Accelerating Cyclic Update Algorithms for Parameter Estimation by Pattern Searches

Antti Honkela, Harri Valpola and Juha Karhunen
Helsinki University of Technology, Neural Networks Research Centre, P.O. Box 5400, FIN-02015 HUT, Finland.
({antti.honkela,harri.valpola,juha.karhunen}@hut.fi *)*

March 4, 2003

Abstract. A popular strategy for dealing with large parameter estimation problems is to split the problem into manageable subproblems and solve them cyclically one by one until convergence. A well-known drawback of this strategy is slow convergence in low noise conditions. We propose using so-called pattern searches which consist of an exploratory phase followed by a line search. During the exploratory phase, a search direction is determined by combining the individual updates of all subproblems. The approach can be used to speed up several well-known learning methods such as variational Bayesian learning (ensemble learning) and expectation-maximization algorithm with modest algorithmic modifications. Experimental results show that the proposed method is able to reduce the required convergence time by 60–85 % in realistic variational Bayesian learning problems.

Keywords: alternating optimization, Bayesian methods, cyclic updating, line search, parameter estimation, pattern search, variational Bayesian learning.

1. Introduction

Several advanced statistical learning methods for neural networks and graphical models are based on a generative approach where the goal is to find a specific model which explains how the observations were generated. It is assumed that there exist certain latent variables (also called factors, sources, or hidden variables or causes) which have generated the observed data through an unknown mapping. In unsupervised learning, the goal is to identify both the unknown latent variables and the generative mapping, while in supervised learning it suffices to estimate the generative mapping.

Especially in unsupervised generative learning of large-scale data sets, the number of unknown parameters to be estimated easily grows very high. This is because all the source values corresponding to each observation vector are essentially unknown parameters which must be estimated from the available data. In this kind of situations, it is computationally prohibitive to estimate all the unknown parameters or values simultaneously. Instead, one must resort to simpler cyclic estimation schemes, also known as alternating optimization or grouped coordinate optimization. In these schemes, one parameter or a group of parameters is picked up at a time and optimized while keeping the other parameters frozen to their current values. Repeating this process for different sets of parameters, their values eventually converge to an optimum [3].

In particular, this strategy is utilized in the variational Bayesian techniques [12], which decouple the difficult problem of describing the posterior probability density of the model parameters into many smaller tractable problems. It is well known that in the case of low noise, the posterior dependences become stronger. This in turn makes the cyclic estimation procedure slow. It is difficult to optimize the values of dependent parameters one at a time since any one of them can only be changed very little if the other parameters stay constant. In this paper we propose a method based on pattern searches for speeding up the convergence of the estimation procedure in such cases.

In the next section, we briefly describe Bayesian methods which apply this type of cyclic estimation. In Section 3, we present a simple example illustrating the considered problem. The pattern search algorithm for accelerating estimation of parameters is explained in Section 4. In Section 5, the proposed method is applied to example problems demonstrating its effectiveness. The paper concludes with a discussion and suggestions for future work.

2. Bayesian estimation methods

Denote by $\boldsymbol{\theta}$ the set of model parameters and all the unknown variables that we wish to estimate from a given data set $\boldsymbol{X}$. In Bayesian estimation methods, it is assumed that there is some prior information on $\boldsymbol{\theta}$ available. This is represented in the form of prior distribution $p(\boldsymbol{\theta})$ of $\boldsymbol{\theta}$. After learning, all the information of the parameters is contained in the posterior probability density $p(\boldsymbol{\theta}|\boldsymbol{X})$ of the parameters given the data $\boldsymbol{X}$. It can be computed from the Bayes' rule

$$p(\boldsymbol{\theta}|\boldsymbol{X}) = \frac{p(\boldsymbol{X}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\boldsymbol{X})}. \tag{1}$$

Here $p(\boldsymbol{X}|\boldsymbol{\theta})$ is the likelihood of the parameters $\boldsymbol{\theta}$, and $p(\boldsymbol{X})$ is a normalizing constant which can be evaluated if necessary by integrating the numerator $p(\boldsymbol{X}|\boldsymbol{\theta})p(\boldsymbol{\theta})$ over all the possible values of $\boldsymbol{\theta}$.

