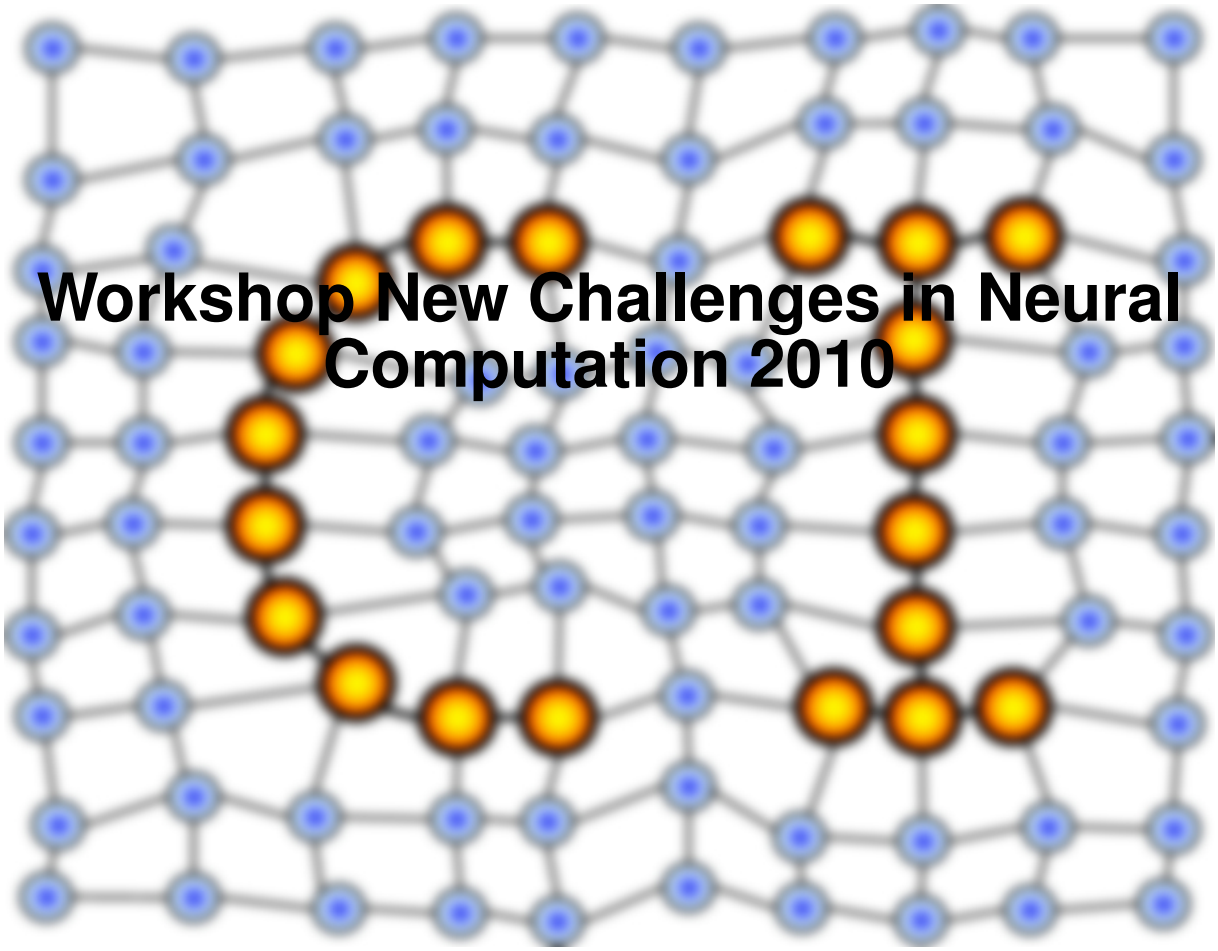
MACHINE LEARNING REPORTS



Workshop New Challenges in Neural Computation 2010

Report 04/2010
Submitted:13.09.2010
Published: 21.09.2010

Barbara Hammer[1] and Thomas Villmann[2] (Eds.)
(1) University of Bielefeld, Dept. of Technology CITEC - AG Computational Intelligence, Universitätsstrasse 21-23, 33615 Bielefeld
(2) University of Applied Sciences Mittweida, Technikumplatz 17, 09648 Mittweida, Germany

Table of contents:

New Challenges in Neural Computation NC^2 – 2010

Barbara Hammer[1] and Thomas Villmann[2]

1 – Cognitive Interaction Technology – Center of Excellence,
Bielefeld University, Germany
2 – Faculty of Mathematics / Natural and Computer Sciences,
University of Applied Sciences Mittweida, Germany

The quite remarkable history of artificial neural networks is well-known: In a nutshell, artificial neural networks were first proposed by McCulloch and Pitts as universal information processing mechanism. In the beginning, training algorithms were restricted to very simple architectures only, in essence single neurons. Thus, with the formal investigation of the limitations of these simple architectures in the famous book 'Perceptrons' by Minsky and Papert, the interest in neural networks rapidly decreased. Starting with the 80th, neural networks experienced a renaissance due to more complex architectures and training algorithms becoming available, in particular Hopfield networks for associative learning and optimization, self-organizing models mimicking phenomena of the human brain, and the famous backpropagation algorithm for training arbitrary feedforward networks. Nowadays, neural computation and biologically inspired data processing systems constitute essential topics in artificial intelligence accompanied by a well established theoretical foundation and numerous successful applications in science and industry. Several scientific journals as well as conferences are dedicated to this subject.

Research concerning neural networks can roughly be decomposed into two main areas: neuroinformatics – the goal of which is to develop biologically inspired algorithms for efficient information processing and to apply these algorithms to real life problems as occur in industry and every day life; and computational neuroscience – the goal of which is to develop biologically plausible models which can help to understand biological neural systems. Scientific landmarks in this latter area are set by the current Bernstein Network Computational Neuroscience established in Germany to focus research in this area, and the Blue Brain Project which final goal is to reverse engineer the mammalian brain. These combined efforts, among others, manifest the fact that Computational Neuroscience currently constitutes a rapidly emerging field with quite a few challenges being unsolved. The question of how information is encoded in the brain still constitutes one of the key open problems in this area, for example.

The standpoint of neuroinformatics within computational intelligence is less clear. Quite a few achievements have been reached in this area and, nowadays, neural networks constitute a standard model within modern data processing toolboxes available on the market. A well established theoretical foundation of neural networks has been achieved in particular in the frame of computational learning theory and Bayesian statistics, respectively. However, with respect to

research, one possible point of view is to think about neural networks as just another method of statistical data analysis. The specifics of neuroinformatics and its fundamental specific open questions and challenges in comparison to the field of general statistical data analysis are less clear. Overall, 'classical' neuroinformatics can be seen as a well emerged field with a solid theory and only a few remaining problems.

Modern information science is probably about to change this situation. We observe an ever increasing complexity of data and learning tasks which is caused by an increasing availability of electronic information in virtually all areas of daily life. This is possible due to improved technology concerning sensors, data storage, dedicated data formats, and the like. In consequence, machine learning in general and neural networks in particular have to deal with more and more complex scenarios such as occur, e.g., in modern robotics, biodata analysis, web technologies, etc. While classical neural networks have been developed to solve comparably narrow machine learning tasks such as inferring a function from vector data, nowadays, neural networks are used in settings where no simple objective is given and the methods have to partially generate reasonable training settings by themselves. Further, data sets are increasing rapidly with respect to its size, heterogeneity, and complexity. Two prime examples of this situation are formalized in two newly established priority programs of the German Science Foundation: Scalable Visual Analytics, which deals with intelligent mechanisms to help persons in the ill-posed domain of data visualization, and autonomous learning, which refers to complex learning scenarios as occur in natural environments. This poses new challenges towards the area of neuroinformatics such as the necessity to deal with very large or streaming data, data settings where the underlying distribution is not i.i.d., partially ill-posed domains, etc. Possibly, even another renaissance of neural networks will occur within these new challenges; certainly, at least a few novel and challenging issues will have to be tackled within this frame.

In the workshop NC^2, exemplary challenges and novel developments of neural systems are covered. Eight high quality contributions shed some light on diverse aspects which are currently at the frontiers of research in neural networks, in particular learning in complex environments and dealing with complex inputs in a biologically plausible manner. The paper 'A vision for reinforcement learning and its implications for neural computation' by S. Lange and M. Riedmiller centers around reinforcement learning (RL) as one key technology to deal with complex learning scenarios where only a reinforcement signal is given rather than explicit teacher information. The authors focus on recent developments in RL which carry the promise to turn RL technology from toy settings towards realistic scenarios, facing in particular the problem of complex state representations and scalability of the techniques to realistic environments. Restricted Boltzmann machines offer one very promising technique to efficiently encode complex signals to arrive iteratively at sparse representations of complex signals which encode states of the environment, for example, with high biological plausibility. Training, however, phases severe problems. The article 'Challenges in training

restricted Boltzmann machines' by A. Fischer and C. Igel investigates possibilities, to optimize the likelihood function in this setting using numerically stable and fast optimization such as resilient propagation. The contribution 'Mental imagery in artificial agents' by A. Kaiser, W. Schenk, and R. Möller deals with the simulation of a phenomenon in artificial agents which occurs frequently in natural agents in complex environments: the prediction of sensory experiences caused by certain actions without actual sensory inflow. A biologically plausible adaptive architecture is proposed in the context of visual inputs caused by robotics manipulations. In the article 'Direct policy search: intrinsic vs. extrinsic perturbations' by V. Heidrich-Meisner and C. Igel, the authors deal with the area of reinforcement learning in general, and they discuss possibilities to substitute the common dynamic programming framework by evolutionary optimization in case only few intermediate reward is available for the process.

