see e.g. Robert, 1994). A full hierarchical Bayesian approach would bring our model even closer to the framework of Baxter (1997).

Even now, under quite different conditions, the experimentally obtained learning curves have the same flavor as the results of Baxter (1997), namely

$$E \approx E_0 + \frac{1}{N}\left(a + \frac{b}{n}\right) ,$$

with E_0 some base-line error, a related to the number of model parameters per task, b related to the number of hyperparameters, N the number of patterns per task, and n the number of tasks. Our first crude explanation of these and other experimental observations leaves plenty of room for better analysis. For example, the differences between intra-task test and inter-task test error are puzzling and not yet understood. Furthermore, it would be interesting to study the dependency of the optimal hyperparameters on the sufficient statistics. This might be done in a student-teacher paradigm, often used in statistical mechanics studies of learning single tasks.

To be useful as a robust and accurate multitask prediction system, we need to relax some of the simplifying assumptions and extend the model. Nonlinear hidden units are not really relevant in the noisy environment of newspaper and magazine sales. More important is an integration with (Bayesian) methodology for time series prediction. But even without these improvements, the current model allows for rapid testing of architectures and all kinds of other options, which would otherwise be infeasible.

Acknowledgments

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

Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, *19*, 716–723.

Baldi, P., & Hornik, K. (1989). Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks*, *2*, 53–58.

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

Bryk, A., & Raudenbusch, S. (1992). *Hierarchical linear models*. Newbury Park: Sage.

Caruana, R. (1997). Multitask learning. *Machine Learning*, *28*, 41–75.

Heskes, T. (1998). Solving a huge number of similar tasks: a combination of multi-task learning and a hierarchical Bayesian approach. *Proceedings of the International Conference on Machine Learning* (pp. 233–241). San Mateo: Morgan Kaufmann.

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

Robert, C. (1994). *The Bayesian choice: A decision-theoretic motivation*. New York: Springer.

Thrun, S., & Pratt, L. (Eds.). (1997). *Learning to learn*. Dordrecht: Kluwer Academic.

Appendix

In this appendix we describe the loss function $L(\Lambda)$, the function that should be minimized with respect to the hyperparameters Λ in the empirical Bayesian approach. Its derivation is really just a matter of bookkeeping, with perhaps the only difficulty the inversion of the noise covariance matrix Σ_ϵ for nonzero ρ. Here the decomposition

$$\Sigma_\epsilon = \sigma^2(1+n\rho^2)U_2 + \sigma^2 U_1 \; , \qquad (4)$$

with $U_2 = \mathbf{1}/n$ ($\mathbf{1}_{ij} = 1\ \forall_{ij}$) and $U_1 = \mathbf{I} - U_2$ two orthogonal projection matrices, may help.

For explainability, we decompose the result:

$$L(\Lambda) = L_0(\Lambda) + \frac{Nn}{2\sigma^2}[L_1(\Lambda) - (1-\kappa)L_2(\Lambda)] \; ,$$

with $\kappa = [1+n\rho^2]^{-1}$ a function of ρ; $\kappa = 1$ for $\rho = 0$. The first term $L_0(\Lambda)$ contains several normalization terms, all independent of the data except for the number of patterns N and the number of tasks n:

$$\begin{aligned} L_0(\Lambda) &= \frac{N}{2}(n\log\sigma^2 - \log\kappa) + \frac{1}{2}\log\det(\kappa\tilde{\Sigma}_A + \mathbf{I}) \\ &\quad + \frac{1}{2}(n-1)\log\det(\tilde{\Sigma}_A + \mathbf{I}) \; . \end{aligned}$$

Here $\tilde{\Sigma}_A = N\Sigma_A/\sigma^2$ is the prior covariance matrix Σ_A relative to the uncertainty induced by the noise variance. $\tilde{\Sigma}_A$ of order one means that prior information is given as much weight as the data specific to each task.

$L_1(\Lambda)$ is further subdivided in two terms:

$$L_1(\Lambda) = L_{11}(B) + L_{12}(B, \beta, \tilde{\Sigma}_A) \; ,$$

and similarly for $L_2(\Lambda)$. Differences between L_1 and L_2 can be traced back to the decomposition (4): L_1 collects all terms related to the identity matrix in U_1, L_2 those from U_2. L_{11} is the standard mean-squared error considered in Baldi and Hornik (1989):

$$L_{11}(B) = \left\langle \langle y^2 \rangle_{\text{tasks}} \right\rangle_{\text{patterns}} - \text{Tr}\; BR_{xx}B^T \; ,$$

with $R_{xx} = \Sigma_{xy}\Sigma_{xy}^T/n$ a covariance matrix of covariances. The simple form derives from the constraint (3). The corresponding term for L_2 reads

$$L_{21}(B) = \left\langle \langle y \rangle^2_{\text{tasks}} \right\rangle_{\text{patterns}} - R_x^T B^T B R_x \; ,$$

where $R_x = \langle \Sigma_{xy} \rangle_{\text{tasks}}$ is the input-output covariance averaged over all outputs. The remaining terms are mainly a function of the hyperparameters β and $\tilde{\Sigma}_A$, but also depend on B and κ. βz plays a role similar to $B\Sigma_{xy}$. With further definitions $R_z = \langle z \rangle_{\text{tasks}}$, $R_{xz} = \Sigma_{xy}z^T/n$, and $R_{zz} = zz^T/n$, we can write

$$\begin{aligned} &L_{12}(B, \beta, \tilde{\Sigma}_A) = \\ &\text{Tr}\left\{ \left[BR_{xx}B^T - 2BR_{xz}\beta^T + \beta R_{zz}\beta^T \right] \left[\tilde{\Sigma}_A + \mathbf{I} \right]^{-1} \right\} \end{aligned}$$

and

$$\begin{aligned} &L_{22}(B, \kappa, \beta, \tilde{\Sigma}_A,) = \\ &(BR_x - \beta R_z)^T [\kappa\tilde{\Sigma}_A + \mathbf{I}]^{-1} [\tilde{\Sigma}_A + \mathbf{I}]^{-1} (BR_x - \beta R_z) \; . \end{aligned}$$