Because integration over parameter space that is needed to evaluate the normalising constant and to use the posterior otherwise is difficult, using the exact posterior is typically intractable, and some approximations are needed. The simplest approximations are based on using point estimates such as maximum a posteriori (MAP) estimate or slightly more advanced EM (expectation maximization) algorithm [5, 17]. In EM algorithm, the parameters are divided into two sets which are then updated cyclically. The algorithm employs point estimates for some of the parameters, which can lead to problems with model order estimation and overfitting [4].

Variational methods form one class of more advanced approximations. Their key idea is to approximate the exact posterior distribution $p(\boldsymbol{\theta}|\boldsymbol{X})$ by another distribution $q(\boldsymbol{\theta})$ that is computationally easier to handle [12]. The approximating distribution is usually chosen to be a product of several independent distributions, one for each parameter or a set of similar parameters. We use a particular variational method known as ensemble learning that has recently become very popular [8, 15, 14].

In ensemble learning the optimal approximating distribution is found by minimizing the Kullback-Leibler divergence between the approximate and true posterior. After some considerations [14, 15], this leads to the cost function

$$\mathcal{C} = \int_{\boldsymbol{\theta}} q(\boldsymbol{\theta}) \ln \frac{q(\boldsymbol{\theta})}{p(\boldsymbol{X}, \boldsymbol{\theta})} d\boldsymbol{\theta}. \tag{2}$$

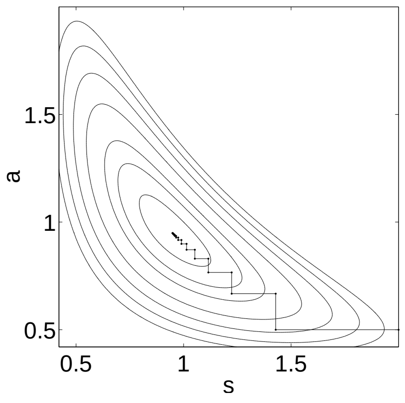
The standard procedure is to optimize the cost function with respect to different groups of the parameters cyclically.

During the last years, variational Bayesian learning has gained popularity because of its desirable properties. It largely avoids the model order estimation and overfitting problems related to the maximum likelihood (ML) and maximum a posterior (MAP) estimation. The method has been successfully applied to various models such as hidden Markov models (HMMs) [16], independent factor analysis [1], nonlinear independent component analysis [13], as well as switching and nonlinear state-space models [7, 18]. It has also been used as a foundation of a general framework of building blocks for latent variable models [20]. This building block framework is employed in the experimental part of this paper.

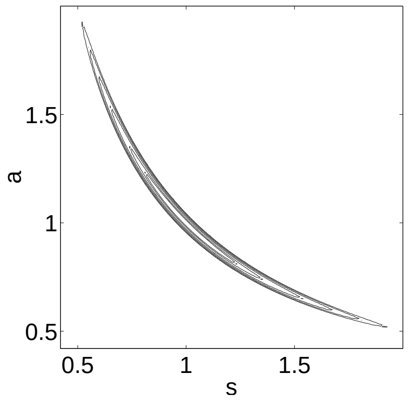
3. A simple illustrative example

As an example, consider the simple linear generative model $\mathbf{x} = \mathbf{A}\mathbf{s} + \mathbf{n}$ used in noisy independent component analysis (ICA) [10]. There $\mathbf{x}$ denotes the known observation vector, $\mathbf{s}$ the unknown source vector, $\mathbf{A}$ the unknown mixing matrix, and $\mathbf{n}$ a vector of additive noise. In ICA, the mixing matrix $\mathbf{A}$ is chosen so that the components of the source vector $\mathbf{s}$ become as independent as possible. In this problem, the mixing matrix $\mathbf{A}$ and the source vectors $\mathbf{s}$ are intimately tied. Changing $\mathbf{A}$ must be compensated by a corresponding change in $\mathbf{s}$ and vice versa to retain the value of the observed vector $\mathbf{x}$. If $\mathbf{A}$ and $\mathbf{s}$ are updated separately, quite large number of steps may be required to find the correct values because both $\mathbf{A}$ and $\mathbf{s}$ can be changed only little at each step.