F. Beuth, J. Wiltschut, and F. Hamker deal with the chicken and egg problem how to realize attention and how to recognize objects in images in their contribution 'Attentive stereoscopic object recognition'. They propose a neural architecture which can solve this problem in a biologically plausible manner. The next two contributions deal with challenges as arise when neural methods are applied in the biomedical domain. The contribution 'Clustering of hyperspectral image signatures using neural gas' by U. Seiffert and F. Bollenbeck centers around the analysis of hyperspectral images and its problems when data are only partially labeled. As for hyperspectral images, genome wide association studies face the problem of extremely high dimensional data. In the contribution 'Interpretive risk assessment on GWA data with sparse linear regression' by I. Brænne, K. Labusch, T. Martinetz, and A. Madany Mamlouk, single relevant variables are identified using various methods with inherent generalization including SVM and sparse linear regression. Finally, the contribution 'Ensemble methods for probability density estimation' by M. Glodek, M. Schels, and F. Schwenker aims at a central problem data analysis: density estimation. The standard EM approach which cannot be used to reliably estimate the number of modes for a Gaussian mixture is extended by ensemble techniques, such that a better stability and robustness can be achieved for realistic settings.

Altogether, these contributions constitute promising steps into the direction of complex information processing with neural systems by providing new paradigms, concepts, models, and benchmark scenarios.

A Vision for Reinforcement Learning and its Implications for Neural Compuation

Sascha Lange and Martin Riedmiller

Albert-Ludwigs-Universität Freiburg - Dept. of Computer Science - Germany

Abstract. New methods in reinforcement learning like policy gradients and batch reinforcement learning as well as a better understanding of the strengths and limits of particular combinations of reinforcement learning and function approximation have allowed for a significant step forward, when it comes to stable and efficient learning on real systems. Due to this progress, old research questions could been answered and completely new arise. With the goal of giving some insights into where the field of reinforcement learning is heading, this paper briefly presents the history of reinforcement learning, continues with a short (subjective) discussion of the present state of the art and finally comes up with ideas for new research directions. A special focus is put on the implications of these developments to neural computation.

1 A Very Brief History of Reinforcement Learning

The early optimism surrounding reinforcement learning (RL) was severely tested, as it became clear that the transition from learning in small discrete state spaces towards solving real problems on real systems would not be as easy as expected but would cause severe problems concerning the stability and complexity—even in rather simple problem settings. Another drawback was the insight multi-layer perceptrons and other popular function approximators do not guarantee stable convergence with common reinforcement learning update rules [1, 2], but—even worse—may lead to divergence in the general case. This was even a more severe strike as researchers had quite high hopes for these techniques as they had allowed for a number of impressive early successes in large and continuous state spaces (see e.g. [3]).

In the following years the research efforts concentrated on finding stable approximative versions of the known RL methods and on scaling them to continuous learning tasks on real systems. At the main focus were tasks from closed loop control that still were of rather limited complexity. However, in the recent past, there has been made significant progress. Thanks to new methods like policy gradients [4–6] and batch reinforcement learning [7–10] it's now possible to reliably solve many of these real-world problems directly by learning from interacting with the real system [5, 11–15]. The number of necessary interactions for learning good policies was reduced by orders from typically several million interactions to a few thousands or hundreds.

2 Once Again Recalibrating the Research Focus

This situation now allows, or even demands, the recalibration of the research focus, shifting it towards problems that have already been on the agenda in the early days of reinforcement learning, but somehow nearly fell into oblivion during the time of more pressing, more fundamental research on RL's stability and scalability. Among these questions are:

- the question about how and where to explore the environment. For maximizing the learning results, while at the same time reducing the invested effort in form of interactions, it's necessary to actively control the exploration in a goal-directed mannor.
- the question about the state and where its particular representation comes from. The design of the state space often involves important insights into the system and in many cases heavily influences the task's complexity and the final results.
- the question about how an agent could autonomously break down a given task into several smaller sub-tasks, define valid sub-goals and learn to solve them sequentially. This is one of the questions treated in hierarchical reinforcement learning—but a real break-through is still missing.
- the related but more philosophical question about the origins of reward and intrinsic motivations of an agent. How can an agent act and learn self-motivated?

3 A Chance for Rebuilding the Cognitive Agent

We strongly believe future systems have to address these questions in order to achieve further performance enhancements, to allow the application to more complex problems, and to allow an improved autonomy of (reinforcement) learning agents. Hence, it seems to be the right time to think about more complex architectures of learning agents, building upon the basic building blocks of stable and efficient RL methods, and introducing higher levels of modules for monitoring, directing and controlling the process and progress of learning. The goal would be to (re-)introduce reinforcement learning into a broader context within artificial intelligence and the cognitive agent. In this respect, we do not expect much less than a small revolution, at its end having symbolic methods and sub-symbolic methods—to which we count reinforcement learning—no longer opposed to each other, but working together hand-in-hand, solving different sub-tasks on different levels of the architecture. We clearly see reinforcement learning and methods from neural computation at the lower levels of such an architecture, having more deliberative and perhaps symbolic methods at its higher levels. From the area of neural computation, we expect important contributions in two different areas: 1. within the "core" of reinforcement learning, the approximation of value functions and the representation of policies when learning on continuous, real systems and 2. within the outlined cognitive architecture and its supporting modules.

We would like to briefly discuss three examples in the following:

Multi-layer perceptrons for approximating value functions Even in the batch approach to reinforcement learning multi-layer perceptrons and other "non-averagers" [7] do not guarantee a stable learning process in the general case [8]. Nevertheless, when using mulit-layer perceptrons, the batch approach with its separation of dynamic programming and function approximation and its synchronous updates of the approximated value function offers a significant improvement over older "direct" (online) methods with asynchronous updates. This and the benefit of using a global approximator are some of the reasons for Neural Fitted Q-Iteration (NFQ) already having scored several impressive successes on real systems, and for NFQ—besides FQI [9] and LSPI [16]—presently becoming an accepted standard method that people discuss, use and compare against [17–19]. New theoretical approaches [18] moreover promise a better understanding of the conditions and circumstances where particular non-averagers like MLPs might be save to use or better should be avoided.

Winner-take-all networks for adapting grids At the border of 1. and 2. is an application of winner-take-all networks (WTA) to the adaptation of the structure of grid approximators. All kinds of constant and interpolating grid approximators are still widely applied in RL, due to their conceptual simplicity and their provable stability within batch methods [8, 9]. WTA methods allow the automatic adaptation of the structure of such grid approximators to the distribution of the relevant data in the the state space that is actually unknown before starting the learning process [20]. This automatic adaptation is especially helpful, if the data is situated on a low-dimensional mannifold in a high-dimensional state space [20]. At the same time those methods could help bridge the gap towards higher-level modules of the cognitive agent architecture. Methods like vector quantization and especially self-organizing maps and neural gas could help detecting clusters and structure in the data and thus could allow to build a basis for a transition to deliberation and symbolic reasoning about the structure of the problem, for example in order to control exploration or to formulate new hypothesis on valid sub-goals.

Deep auto-encoders for learning feature spaces One important thing for a success of RL is to contruct a suitable state space that carries the necessary information (has the markov property), but is not overly complex. When using basic online-RL methods, this usually had to be done before starting the learning procedure, since changing the state representation afterwards had roughly the same effect as changing the structure of the function approximator (moving a support or basis function): all information learned about the value function so far was lost in the worst case [20]. But batch RL techniques allow for an immediate recalculation of the value functions in the changed state space and thus make "seamlessly" switching the state space during the learning process possible, without increasing the task's complexity in form of necessary interactions with the system [20]. Hence, the agent can deliberate over an appropriate state representation by itself and can adapt it during the learning process as thought necessary. This possibility already has been implemented in practice. The DFQ algorithm [21] uses deep auto-encoder

neural networks [22, 23] for unsupervised learning of low-dimensional feature spaces—these are used as basis for learning a value function—from high-dimensional observations (images), as observed during learning to control real systems based on visual data only [20].

One final thing to note is, from now on, all newly developed solutions have to consider the needs of real systems. If we have learned one thing from the past, it is that we can not develop fancy solutions for the discrete case and then hope these will somehow scale to stochastic problems with continuous state and action spaces. This already has failed once, even for the most simple algorithms, not only for complexity reasons, but also for the fundamental difference of problems when function approximation and stochasticity are involved. Thus, it's now time to once again rebuild the cognitive agent, but this time starting from the bottom, building on practice-proven algorithms, first addressing the fundamental questions and slowly moving towards the top.

4 Conclusion

There are many interesting challenges for neural methods in RL. We see many opportunities for such methods directly at the heart of reinforcement learning as well as in key areas of a more general, cognitive agent architecture. The newly proposed techniques for training deep neural architectures already have led to a notably increased interest in neural methods and are also of great importance in the context of RL. Winner-take-all networks can help to adapt the structure of function approximators to the distribution of the data in a very natural way and may play a role in bridging the gap to symbolic and deliberative modules in the higher levels of a cognitive agent architecture. Eventually, autonomous aerial and ground vehicles and especially autonomous robots will find entrance into our every-day live. In such systems that ultimately should be able to operate in our environment and to interact with us autonomously, modules using reinforcement learning and neural computation will almost certainly realize important aspects.