Take now the simplest possible model $x = as + n$ where the observation x, source s, noise n, and the mixing coefficient a are all scalars. Though seemingly trivial, this case already illustrates several problems encountered in more difficult generalizations, and it is easy to visualize. Assume that there is a single observation $x = 1$, and that the variables a and s have Gaussian priors with zero mean and unit variance. The noise term n is also Gaussian with zero mean and predefined variance σ_n^2 whose value is changed in different experiments. Variational Bayesian learning is used to solve this problem by assuming that a and s are statistically independent.



a. Noise variance $\sigma_n^2 = 10^{-1}$



b. Noise variance $\sigma_n^2 = 10^{-3}$

Figure 1. Contour plots of the posterior distribution of the parameters of the linear model $x = as+n$ under different noise levels. The convergence of the means of a and s in cyclic iteration to the optimum is shown in the left subfigure. A corresponding iteration in the right subfigure would need to take much shorter steps and thus would need many more iterations.

It turns out that the optimal approximate posterior distributions of a and s are Gaussian. Ignoring the variances of the Gaussians, the cost function of posterior means of a and s is quite similar to the posterior distributions shown in Fig. 1. For a moderate noise variance of $\sigma_n^2 = 10^{-1}$ in the left subfigure, a and s depend on each other only weakly. The figure shows that the cyclic iteration scheme converges relatively quickly in this case.

When the noise variance decreases, the optimal region of the cost function becomes narrower. For the noise variance $\sigma_n^2 = 10^{-3}$ in the right subfigure, the optimization problem is already much more difficult, and convergence of the cyclic iteration slows down considerably. This problem becomes more pronounced in higher-dimensional real-world problems.

4. The algorithm

We propose speeding up convergence of cyclic update schemes by applying the idea of pattern searches introduced by Hooke and Jeeves in [9]. The pattern search consists of two phases. In the first exploratory phase, the objective function is optimized in each coordinate direction separately as usual. This phase is called *parameter-wise update phase.* The updates performed there are then combined to form a diagonal direction for the *line search.*

Pattern search methods have mostly been abandoned in standard optimization literature in favor of conjugate gradient and other more advanced methods. Using such methods with the variational Bayesian approach would, however, require significant changes to the existing algorithms, and would not utilize the ability to solve independent subproblems easily. The pattern search approach takes advantage of the existing methodology for performing the parameter-wise updates and requires only minor changes to the update process as illustrated by the algorithm optimize_pattern below. It is especially noteworthy that the pattern search approach does not need the derivatives of the cost function.

Assume that we are optimizing the function $C : \mathbb{R}^n \to \mathbb{R}$, and denote by $\mathbf{d}_1, \ldots, \mathbf{d}_n$ the standard basis of $\mathbb{R}^n$. One iteration of the parameter-wise updates can be implemented as follows.

```
function optimize_parameter-wise($C, \mathbf{z}_1$):
    $\mathbf{z}_2 \leftarrow \mathbf{z}_1$
    for $i = 1, \ldots, n$:
        $\lambda \leftarrow \mathrm{argmin}_\lambda\, C(\mathbf{z}_2 + \lambda \mathbf{d}_i)$
        $\mathbf{z}_2 \leftarrow \mathbf{z}_2 + \lambda \mathbf{d}_i$
    return $\mathbf{z}_2$
```

In practice the parameter-wise updates are not performed using a general purpose optimization algorithm. In the variational Bayesian approach the problem is split into smaller subtasks. They can be solved more easily, which is utilized here. The ability to solve these subtasks analytically in many problems is in fact one of the most important reasons for using a cyclic update scheme in the first place. The updates can also be carried out for a larger group of parameters at a time rather than for just one parameter.

The complete pattern search scheme is a straightforward extension to this standard algorithm:

```
function optimize_pattern($C, \mathbf{z}_1$):
    $\mathbf{z}_2 \leftarrow$ optimize_parameter-wise($C, \mathbf{z}_1$)
    $\Delta\mathbf{z} \leftarrow \mathbf{z}_2 - \mathbf{z}_1$
    $\lambda \leftarrow \mathrm{argmin}_\lambda\, C(\mathbf{z}_1 + \lambda \cdot \Delta\mathbf{z})$
    $\mathbf{z}_3 \leftarrow \mathbf{z}_1 + \lambda \cdot \Delta\mathbf{z}$
    return $\mathbf{z}_3$
```

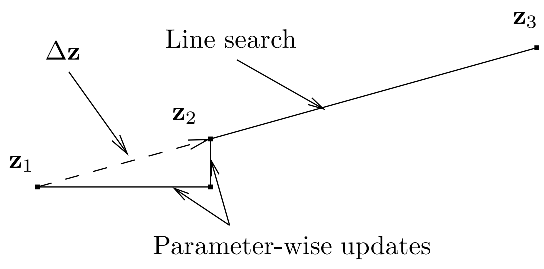


Figure 2. Illustration of the pattern search algorithm.