References

1. L. Baird. Residual algorithms: Reinforcement learning with function approximation. In *Proc. of the 12th International Conference on Machine Learning*, pages 30–37, 1995.
2. G.J. Gordon. Chattering in SARSA (λ). Technical report, 1996.
3. G. Tesauro and T. Sejnowski. A Parallel Network that Learns to Play Backgammon. *Artificial Intelligence*, 39(3):357–390, 1989.
4. R. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In *Advances in Neural Information Processing Systems 12 (NIPS 1999)*, pages 1057–1063, 2000.
5. A. Ng, A. Coates, M. Diel, V. Ganapathi, J. Schulte, B. Tse, E. Berger, and E. Liang. Autonomous Inverted Helicopter Flight via Reinforcement Learning. In *Experimental Robotics IX, The 9th International Symposium on Experimental Robotics (ISER)*, pages 363–372, 2004.

6. J. Peters and S. Schaal. Policy Gradient Methods for Robotics. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2006.
7. G. Gordon. Stable Function Approximation in Dynamic Programming. In *Proc. of the 12th ICML*, pages 261–268, 1995.
8. D. Ormoneit and Ś. Sen. Kernel-based reinforcement learning. *Machine Learning*, 49(2):161–178, 2002.
9. D. Ernst, P. Geurts, and L. Wehenkel. Tree-Based Batch Mode Reinforcement Learning. *Journal of Machine Learning Research*, 6(1):503–556, 2006.
10. M. Riedmiller. Neural Fitted Q Iteration – First Experiences with a Data Efficient Neural Reinforcement Learning Method. In *ECML 2005*. Springer, 2005.
11. L. Wehenkel, M. Glavic, and D. Ernst. New Developments in the Application of Automatic Learning to Power System Control. In *Proc. of the 15th Power Systems Computation Conference (PSCC05)*, 2005.
12. M. Riedmiller, M. Montemerlo, and H. Dahlkamp. Learning to Drive in 20 Minutes. In *Proc. of the FBIT 2007*, Jeju, Korea, 2007.
13. M. Riedmiller, T. Gabel, R. Hafner, and S. Lange. Reinforcement learning for robot soccer. *Autonomous Robots*, 27(1):55–73, 2009.
14. J. Peters and S. Schaal. Learning to Control in Operational Space. *The International Journal of Robotics Research*, 27(2):197–212, 2008.
15. M. Deisenroth, J. Peters, and C. Rasmussen. Approximate Dynamic Programming with Gaussian Processes. In *Proceedings of the 2008 American Control Conference (ACC 2008)*, pages 4480—4485, Seattle, USA, 2008. IEEE Press.
16. M. Lagoudakis and R. Parr. Least-Squares Policy Iteration. *Journal of Machine Learning Research*, 4:1107–1149, 2003.
17. D. Schneegaß, S. Udluft, and T. Martinetz. Improving optimality of neural rewards regression for data-efficient batch near-optimal policy identification. *Artificial Neural Networks–ICANN 2007*, pages 109–118, 2007.
18. A. Antos, R. Munos, and C. Szepesvari. Fitted Q-iteration in continuous action-space MDPs. *Advances in neural information processing systems*, 20:9–16, 2008.
19. S. Whiteson and P. Stone. Evolutionary function approximation for reinforcement learning. *The Journal of Machine Learning Research*, 7:917, 2006.
20. S. Lange. Tiefes Reinforcement Lernen auf Basis visueller Wahrnehmungen. Dissertation, Universität Osnabrück, 2010.
21. S. Lange and M. Riedmiller. Deep auto-encoder neural networks in reinforcement learning. In *International Joint Conference on Neural Networks (IJCNN 2010)*, Barcelona, Spain, 2010.
22. G.E. Hinton and R.R. Salakhutdinov. Reducing the Dimensionality of Data with Neural Networks. *Science*, 313(5786):504–507, 2006.
23. Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, and Q. Montreal. Greedy Layer-Wise Training of Deep Networks. In *Proc. of the NIPS 2006*. MIT Press, 2007.

Challenges in Training Restricted Boltzmann Machines

Asja Fischer and Christian Igel*

Institut für Neuroinformatik
Ruhr-Universität Bochum, 44780 Bochum, Germany
asja.fischer@ini.rub.de,christian.igel@ini.rub.de

Abstract. Optimization based on k-step Contrastive Divergence (CD) is promising for training Restricted Boltzmann Machines and has been successfully applied in practice. However, CD-learning is based on a biased approximation of the log-likelihood-gradient and this bias can lead to serious problems. Here, we review recent theoretical and empirical studies revealing and analyzing this issue. Among other things, it has been claimed that – despite of the bias – the signs of most components of the CD update direction are equal to the corresponding signs of the gradient of the log-likelihood. This suggests combining CD with optimization algorithms just depending on the signs of the components of the gradient of the objective function. However, we empirically show that combining CD and Rprop does not solve divergence problems occurring in CD learning.

Key words: Unsupervised Learning, Restricted Boltzmann Machines, Contrastive Divergence, Gibbs Sampling

1 Introduction

Understanding and modeling how brains learn higher-level representations from sensory input is one of the key challenges in computational neuroscience and machine learning. Using multiple levels of more and more abstract representations is a characteristic property of the brain leading to the ability to abstract and to generalize. This idea is picked up by the machine learning research field of "deep learning", which refers to learning layers of representations.

Layered generative models such as deep belief networks (DBNs) are promising for learning such representations, and new algorithms that operate in a layer-wise fashion make learning these models computationally tractable [4–8]. Restricted Boltzmann Machines (RBMs) are the typical building blocks for DBN layers. They are undirected graphical models (Markov random fields), and their structure is a bipartite graph connecting visible and hidden neurons. Training large undirected graphical models by likelihood maximization in general involves averages over an exponential number of terms, and obtaining unbiased estimates

* This paper is based on [1–3].

of these averages by Markov chain Monte Carlo methods typically requires too many sampling steps. However, recently it was shown that estimates obtained after running the chain for just a few steps can be sufficient for model training [4]. In particular, *Contrastive Divergence* (CD, [4]) learning has become a standard way to train RBMs. [4–8]. The k-step CD learning is based on steepest-ascent on an estimate of the log-likelihood gradient gained by k steps of Gibbs sampling. The CD estimate is a biased approximation of the true gradient. Therefore, CD-learning does not necessarily reach the maximum likelihood estimate of the parameters. The bias of the approximation depends on the mixing rate of the Markov chain, and mixing slows down with increasing absolute value of the model parameters [4, 9, 7]. Hence, the bias increases with increasing parameter magnitude during RBM training. Recently it has been shown that this can lead to divergence of the log-likelihood. This has also been observed for the refined methods *Persistent Contrastive Divergence* (PCD, [10]) and *Fast Persistent Contrastive Divergence* (FPCD, [11]) [1, 12, 3]. Nevertheless, Bengio and Delalleau [7] found by comparing the log-likelihood gradient and the average of CD-k in small RBMs (where both are tractable) that the fraction of parameter updates for which the log-likelihood gradient and the average over CD-k have different signs remains small. This raises the question whether optimization algorithms which consider only the signs of the partial derivatives, such as Resilient Backpropagation (Rprop, [13, 14]), could lead to better learning results. This is empirically investigated in this paper. After briefly describing CD, PCD, and FPCD leaning and their limitations, we describe our experiments, discuss the results, and finally draw our conclusions.

2 RBMs

An RBM is an undirected graphical model [4, 15]. Its structure is a bipartite graph consisting of one layer of m visible units $\boldsymbol{V} = (V_1, \ldots, V_m)$ representing observable data and one layer of n hidden units $\boldsymbol{H} = (H_1, \ldots, H_n)$ capturing dependencies between observed variables. It is parametrized by the connection weights w_{ij} as well as the biases b_j and c_i of visible and hidden units, respectively ($i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$). Given these parameters, jointly denoted as $\boldsymbol{\theta}$, the joint distribution of $\boldsymbol{V}$ and $\boldsymbol{H}$ under the model is

$$p(\boldsymbol{v}, \boldsymbol{h}) = e^{-\mathcal{E}(\boldsymbol{v},\boldsymbol{h})}/Z \ , \tag{1}$$

where $Z = \sum_{\boldsymbol{v},\boldsymbol{h}} e^{-\mathcal{E}(\boldsymbol{v},\boldsymbol{h})}$ is a normalization constant and the energy $\mathcal{E}$ is given by

$$\mathcal{E}(\boldsymbol{v}, \boldsymbol{h}) = -\sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij} h_i v_j - \sum_{j=1}^{m} b_j v_j - \sum_{i=1}^{n} c_i h_i \ . \tag{2}$$

The log-likelihood of the model parameters $\boldsymbol{\theta}$ given one training example $\boldsymbol{v}_l$ is

$$\log L(\boldsymbol{\theta}|\boldsymbol{v}_l) = \log P(V = \boldsymbol{v}_l) = \log \sum_{\boldsymbol{h}} e^{-\mathcal{E}(\boldsymbol{v}_l,\boldsymbol{h})} - \log \sum_{\boldsymbol{v},\boldsymbol{h}} e^{-\mathcal{E}(\boldsymbol{v},\boldsymbol{h})} \ . \tag{3}$$

Differentiating the above with respect to $\boldsymbol{\theta}$ yields

$$\frac{\partial}{\partial \boldsymbol{\theta}} L(\boldsymbol{\theta}|\boldsymbol{v}_l) = -\sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{v}_l) \frac{\partial \mathcal{E}(\boldsymbol{v}_l, \boldsymbol{h})}{\partial \boldsymbol{\theta}} + \sum_{\boldsymbol{v}} p(\boldsymbol{v}) \sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{v}) \frac{\partial \mathcal{E}(\boldsymbol{v}, \boldsymbol{h})}{\partial \boldsymbol{\theta}} \quad . \tag{4}$$

Computing the first term on the right side of the equation is straightforward because it factorizes. For example, for a weight w_{ij} it reduces to

$$-\sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{v}_l) \frac{\partial \mathcal{E}(\boldsymbol{v}_l, \boldsymbol{h})}{\partial w_{ij}} = E_{p(h_i|\boldsymbol{v}_l)}[h_i] v_{lj} \quad , \tag{5}$$

where v_{lj} denotes the j-th component of $\boldsymbol{v}_l$. The computation of the second term is intractable for regular sized RBMs because its complexity is exponential in the size of the smallest layer. However, the expectation over $p(\boldsymbol{v})$ can be approximated by alternating Gibbs sampling [16, 17]. But since the sampling chain needs to be long to get almost unbiased samples of the distribution modeled by the RBM, the computational effort is still too large.