The algorithm is illustrated in Fig. 2. The figure shows how the direction of the line search is obtained as the difference vector $\Delta\mathbf{z} = \mathbf{z}_2 - \mathbf{z}_1$ of the value of the parameters $\mathbf{z}_2$ after the parameter-wise update round and the corresponding value $\mathbf{z}_1$ before the updates. Parameters such as variances that have a positivity constraint are treated on logarithmic scale for the purposes of evaluating the difference and later when extrapolating for the iterates $\mathbf{z}_1 + \lambda \cdot \Delta\mathbf{z}$. This leads to formulas

$$\Delta\sigma := \log\sigma_2 - \log\sigma_1 = \log\frac{\sigma_2}{\sigma_1} \tag{3}$$

and

$$\log\sigma_3 = \log\sigma_1 + \lambda \cdot \Delta\sigma \ \Rightarrow\ \sigma_3 = \sigma_1 \exp(\lambda \cdot \Delta\sigma) \tag{4}$$

that are used for such parameters.

Figure 2 also shows that although the optimizations in the function optimize_pattern appear to be similar to those in optimize_parameter-wise, there is a big difference: the line search is made in an arbitrary direction instead of standard coordinate directions. In many cases such as ours, the parameter-wise optimizations in the directions of coordinate axes can be easily carried out analytically. The arbitrary line search is a very different operation, but essentially it requires only a method to evaluate the cost function of the model and a generic line search algorithm. Thus it is actually simpler than the standard update method which requires at least some local gradient information.

The computational requirements of the line search are typically comparable to a few rounds of parameter-wise optimizations. The little extra work really pays off as the value of λ can easily be more than 100. Therefore a single line search can save at least that many ordinary rounds of optimization.

4.1. Line searches

The line search in the function optimize_pattern can be implemented by any standard algorithm. As the other parts of the algorithm do not use derivatives of the cost function, it is reasonable to choose a derivative free line search algorithm. Good general candidates for such algorithms are the golden section method and the method of quadratic fit [6, 2].

The golden section method gives decent worst case performance, but does usually not perform as well as its competitors in more realistic situations. In our examples, the cost function appeared roughly quadratic along the line search direction. Hence a simple implementation of the method of quadratic fit with golden section method as a fallback for the most difficult cases was used. In practice the best method would probably be some kind of a combination of these two basic methods [6].

4.2. Implementation details

In standard Hooke–Jeeves algorithm there is only one cyclic optimization step between two line searches [2]. This turned out to be suboptimal for our purposes, because the benefits of the line searches with increased step length were rather small. The resulting algorithm was even slower than the standard update scheme in some cases.

Letting the iteration stabilize by doing several parameter-wise optimization phases before the next line search seemed to work much better. In the experiments we used ten parameter-wise optimization rounds between two consecutive line search steps. We also tried to average the search direction over several parameter-wise optimization rounds but this did not improve the results.

5. Experiments

We demonstrate the usefulness of the pattern search algorithm with three experiments using variational Bayesian learning. In the examples considered here, the approximate posterior is a product of independent Gaussian distributions. Hence it is completely characterized by the means and variances of these distributions. To ensure the validity of the values of the variances, they are treated on a logarithmic scale as explained in Section 4.