3 Training methods based on Gibbs sampling

The basic idea of the k-step Contrastive Divergence (CD-k) algorithm [4] is as follows. Instead of running the Gibbs chain until a near-to-equilibrium distribution is reached, the chain is run for only k steps starting from a training example $\boldsymbol{v}^{(0)}$ producing the sample $\boldsymbol{v}^{(k)}$. Each step t consists of sampling $\boldsymbol{h}^{(t)}$ from $p(\boldsymbol{h}|\boldsymbol{v}^{(t)})$ and sampling $\boldsymbol{v}^{(t+1)}$ from $p(\boldsymbol{v}|\boldsymbol{h}^{(t)})$ subsequently. The gradient (4) with respect to $\boldsymbol{\theta}$ of the log-likelihood for one training pattern $\boldsymbol{v}^{(0)}$ is then approximated by

$$\mathrm{CD}_k(\boldsymbol{\theta}, \boldsymbol{v}^{(0)}) = -\sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{v}^{(0)}) \frac{\partial \mathcal{E}(\boldsymbol{v}^{(0)}, \boldsymbol{h})}{\partial \boldsymbol{\theta}} + \sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{v}^{(k)}) \frac{\partial \mathcal{E}(\boldsymbol{v}^{(k)}, \boldsymbol{h})}{\partial \boldsymbol{\theta}} \quad . \tag{6}$$

Based on this basic idea of approximating the log-likelihood via Gibbs sampling, refinements of vanilla CD learning have been proposed more recently [10, 11]. In the *Persistent Contrastive Divergence* (PCD, [10]) algorithm, $\boldsymbol{v}^{(k)}$ in (6) is sampled from a Markov chain defined by the RBM parameters that is independent of $\boldsymbol{v}^{(0)}$. This corresponds to standard CD learning without reinitializing the visible units of the Markov chain with the current training sample. It is assumed that the chain stays close to the stationary distribution if the learning rate is sufficiently small and thus the model changes only slightly between parameter updates [18, 10]. In *Fast Persistent Contrastive Divergence* (FPCD, [11]), a set of additional parameters is introduced, which are only used for Gibbs sampling. These parameters are referred to as *fast* parameters and should lead to higher mixing rates. When calculating the conditional distributions for Gibbs sampling, the regular parameters are replaced by the sum of the regular and the fast parameters. The update rule for the fast parameters is equal to that of the regular parameters, but with an independent, large learning rate and a

large weight-decay parameter. For details about PCD and FPCD we refer to the original publications [10] and [11], respectively.

Desjardins et al. as well as Salakhutdinov [12, 19] showed that using Markov Chain Monte Carlo algorithms based on tempered transitions yields better training results than Gibbs sampling. But up until now it is not clear if these algorithms can compete with the Gibbs sampling based algorithms when they are used in real world scenarios with limited computation time.

4 Limitations of the proposed learning algorithms

In the following, we discuss some theoretical properties of the proposed RBM learning algorithms as well as selected empirical studies revealing problems when using these methods for RBM training.

4.1 Theoretical considerations

Training RBMs using CD does not necessarily lead to a maximum likelihood estimate of the model parameters. Examples of energy functions and Markov chains for which CD-1 learning does not converge are given in [20]. Yuille [21] specifies conditions under which CD learning is guaranteed to converge to the maximum likelihood solution, which need not hold for RBM training in general.

Bengio and Delalleau [7] show that CD-k is an approximation of the true log-likelihood gradient by finding an expansion of the gradient that considers the k-th sample in the Gibbs chain and showing that CD-k is equal to a truncation of this expansion. Furthermore, they prove that the residual term (i.e., the bias of CD) converges to zero as k goes to infinity. Bengio and Delalleau [7] additionally prove a bound for the bias of CD, which reflects the dependence on the magnitude of the parameters and the value of k. However, this bound is very loose (it approximates the number of configurations of the visible units, while the absolute value of the bias is never larger than one.)

Fischer and Igel [2] recently derived a tighter upper bound based on general results for the periodic Gibbs Sampler [22]. Its magnitude depends on k, on the number of variables in the RBM, and on the maximum change in energy that can be produced by changing a single variable. The latter reflects the dependence on the absolute values of the RBM parameters. The bound emphasizes that the magnitude of the bias is also affected by the distance in variation between the modeled distribution and the starting distribution of the chain.

4.2 Empirical studies

Experiments comparing the quality of small RBMs trained based on the expectation of CD-1 (to avoid sampling noise) and true likelihood maximization are presented in [9] and [7]. Carreira-Perpiñán and Hinton [9] show that in general CD learning does not lead to the maximum likelihood solutions. In their experiments is reaches solutions close by. Bengio and Delalleau [7] show that the

quality of CD-k as an approximation of the log-likelihood gradient decreases as the norm of the parameters increases. Anyhow, the RBMs are still able to model the considered simple target distributions. Additionally, they find that the bias of CD-k also increases with increasing number of visible units, but that the fraction of parameter updates for which the log-likelihood gradient and the average over CD-k have different signs remains small.

However, there is empirical evidence that the bias of CD can also have serious effects: Fischer and Igel [1] show that CD can even lead to a steady decrease of the log-likelihood during learning. This is confirmed in [12, 3] also for PCD and FPCD. The divergence is a serious problem in practice because there does not exist any efficiently computable stopping criteria indicating the decrease of the log-likelihood. The results of Fischer and Igel [3] indicate that the log-likelihood seems to diverge especially if the target distribution is difficult to learn for an RBM. Furthermore, they show that the decrease of the likelihood can not be detected by an increase of the reconstruction error, which has been proposed as a stopping criterion for CD learning. Weight-decay with a carefully chosen weight-decay-parameter can prevent divergence.

5 Training RBMs with Resilient Backpropagation

In our experiments, we study the evolution of the log-likelihood during gradient-based training of RBMs using the Resilient Backpropagation algorithm (Rprop) based on CD-k. We first give a description of the Rprop algorithm, then briefly describe our benchmark problems, give details of the experimental setup, and finally describe and discuss the results of our experiments.

5.1 Resilient Backpropagation

The speed of steepest-ascent CD learning crucially depends on the learning rate (see for example the empirical results in [3]). For large-scale problems, optimization algorithms are needed that increase the likelihood in few iterations without extensive hyperparameter tuning. However, as the likelihood is in general intractable the choice of the gradient-based optimization algorithm is limited to methods that do not need the absolute objective function value. Resilient Backpropagation is such a method. It is an iterative algorithm with adaptive individual step sizes [13]. It is frequently used for unconstrained optimization in machine learning because it is fast; robust with respect to the choice of the internal (hyper-) parameters; has linear time and space complexity in the number of parameters to be optimized; and is not very sensitive to numerical problems. The Rprop algorithm considers only the signs of the partial derivatives of the function to be optimized and not their values. Because experiments suggest that the CD estimator has the correct sign most of the time [7], Rprop seems to be promising for CD learning.

In each iteration g of Rprop, every parameter $\theta_i^{(g)}$ is updated according to

$$\theta_i^{(g+1)} = \theta_i^{(g)} + \operatorname{sign}\left(\left[\overline{\mathrm{CD}}_k(\boldsymbol{\theta}^{(g)})\right]_i\right) \cdot \Delta_i^{(g)} \quad . \tag{7}$$

Prior to this update, the step size $\Delta_i^{(g)}$ is adapted based on changes of sign of the (approximated) partial derivative $[\overline{\mathrm{CD}}_k(\boldsymbol{\theta}^{(g)})]_i$ in consecutive iterations. If the sign changes, indicating that a local minimum has been overstepped, then the step size is multiplicatively decreased, otherwise, it is increased. The update rule for the step size is given by:

$$\Delta_i^{(g)} = \begin{cases} \min(\eta^+ \Delta_i^{(g-1)}, \Delta_{\max}) & \text{if } [\overline{\mathrm{CD}}_k(\boldsymbol{\theta}^{(g-1)})]_i [\overline{\mathrm{CD}}_k(\boldsymbol{\theta}^{(g)})]_i > 0 \\ \max(\eta^- \Delta_i^{(g-1)}, \Delta_{\min}) & \text{if } [\overline{\mathrm{CD}}_k(\boldsymbol{\theta}^{(g-1)})]_i [\overline{\mathrm{CD}}_k(\boldsymbol{\theta}^{(g)})]_i < 0 \\ \Delta_i^{(g-1)} & \text{else} \ , \end{cases} \tag{8}$$

where $0 < \eta^- < 1 < \eta^+$ and the step size is bounded by $\Delta_{\min}$ and $\Delta_{\max}$. The hyperparameters η^-, η^+, $\Delta_{\min}$ and $\Delta_{\max}$ can be fixed to default values.

5.2 Benchmark problems

We consider four artificial benchmark problems taken from the literature. The first two problems are described in [17, 23] and are also used in [3]. The other two are taken from [7].