5.1. One-dimensional example

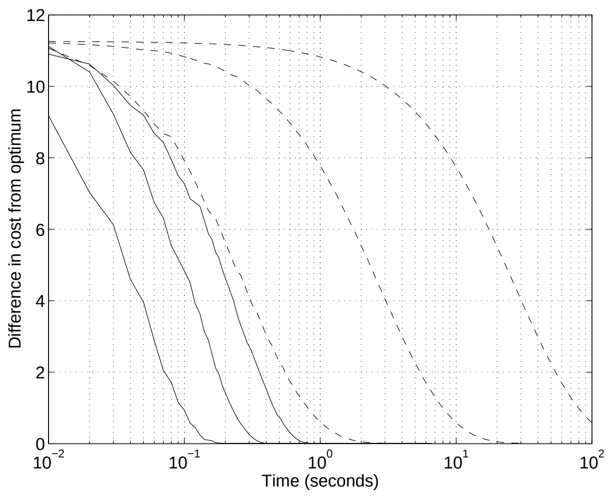


Figure 3. The cost function values attained by the pattern search algorithm (solid lines) and standard cyclic updating (dashed lines) in the simple one-dimensional example. The noise variance decreases by a factor of 10 between consecutive curves from left to right.

Consider first the very simple linear model $x = as + n$ discussed in Section 3 and in Figure 1. The progress of the optimization procedures as a function of time is illustrated in Figure 3. The cyclic iteration performs very badly in this case. For low noise levels, the length of the steps taken by the iteration is directly proportional to the noise variance. Thus halving the noise level (standard deviation) quadruples the time needed for the algorithm to converge. The pattern search method does much better, slowing down clearly less when the noise level decreases. The time scale is logarithmic so that a constant difference is in fact difference by a constant factor. The noise variance σ_n^2 decreases by a factor of 10 between the consecutive curves starting from the left.

5.2. Real ICA example

The method was also tested with a more realistic noisy independent component analysis (ICA) [10] model $\mathbf{x} = \mathbf{A}\mathbf{s} + \mathbf{n}$ where $\mathbf{n}$ is the noise vector. The data set used was an eight-dimensional artificial mixture $\mathbf{x}$ of four source signals $\mathbf{s}$. A varying amount of Gaussian noise was added to the data. The number of data points was 200. The prior distributions of the sources were Gaussians with varying variance, which corresponds to a super-Gaussian distribution. The actual simulations were run using the building block library discussed in [20]. All the parameters of the model had hierarchical priors which were also estimated from the data. The total number of parameters and hyperparameters was 1662. For each of them, both the mean and variance of the approximating posterior distribution are estimated in the variational Bayesian method.

The convergence times of different methods are shown in Figure 4 in a similar manner as in Fig. 3. The cyclic update algorithm behaves in a very similar manner as in the simple one-dimensional example. Finding the correct rotation for the sources takes a long time, and this time seems to increase in proportion to the inverse of the variance of additive noise. The pattern search method is again significantly faster, typically reaching the same value of cost function in around one fourth or one eighth of the time required by the cyclic update method.

5.3. Hierarchical nonlinear factor analysis example

The third experiment was conducted with a hierarchical nonlinear factor analysis model [19]. The data set used in the experiment was the same 20 dimensional artificial data generated by a random multi-layer perceptron (MLP) network from 8 independent random sources that was also used in [13]. The number of samples was 1 000. More details on the data set can be found in [13].

The generative model for the nonlinear model is basically $\mathbf{x}(t) = \mathbf{A}\mathbf{s}_2(t) + \mathbf{f}(\mathbf{s}_2(t))$ with MLP-like structure $\mathbf{f}(\mathbf{s}_2(t)) = \mathbf{W}_1\boldsymbol{\phi}(\mathbf{W}_2\mathbf{s}_2(t) + \mathbf{b}_2) + \mathbf{b}_1$ and activation function $\phi(t) = \exp(-t^2)$ applied componentwise. In order to avoid problems caused by propagation of variance through multiple paths (see [13]), the values of the hidden neurons are taken as latent variables $\mathbf{s}_1(t)$ so that actually

$$\mathbf{s}_1(t) \sim N(\mathbf{W}_2\mathbf{s}_2(t) + \mathbf{b}_2, \mathbf{\Sigma_1}) \tag{5}$$

and

$$\mathbf{x}(t) \sim N(\mathbf{A}\mathbf{s}_2(t) + \mathbf{W}_1\boldsymbol{\phi}(\mathbf{s}_1(t)) + \mathbf{b}_1, \mathbf{\Sigma}_{\mathbf{x}}), \tag{6}$$

where $\mathbf{\Sigma_1}$ is the (diagonal) covariance matrix of the first level sources and $\mathbf{\Sigma}_{\boldsymbol{x}}$ is the noise covariance matrix.

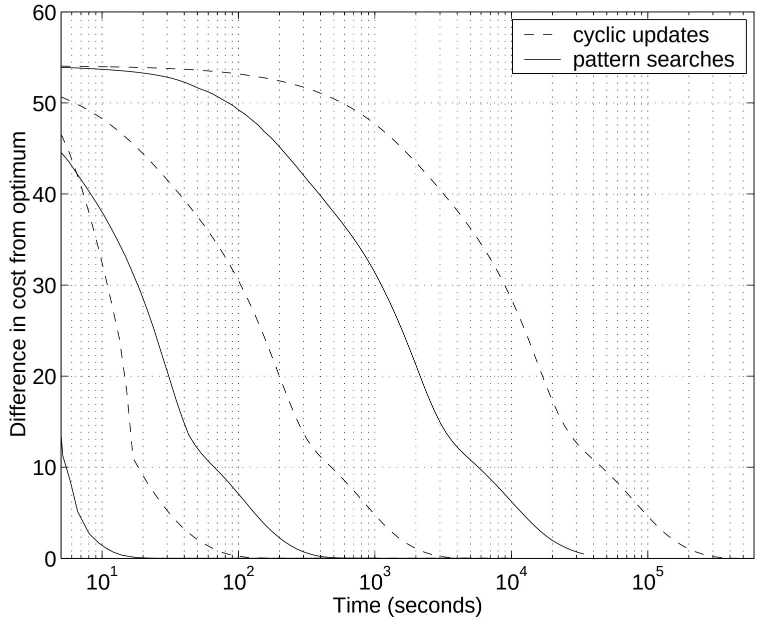


Figure 4. The cost function values attained by the pattern search algorithm (solid lines) and standard cyclic update algorithm (dashed lines) in the real ICA example. The noise variance decreases by a factor of 10 between consecutive curves starting from the left.

The results of applying the two optimization algorithms to this problem are illustrated in Fig. 5. The final value of the cost function attained and the time required for the convergence vary between different initializations, but the difference between standard cyclic updates and our pattern search method stays roughly the same. The comparison is, however, somewhat difficult as the methods do not always converge to the same local optimum. This multitude of local optima is an inherent feature of flexible nonlinear models such as the one used here. The pattern searches did not affect the average quality of the local optima that were found.

For Figure 5, we have selected four simulations that did converge to more or less same result. The results of the other simulations are similar: the simulations using pattern searches converge more quickly to some local minimum of the cost function than the standard method. This overall behaviour is demonstrated in Figure 6 which shows the average speedups in 20 simulations with different initialisations. As the different algorithms do not converge to the same point we have used the times needed to reach certain level of cost function value as measurement points. The levels were chosen to be 1000, 100, 10 and

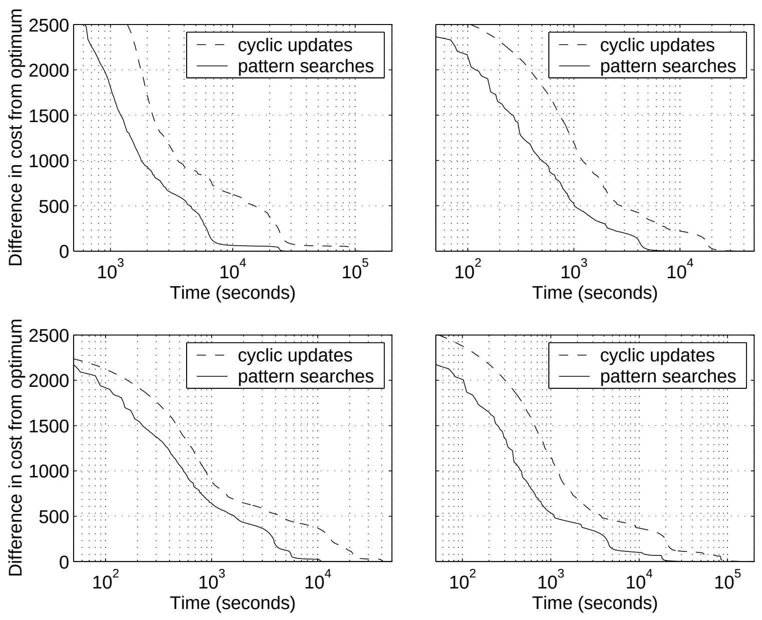


Figure 5. The cost function values attained by the pattern search algorithm (solid lines) and standard cyclic update algorithm (dashed lines) in the hierarchical nonlinear factor analysis example. The subfigures show the results of four experiments run with different initializations.