Labeled Shifter Ensemble. The 19 dimensional data set contains 768 samples, 3 of which shown in Figure 1. The samples are generated in the following way: The states of the first 8 visible units are set uniformly at random. The states of the following 8 units are cyclically shifted copies of the first 8. The shift can be zero, one unit to the left, or one to the right and is indicated by the last three units. The log-likelihood is $768 \log \frac{1}{768} \approx -5102.43$ if the distribution of the data set is modeled perfectly.

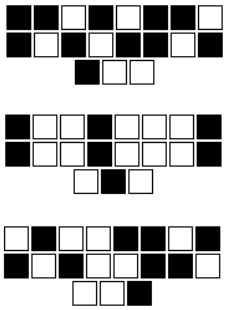


Fig. 1. Three sample patterns of the Shifter benchmark problem. The first input row is shifted to the left, not shifted, and shifted to the right, respectively, as indicated by the three label units.

Bars and Stripes Ensemble. We consider a smaller variant of the Bars-and-Stripes problem described in [23] with 16 instead of 25 visible units. Examples of input patterns are shown in Fig. 2. Each pattern corresponds to a square of 4×4 units and is generated by first randomly choosing an orientation, vertical or horizontal with equal probability, and then picking the state for all units of every row or column uniformly at random. Since each of the two completely uniform patterns can be generated in two ways, the lower bound of the log-likelihood is -102.59.

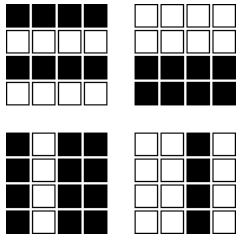


Fig. 2. Four patterns of the (simplified) Bars-and-Stripes benchmark problem.

Diag. The $Diag_d$-Problem is a d-dimensional data set containing $d+1$ binary vectors. In this study we focus on $d = 6$, which is the smallest dimension used in the experiments of [7]. The $Diag_6$ data set is given by:

$$
\begin{aligned}
&(0,0,0,0,0,0)\\
&(1,0,0,0,0,0)\\
&(1,1,0,0,0,0)\\
&(1,1,1,0,0,0)\\
&(1,1,1,1,0,0)\\
&(1,1,1,1,1,0)\\
&(1,1,1,1,1,1)
\end{aligned}
$$

An upper bound for the log-likelihood is given by -13.62.

ID-Ball. The $IDBall_d$ data set consists of $2d\lfloor\frac{d-1}{2}\rfloor$ binary vectors, which are generated as described in [7]. Again, we focus only on $d = 6$, for which the

training set is given by:

$(1,0,0,0,0,0)$	$(0,1,1,1,1,1)$
$(1,1,0,0,0,0)$	$(0,0,1,1,1,1)$
$(0,1,0,0,0,0)$	$(1,0,1,1,1,1)$
$(0,1,1,0,0,0)$	$(1,0,0,1,1,1)$
$(0,0,1,0,0,0)$	$(1,1,0,1,1,1)$
$(0,0,1,1,0,0)$	$(1,1,0,0,1,1)$
$(0,0,0,1,0,0)$	$(1,1,1,0,1,1)$
$(0,0,0,1,1,0)$	$(1,1,1,0,0,1)$
$(0,0,0,0,1,0)$	$(1,1,1,1,0,1)$
$(0,0,0,0,1,1)$	$(1,1,1,1,0,0)$
$(0,0,0,0,0,1)$	$(1,1,1,1,1,0)$
$(1,0,0,0,0,1)$	$(0,1,1,1,1,0)$

As $IDBall_6$ consists out of 24 vectors generated with equal probability the upper bound for the log-likelihood is given by -76.27.

5.3 Experimental Setup

The RBMs were initialized with weights drawn uniformly from $[-0.5, 0.5]$ and zero biases. The numbers of hidden units were chosen to be equal to the number of the visible units.

The models were trained with Rprop based on CD-1 or CD-100 on all four benchmark problems. If not stated otherwise, the hyperparameters where set to the default values $\eta^- = 0.5$, $\eta^+ = 1.1$, $\Delta_{\min} = 0.0$ and $\Delta_{\max} = 10^{100}$.

To save computation time, the exact likelihood was calculated only every 10 iterations of the learning algorithm. All experiments were repeated 25 times.

5.4 Results

Figure 3 shows (in the top left plot) the evolution of the log-likelihood during learning of the Shifter-problem with Rprop based on CD-1. Shown are the medians over 25 trails with different parameter initializations. After an increase in the first iterations the log-likelihood starts to decrease. The development of the likelihood differs a lot depending on the parameter initialization. This can be seen exemplarily in the top right plot of Fig. 3 depicting some single trials. When using the CD-100 instead of the CD-1 approximation of the gradient a stagnation of the log-likelihood on an unsatisfying level during Rprop based learning is observed (see bottom plot of Fig. 3). This happens systematically in every trail independent of the initialization of the parameters as indicated by the quartiles.

During training an RBM on the Bars-and-Stripes-problem the log-likelihood stagnates when Rprop is based on CD-1 as well as on CD-100. In both settings similar log-likelihood values are reached which are low compared to the

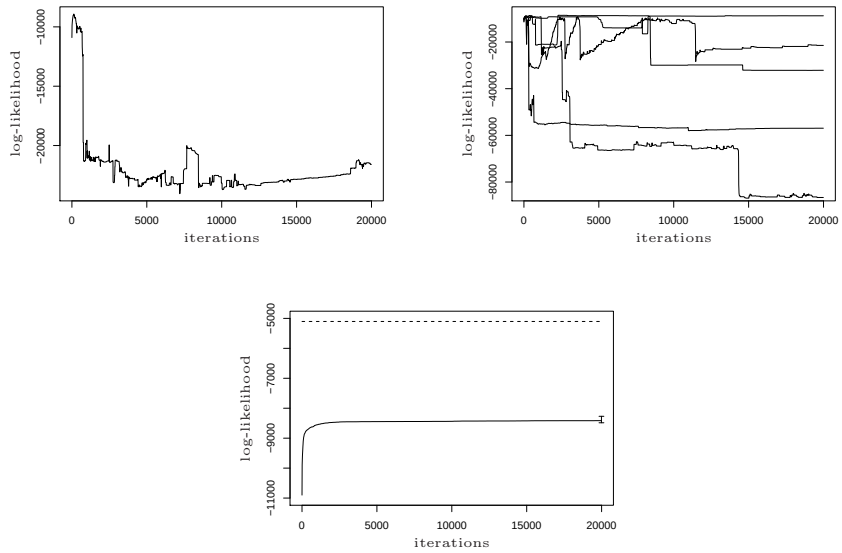


Fig. 3. Top: The development of the log-likelihood during training an RBM on the Shifter-problem with Rprop based on CD-1. The medians over 25 trails are shown in the top left plot. Five single trails with different parameter initializations are exemplarily shown on the right. Bottom: Training with Rprop based on CD-100. Shown are the medians over 25 trails, error bars indicate quartiles, the dashed line indicates the upper bound of the log-likelihood.

upper bound and to the maximum values reached when an RBM is trained with steepest descent (see empirical analysis of [3]).

As shown in Fig. 5 and Fig. 6, the log-likelihood also stagnates when learning the Diag- and IDBall-problem. Here we also observe similar learning curves if the Rprop algorithm is based on CD-1 and CD-100.

The stagnation of the log-likelihood could indicate a frequently changing sign of the CD-approximation during the learning process. A frequently changing sign of the approximation causes the step size for the parameter updates (see (8)) to get smaller and smaller and – if the step size is not bounded – to finally approach zero. Thus it could be possible to avoid the stagnation by enlarging the minimal possible step size $\Delta_{\min}$. This idea is verified by the following results.

If the minimal step size $\Delta_{\min}$ is set to a value larger than zero and the maximal step size $\Delta_{\max}$ is set to a value smaller than the default value, we can observe big differences in the evolution of the log-likelihood during Rprop based training. As shown in Fig. 7, high (relative to the upper bound) log-likelihood values are reached during learning the Diag- and the IDBall-problem with Rprop based on CD-1 if the hyperparameters are set to $\Delta_{\min} = 0.0001$ or $\Delta_{\min} = 0.001$, respectively, and $\Delta_{\max} = 1$. A nearly identical evolution of the log-likelihood can

be observed (results not shown) if only the minimal step size value is enlarged and $\Delta_{\max}$ is set to its default value.

When learning the Bars-and-Stripes-problem with a restriction of the step size parameters ($\Delta_{\min} = 0.0001$ and $\Delta_{\max} = 1$) the log-likelihood starts to diverges (see top plots in Fig. 8). The experiments with the Shifter-problem with restricted step size parameters lead to similar results as the experiments with the parameters set to the default values (see bottom plots in Fig. 8).

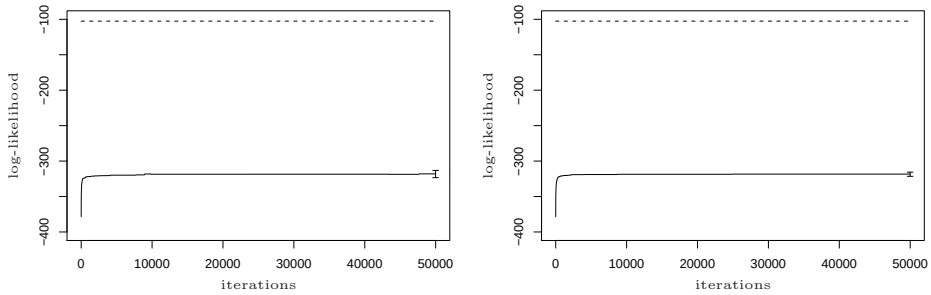


Fig. 4. Evolution of the log-likelihood during CD-learning of the Bars-and-Stripes-problem based on Rprop. Training is based on CD-1 (shown left) or CD-100 (shown right) respectively.

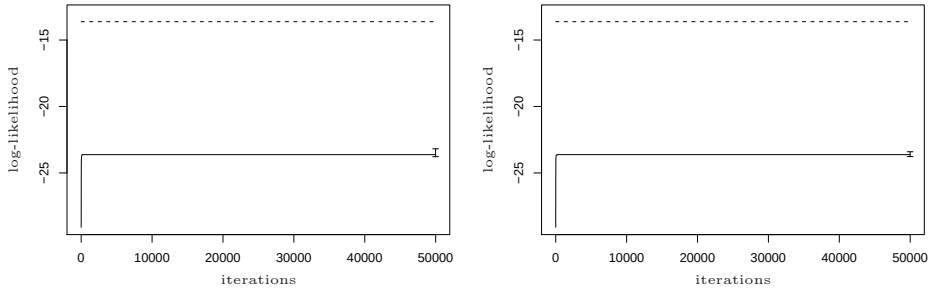


Fig. 5. Log-likelihood during learning of the *Diag*-problem based on CD-1(shown on the left) or CD-100 (shown on the right).

5.5 Discussion

The experiments show that the success of Rprop based CD learning depends on the data distribution to be learned and on the values of the hyperparameters ($\Delta_{\min}$ and $\Delta_{\max}$). If the step size is allowed to get arbitrary close to zero ($\Delta_{\min} = 0.0$), the training progress stagnated on an unsatisfying level for some target

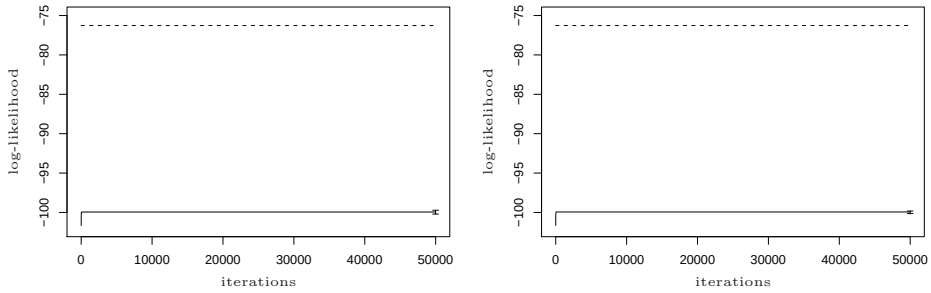


Fig. 6. Log-likelihood for Rprop based on k-step-CD applied to the *IDBall*-problem. Left: Rprop based on CD-1. Right: Rprop based on CD-100.

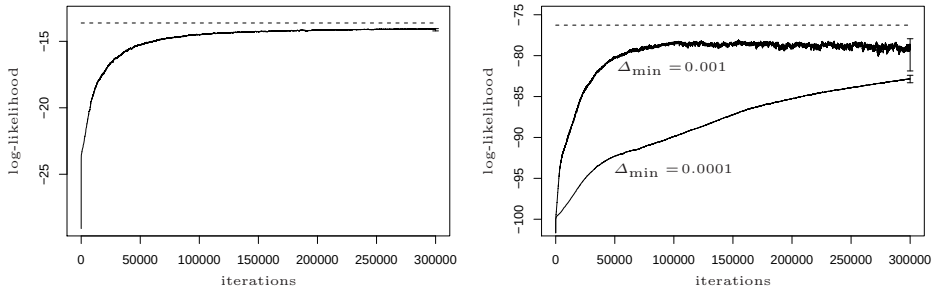


Fig. 7. Log-likelihood during training with Rprop with limited step size hyperparameters. On the left: Learning of the *Diag*-problem. The step size is limited by $\Delta_{\min} = 0.0001$ and $\Delta_{\max} = 1$. On the right: Learning of the *IDBall*-problem with $\Delta_{\max} = 1$ and $\Delta_{\min} = 0.001$ and $\Delta_{\min} = 0.0001$ respectively.

distribution. With an appropriate $\Delta_{\min}$, Rprop was able to learn good models depending on the problem.

The reason for the stagnation could be convergence to a suboptimal local maximum. However, experiments using the expectation of CD-1 (not shown) did not suffer from the stagnation problem. As we see no reason why learning based on the expectation of CD-1 should be less prone to getting stuck in undesired local optima than learning based on the CD approximation, local maxima are not likely to be the reason.

We believe that the reason is the fast reduction of the Rprop step size parameters Δ_i due to changes in sign of the gradient components due to stochastic effects and errors in the CD approximation.

If steepest-ascent can learn a distribution, this is also possible using Rprop, but in our experiments this required $\Delta_{\min} > 0$. When applying Rprop to Shifter and Bars-and-Stripes, the log-likelihood diverged before a good model was learned even if we constrained $\Delta_{\min}$ and $\Delta_{\max}$. That is, if learning diverges using steepest-ascent (as reported in [3]), it also diverged using Rprop. Thus, albeit it has been

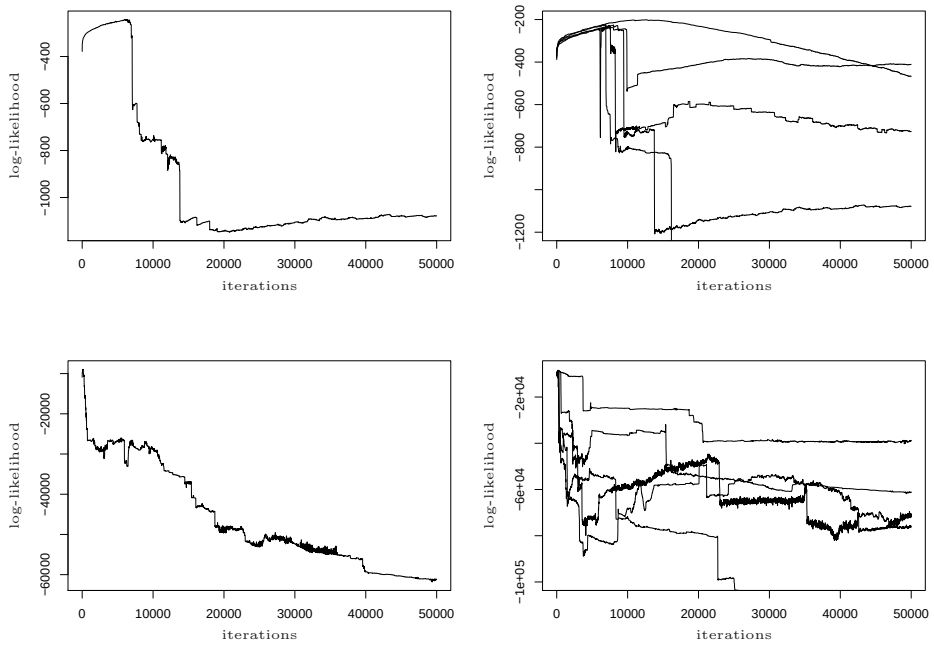


Fig. 8. Log-likelihood during training with Rprop with limited step size hyperparameters ($\Delta_{\min} = 0.0001$ and $\Delta_{\max} = 1$). Top: Results for the Shifter-problem. Bottom: Results for the Bars-and-Stripes-problem. The medians over 25 trails are shown on the left. Five single trails with different parameter initializations are exemplarily shown on the right.

reported that the sign of the components of the CD update direction vector is often right, learning based on these signs tends to diverge.

6 Conclusion

Combining multiple layers of Restricted Boltzmann Machines (RBMs) is a promising approach to learning abstract representations. Despite the recent progress in the development of learning algorithms for RBMs, the problem of efficient, robust RBM learning is far from being solved. While learning using k-step Contrastive Divergence (CD) or (Fast) Persistent CD has been applied successfully, the CD is only a biased estimate of the desired update direction and thus CD learning may not converge the maximum likelihood solution. The learning process may even diverge in the sense that the model systematically gets worse. This happens if the absolute values of the model parameters – the RBM weights – get to large leading to a strong bias in the estimate of the log-likelihood gradient. Although it has been argued that the signs of the gradient components are estimated correctly most of the time, learning algorithms just considering

these signs – and not the absolute values – do not solve the divergence problem. Therefore, for training algorithms relying on Gibbs sampling based stochastic approximations of the log-likelihood gradient, there is a need for robust mechanisms that control the weight growth in CD and related learning algorithms, for example, reliable heuristics for choosing the weight decay parameters or suitable criteria for early-stopping. New learning methods for RBMs using Markov Chain Monte Carlo algorithms based on tempered transitions are promising [19, 12], but their learning and scaling behavior needs to be further explored.

Acknowledgements

We acknowledge support from the German Federal Ministry of Education and Research within the National Network Computational Neuroscience under grant number 01GQ0951.