1 units above the worse local minimum found in that case. The wider confidence intervals in the last case are thus due to the fact that in some cases the convergence of the worse method has already started to slow down close to the local minimum whereas the other method is still descending rapidly. The figure is almost the same even if the levels are chosen for each method separately except that in that case the confidence intervals behave more smoothly.

Overall, the relative time savings in the hierarchical nonlinear factor analysis experiments are approximately the same as in the ICA experiment, around 60–80 %, with the mean of all the results used in Figure 6 being 73 % and the worst individual point 37 %.

6. Discussion

The pattern search algorithm is a straightforward extension of the standard cyclic update scheme, but it accelerates the convergence significantly. It is easy to implement and utilizes the ability to solve the independent optimization problems easily. All the operations required scale linearly with respect to the number of parameters in the

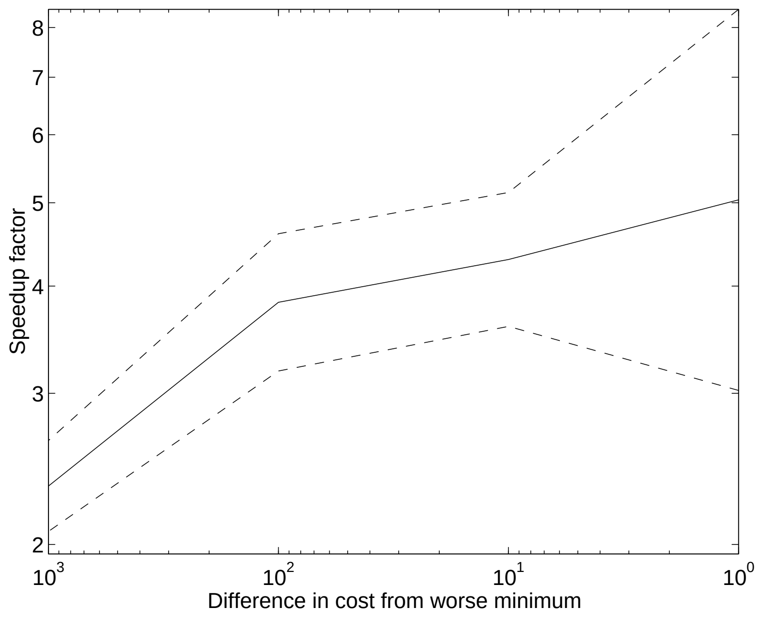


Figure 6. The average speedup obtained by pattern searches in different phases of learning. The speedup is measured by the ratio of times required by the basic algorithm and pattern search method to reach certain level of cost function value. The solid line shows the mean of the speedups over 20 simulation with different initialisations and the dashed lines show 99 % confidence intervals for the mean.

problem. The method could probably be refined to further improve its performance. In this paper, we have demonstrated that even in its current fairly simple form, the proposed method can provide significant speedups in learning.

In optimization literature pattern search methods have, however, mostly been abandoned and replaced by more advanced methods like conjugate gradient optimization. The conjugate gradient method is very different from the standard procedure used in variational Bayesian approach, because it requires derivatives of the cost function and does not take advantage of the ability to solve independent subproblems easily. It has anyway been used successfully for speeding up the EM algorithm in [11]. It would be interesting to study whether a similar method will work for other algorithms such as variational Bayesian learning as well, and how it would compare with the pattern search approach proposed in this paper. Nevertheless, the conjugate gradient method requires significant algorithmic modifications, and does not therefore directly compete with the pattern search approach, which attains a significant improvement in the convergence speed with minor modifications to the standard algorithm.

Acknowledgements

This research has been funded by the European Commission project BLISS, and the Finnish Center of Excellence Programme (2000–2005) under the project New Information Processing Principles. The authors thank Tapani Raiko for useful comments and discussions as well as Tomas Östman for helping with the third experiment. Additionally we would like to thank an anonymous reviewer for useful suggestions for improving the paper.