References

1. Fischer, A., Igel, C.: Contrastive divergence learning may diverge when training restricted Boltzmann machines. Frontiers in Computational Neuroscience. Bernstein Conference on Computational Neuroscience (BCCN 2009) (2009)
2. Fischer, A., Igel, C.: Bounding the bias of contrastive divergence used for maximum likelihood learning in Restricted Boltzmann Machines. In: Second Joint Statistical Meeting Deutsche Arbeitsgemeinschaft Statistik "Statistics Under One Umbrella" (DAGStat 2010), Universität Dortmund (2010) 98
3. Fischer, A., Igel, C.: Empirical analysis of the divergence of Gibbs sampling based learning algorithms for Restricted Boltzmann Machines. In: International Conference on Artificial Neural Networks (ICANN 2010). LNCS, Springer-Verlag (2010)
4. Hinton, G.E.: Training products of experts by minimizing contrastive divergence. Neural Computation **14** (2002) 1771–1800
5. Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H., Montreal, U.: Greedy layer-wise training of deep networks. In Schölkopf, B., Platt, J., Hoffman, T., eds.: Advances in Neural Information Processing (NIPS 19), MIT Press (2007) 153–160
6. Hinton, G.E., Osindero, S., Teh, Y.W.: A fast learning algorithm for deep belief nets. Neural Computation **18**(7) (2006) 1527–1554
7. Bengio, Y., Delalleau, O.: Justifying and generalizing contrastive divergence. Neural Computation **21**(6) (2009) 1601–1621
8. Hinton, G.E.: Learning multiple layers of representation. Trends in Cognitive Sciences **11**(10) (2007) 428–434
9. Carreira-Perpiñán, M.Á.., Hinton, G.E.: On contrastive divergence learning. In: 10th International Workshop on Artificial Intelligence and Statistics (AISTATS 2005). (2005) 59–66
10. Tieleman, T.: Training restricted Boltzmann machines using approximations to the likelihood gradient. In Cohen, W.W., McCallum, A., Roweis, S.T., eds.: International Conference on Machine learning (ICML), ACM (2008) 1064–1071
11. Tieleman, T., Hinton, G.E.: Using fast weights to improve persistent contrastive divergence. In Pohoreckyj Danyluk, A., Bottou, L., Littman, M.L., eds.: International Conference on Machine Learning (ICML), ACM (2009) 1033–1040

12. Desjardins, G., Courville, A., Bengio, Y., Vincent, P., Dellaleau, O.: Parallel tempering for training of restricted Boltzmann machines. Journal of Machine Learning Research Workshop and Conference Proceedings **9**(AISTATS 2010) (2010) 145–152
13. Riedmiller, M.: Advanced supervised learning in multi-layer perceptrons – From backpropagation to adaptive learning algorithms. Computer Standards and Interfaces **16**(5) (1994) 265–278
14. Igel, C., Hüsken, M.: Empirical evaluation of the improved Rprop learning algorithm. Neurocomputing **50**(C) (2003) 105–123
15. Smolensky, P.: Information processing in dynamical systems: Foundations of harmony theory. In Rumelhart, D.E., McClelland, J.L., eds.: Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol. 1: Foundations. MIT Press (1986) 194–281
16. Ackley, D.H., Hinton, G.E., Sejnowski, T.J.: A learning algorithm for Boltzmann machines. Cognitive Science **9** (1985) 147–169
17. Hinton, G.E., Sejnowski, T.J.: Learning and relearning in Boltzmann machines. In Rumelhart, D.E., McClelland, J.L., eds.: Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol. 1: Foundations. MIT Press (1986) 282–317
18. Younes, L.: Maximum likelihood estimation of gibbs fields. In Possolo, A., ed.: Proceedings of an AMS-IMS-SIAM Joint Conference on Spacial Statistics and Imaging. Lecture Notes Monograph Series. Institute of Mathematical Statistics, Hayward, California (1991)
19. Salakhutdinov, R.: Learning in markov random fields using tempered transitions. In Bengio, Y., Schuurmans, D., Lafferty, J., Williams, C.K.I., Culotta, A., eds.: Advances in Neural Information Processing Systems 22. (2009) 1598–1606
20. MacKay, D.J.C.: Failures of the one-step learning algorithm. Cavendish Laboratory, Madingley Road, Cambridge CB3 0HE, UK. http://www.cs.toronto.edu/ mackay/gbm.pdf (2001)
21. Yuille, A.L.: The convergence of contrastive divergence. In Saul, L., Weiss, Y., Bottou, L., eds.: Advances in Neural Processing Systems (NIPS 17), MIT Press (2005) 1593–1600
22. Brémaud, P.: Markov chains: Gibbs fields, Monte Carlo simulation, and queues. Springer-Verlag (1999)
23. MacKay, D.J.C.: Information Theory, Inference & Learning Algorithms. Cambridge University Press (2002)

Mental Imagery in Artificial Agents

Alexander Kaiser[1], Wolfram Schenck[1,2], and Ralf Möller[1,2]

[1] Computer Engineering Group, Faculty of Technology, Bielefeld University
[2] Center of Excellence Cognitive Interaction Technology (CITEC), Bielefeld University

Abstract. We present a model architecture for a basic form of mental imagery, based on an appearance based approach to view synthesis. This model architecture is developed for an experimental study in which an artificial agent learns how to associate the values of a set of posture variables and the corresponding view of its robotic gripper. Our model is fully adaptive and does not require any a priori information.

1 Introduction

Mental imagery is the process of generating internal sensory experiences and impressions of motor activity without actual sensory inflow. Humans are obviously capable of recalling sensory experiences even in a willful manner. These sensory experiences may relate to changes in visual scenes caused by the execution of covert motor commands (e.g. walking through an environment) or may just be a recall of previously perceived sensations. Neuroimaging studies suggest that cortical areas which are involved in the processing of perceived stimuli are also active during mental imaging [4]. A similar finding suggests that motor areas are used for executed as well as covert actions [3].

In this article we suggest an artificial neural architecture which enables an agent to generate mental views of parts of itself based on the values of a set of postural variables and motor commands. The agent might use these mental images to identify itself (or parts of itself) in its current view. This form of "*self-awareness*" becomes important when the agent starts interacting with objects or with other agents: here, the self-aware agent is able to discriminate portions of the visual input (e.g. pixels) that belong to the objects it interacts with from portions that belong to its own body.

In the following, we will consider a stationary agent which consists of an arm with an attached two-finger gripper and a stereo camera head. The mental imagery is restricted to views of the gripper based on (a) the arm posture and (b) the current gaze direction. Thus, we can model the generation of a mental image as a mapping from parameter into image space — a process commonly termed *view synthesis* in the literature [2].

The field of view synthesis can be divided into two main branches; *image synthesis* methods are based on models of the scene whereas *morphing* approaches use one or more images of the scene which are morphed into a new view. A

special case of the latter is *image warping* which transforms one input image based on a set of parameters [1].

Typically, image synthesis requires a precise geometrical model of the scene, e.g. a CAD model. While such a model can be acquired for artificial applications, it seems very unlikely that this form of representation is used in biological brains. Appearance-based view synthesis [2] does not require an explicit model; the model is rather "learned" directly from provided image data. The view (i.e. an image) is parameterized by an *appearance vector* whose dimensionality is much lower than the number of pixels in the original image. For our application, the objective is now to establish a mapping from postural parameters onto the appearance vector.

Image warping [1] is a general term referring to a class of methods that involve the mapping of pixel positions in one image plane onto another. Schenck and Möller describe [8] a *visual forward model* which takes as input the current view and predicts an image as it would appear after a given saccade, i.e. a change in gaze direction. Thus, the synthesized view only contains information already present in the input image but "warped" according to the new gaze direction.

In this article we present a method that combines image synthesis with image warping. A robotic agent equipped with a stereo camera head and a serial manipulator with an attached two-finger gripper learns to associate a certain arm posture and gaze direction with the corresponding view of its gripper. Here, the image synthesis part — termed *visual associative model* — takes as inputs the joint angles of the manipulator and returns the corresponding image of the gripper. These images always appear as if the gaze is directed towards the gripper. To generate views of the gripper from arbitrary gaze directions, we employ a *visual forward model* which is used to warp the image of the current view according to a given saccade. The saccade is generated from the desired gaze direction and the gaze direction that would fixate the gripper. Thus we furthermore require a model — the *kinesthetic associative model* — which associates an arm posture with a gaze direction such that the gripper is fixated.

2 Robotic Agent

The robotic agent used throughout this study resembles roughly the upper torso of a human: it consists of a camera head with two cameras, each mounted on a pan-tilt unit (PTU), and a 6-degrees-of-freedom serial manipulator with an attached two-finger gripper. The cameras are mounted above the arm. The whole set-up faces a table which serves in our experiments only as a background.

2.1 Camera Head

The camera head consists of two analog cameras which provide RGB images of 320×240 pixels. We denote the gaze direction by a vector $\tilde{\boldsymbol{v}} = (p_l, t_l, p_r, t_r)^T$, where p_l, t_l denote the left pan/tilt angles and p_r, t_r denote the right pan/tilt angles, respectively. In the following, we will employ a vergence model [7] in

which the right and left pan/tilt angles are coupled. The mapping from the coupled parameters (p, t, v_h, v_v) onto the individual ones is given by the relation [7]: $p_{l/r} = (p \pm \lambda_h(\frac{1}{2}v_h + \frac{1}{2}))/(1 + \lambda_h)$, $t_{l/r} = (t \pm \lambda_v v_v)/(1 + \lambda_v)$, where p, t denote the coupled pan/tilt values and v_h, v_v denote the horizontal and vertical vergence values, respectively ($\lambda_h = 0.5, \lambda_v = 0.2$).

The vergence model is biologically more plausible than controlling both PTUs independently. Furthermore, additional depth information is implicitly encoded in the horizontal vergence value: fixations in the near result in smaller horizontal vergence values (i.e. the cameras are rotated towards each other), while fixations in the distance result in greater values (i.e. the angle between the cameras axes approaches 0°).

2.2 Manipulator

The agent is furthermore equipped with a robot manipulator with six rotatory degrees of freedom. The joint angles will be denoted by $\theta_1, \ldots, \theta_6$. The manipulator can be decoupled into arm (joints 1–3) and wrist (joints 4–6). Thus, its inverse kinematics can be calculated in closed form. Note that this computation is just a shortcut which is needed in order to collect a large sample of training patterns.