References

1. Attias, H.: 1999, 'Independent Factor Analysis'. *Neural Computation* **11**(4), 803–851.
2. Bazaraa, M. S., H. D. Sherali, and C. M. Shetty: 1993, *Nonlinear Programming: Theory and Algorithms*. J. Wiley, second edition.
3. Bezdek, J. C. and R. J. Hathaway: 2002, 'Some Notes on Alternating Optimization'. In: N. R. Pal and M. Sugeno (eds.): *Advances in Soft Computing – AFSS 2002*, Vol. 2275 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, pp. 288–300.
4. Bishop, C.: 1995, *Neural Networks for Pattern Recognition*. Clarendon Press.
5. Dempster, A. P., N. M. Laird, and D. B. Rubin: 1977, 'Maximum Likelihood from Incomplete Data via the EM Algorithm'. *J. of the Royal Statistical Society, Series B (Methodological)* **39**(1), 1–38.
6. Fletcher, R.: 1987, *Practical Methods of Optimization*. J. Wiley, second edition.
7. Ghahramani, Z. and G. E. Hinton: 2000, 'Variational Learning for Switching State-Space Models'. *Neural Computation* **12**(4), 963–996.
8. Hinton, G. E. and D. van Camp: 1993, 'Keeping neural networks simple by minimizing the description length of the weights'. In: *Proc. of the 6th Ann. ACM Conf. on Computational Learning Theory*. Santa Cruz, CA, USA, pp. 5–13.
9. Hooke, R. and T. A. Jeeves: 1961, "Direct Search' Solution of Numerical and Statistical Problems'. *J. of the ACM* **8**(2), 212–229.
10. Hyvärinen, A., J. Karhunen, and E. Oja: 2001, *Independent Component Analysis*. J. Wiley.
11. Jamshidian, M. and R. I. Jennrich: 1993, 'Conjugate Gradient Acceleration of the EM Algorithm'. *J. of the American Statistical Association* **88**(421), 221–228.
12. Jordan, M., Z. Ghahramani, T. Jaakkola, and L. Saul: 1999, 'An Introduction to Variational Methods for Graphical Models'. In: M. Jordan (ed.): *Learning in Graphical Models*. Cambridge, MA, USA: The MIT Press, pp. 105–161.
13. Lappalainen, H. and A. Honkela: 2000, 'Bayesian Nonlinear Independent Component Analysis by Multi-Layer Perceptrons'. In: M. Girolami (ed.): *Advances in Independent Component Analysis*. Berlin: Springer-Verlag, pp. 93–121.

14. Lappalainen, H. and J. Miskin: 2000, 'Ensemble Learning'. In: M. Girolami (ed.): *Advances in Independent Component Analysis*. Berlin: Springer-Verlag, pp. 75–92.
15. MacKay, D. J. C.: 1995, 'Developments in Probabilistic Modelling with Neural Networks – Ensemble Learning'. In: *Neural Networks: Artificial Intelligence and Industrial Applications. Proc. of the 3rd Annual Symposium on Neural Networks*. pp. 191–198.
16. MacKay, D. J. C.: 1997, 'Ensemble Learning for Hidden Markov Models'. Available from `http://wol.ra.phy.cam.ac.uk/mackay/`.
17. Neal, R. M. and G. E. Hinton: 1999, 'A View of the EM Algorithm that Justifies Incremental, Sparse, and Other Variants'. In: M. I. Jordan (ed.): *Learning in Graphical Models*. Cambridge, MA, USA: The MIT Press, pp. 355–368.
18. Valpola, H. and J. Karhunen: 2002, 'An Unsupervised Ensemble Learning Method for Nonlinear Dynamic State-Space Models'. *Neural Computation* **14**(11), 2647–2692.
19. Valpola, H., T. Östman, and J. Karhunen: 2003, 'Nonlinear Independent Factor Analysis by Hierarchical Models'. In: *Proc. 4th Int. Symp. on Independent Component Analysis and Blind Signal Separation (ICA2003)*. To appear.
20. Valpola, H., T. Raiko, and J. Karhunen: 2001, 'Building Blocks for Hierarchical Latent Variable Models'. In: *Proc. 3rd Int. Conf. on Independent Component Analysis and Signal Separation (ICA2001)*. San Diego, USA, pp. 710–715.