Attached to the manipulator is a gripper with two fingers. The longitudinal axis of the gripper is always kept parallel to the ground while it can take three different orientations about the vertical axis, i.e. $\{0°, 15°, 30°\}$.[3]

3 Kinesthetic Association

The kinesthetic associative model directs the agent's gaze towards the center of its gripper by associating an arm posture (i.e. a set of joint angles) with a gaze direction (i.e. pan, tilt and vergence values). The purpose of this model is two-fold: during the training of the visual associative model it is used to collect training images of the gripper for different arm postures. In the application phase, the corresponding arm posture is used to generate saccades which are then used to drive the visual prediction (visual forward model).

The associative model is implemented as a 3-layer feed-forward neural network [6] with 6 inputs (corresponding to the 6 joint angles), a hidden layer with 40 units and 4 outputs (corresponding to the pan, tilt and horizontal/vertical vergence values). Linear activation functions were used for the output layer and sigmoid functions (tanh) for the hidden layer. The training data was collected by approaching points within a regular grid of end effector coordinates. The cameras were controlled by a saccade controller [7] such that the gripper was fixated for every grid point.

We chose an adaptive solution to this problem, although we are aware that there exist simple engineering solutions which rely on computing the inverse kinematics of the camera head. However, we think that using an adaptive approach here is more appropriate for a biologically oriented model like ours.

[3] At an orientation of 0°, the gripper is pointing away from the cameras.

3.1 Saccade Control

A saccade controller [7] is a controller which directs the gaze towards a salient object. In terms of control theory, the reference corresponds to both image centers, the system input are the pan, tilt and vergence values and the measured output is the centroid of the salient object in image coordinates (separately for both images). Thus, the measured error is the deviation between the image centers and the centroid in the left and right camera image.

We chose a simple P-type Controller for this purpose. The controller equation is given by $\Delta\boldsymbol{v} = \boldsymbol{G}\boldsymbol{e}$, where $\Delta\boldsymbol{v}$ denotes the change in gaze direction (i.e. the saccade), $\boldsymbol{G}$ denotes the gain matrix (see [7] for details), and $\boldsymbol{e}$ is the error vector containing the deviations between the image center and the centroid of the salient object in x and y direction for the left and right image, respectively. Note that the image coordinates are scaled to be in the range $[-1; 1]$. In order to avoid oscillations of the controller, the controller tolerates errors of 1 pixel in each direction.

3.2 Training

For the training we defined a rectangular workspace of size $150\,mm \times 120\,mm \times 300\,mm$ that was sampled by a $7 \times 6 \times 15$ regular grid. Thus, the distance between adjacent grid positions is approximately $20\,mm$ in each direction. The whole grid was sampled for each gripper orientation $\alpha \in \{0°, 15°, 30°\}$ separately. The six joint angles were calculated from the Cartesian coordinates using inverse kinematics (IK).[4] From the 8 theoretically possible IK solutions of the given arm only those belonging to a specific family (i.e. elbow down) were selected. Furthermore, a collision detector was used in order to avoid collisions of the arm with itself or its environment. Thus, the actual number of approached grid points deviates from the theoretical number (630) for the different orientations: 0° (563), 15° (483), 30° (355).

During the collection of the training patterns, the gripper held a salient target that was fixated using the saccade controller; this was repeated for every grid position. A training pattern is a pair $(\boldsymbol{\theta}, \boldsymbol{v})$, where $\boldsymbol{\theta} \in \mathbb{R}^6$ denotes the vector of joint angles and $\boldsymbol{v} \in \mathbb{R}^4$ denotes the gaze direction. There are 1381 such examples in total. The three-layer network was trained off-line using resilient propagation (RProp) [5]. Furthermore, the total number of patterns was divided into a training set (70%), and disjoint test and validation sets (both 15%). If the error on the validation set did not decrease for 200 epochs, the training was terminated (early stopping).

4 Visual Association

The visual associative model takes the joint angles ($\boldsymbol{\theta}$) as input and returns the corresponding (fixated) view of the gripper. Image data is usually high-dimensional (depending on the resolution of the images) which would require a

[4] Note that using the regular grid and inverse kinematics for the collection of the training examples is just a technical shortcut.

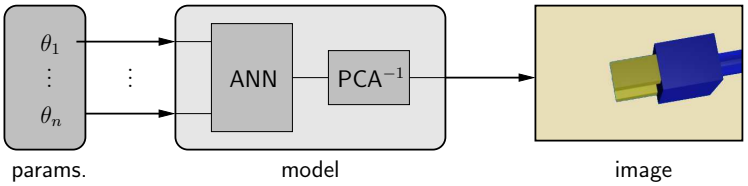


Fig. 1. Model architecture for visual association.

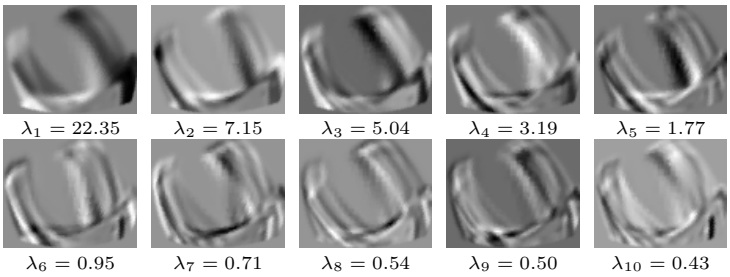


Fig. 2. The 10 principal eigen-images and the corresponding eigenvalues. Note that the values were scaled to be in the range $[0; 1]$.

large associative network. For this reason, we propose a model that has a two-stage architecture (see figure 1): the images are represented as low-dimensional appearance vectors which are then associated with the corresponding arm postures. Seeking a low dimensional representation for the gripper images seems reasonable, because it is most likely that they occupy a sub-space of relatively low dimensionality within the full image space.

The association between the joint angles and the appearance vectors is performed by a three-layer feed-forward network with linear output units and sigmoid hidden units. During the training phase, the images are transformed into appearance vectors which served as targets for the network training. During the application phase, the images are reconstructed from the network output by applying the inverse transformation. Irrelevant information is discarded during this transformation, which results in a small reconstruction error.

4.1 Eigen-Images

We use a principal component analysis (PCA) approach to extract the appearance vectors from the images. Let an image be denoted by $\boldsymbol{i} \in \mathbb{R}^M$, where M is the number of pixels, then its appearance vector is given by $\boldsymbol{a} = \boldsymbol{V}^T(\boldsymbol{i} - \bar{\boldsymbol{i}})$. Here $\boldsymbol{V} \in \mathbb{R}^{M \times m}$ is a projection matrix with orthonormal columns and with $m \ll M$, and $\bar{\boldsymbol{i}}$ is the mean image. Note that if $\boldsymbol{V}$ is chosen such that the elements of $\boldsymbol{a}$ are mutually uncorrelated and the variance $\mathrm{Var}[a_i]$ is maximized, then the elements of $\boldsymbol{a}$ are the principal components of $\boldsymbol{i}$. Such a projection matrix can be efficiently calculated by using a trick which is closely associated with eigenfaces [9].

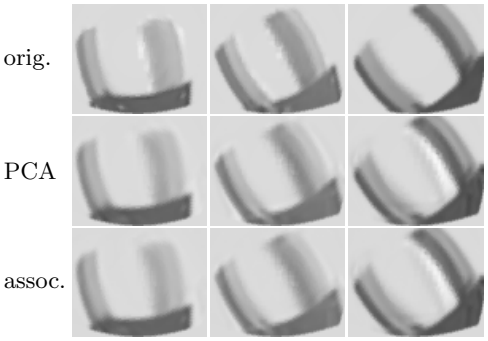


Fig. 3. Example images from three different grid positions. Original images (top row), PCA reconstruction (middle row), and output of the associative model (bottom row).

Let $\boldsymbol{X} = [\tilde{\boldsymbol{i}}_1, \ldots \tilde{\boldsymbol{i}}_N]$ denote the data matrix of all N images, where $\tilde{\boldsymbol{i}}_k = \boldsymbol{i}_k - \bar{\boldsymbol{i}}$ denotes the k-th mean-centered image, then the projection can be calculated by performing an eigen-decomposition $\boldsymbol{X}\boldsymbol{X}^T = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T$, and computing $\tilde{\boldsymbol{V}} = \boldsymbol{X}^T\boldsymbol{U}$. Normalizing the columns of $\tilde{\boldsymbol{V}}$ yields the desired $\boldsymbol{V}$.

The inverse transformation which is used to reconstruct an image from its appearance vector is given by $\hat{\boldsymbol{i}} = \boldsymbol{V}\boldsymbol{a} + \bar{\boldsymbol{i}}$. Note that the reconstruction is only approximately correct; a measure for the deviation between an original image and its reconstruction is the reconstruction error $E_{\text{reco}} = \|\hat{\boldsymbol{i}} - \boldsymbol{i}\|^2$.

4.2 Image Processing and Training

The training was conducted in analogous fashion to section 3.2. This time, the gripper fingers were in an open position as it would be before grasping. For each grid position, the inverse kinematics is used to calculate the corresponding joint angles and the arm is moved into this position. The previously trained kinesthetic associative model then generates a gaze direction such that both cameras fixate the gripper. The PTUs are moved and the camera images are stored.[5]

The original 320×240 images were cropped to a small 45×35 region around the gripper (which is roughly located in the image center). The gripper was then separated from the background based on color information. The RGB values were converted to gray scale values and inserted into new images of size 240×240 by adding a margin. These images were warped by a retinal mapping [8, 7]. Thus, the resolution is higher in the central part of the image and lower in the periphery (fovea effect).

The retinal images were cropped to 85×69 pixels around the center. These center crops were used for computing the appearance vectors as described in section 4.1. Here we used a number of $m = 10$ eigen-images (see figure 2). The average reconstruction error, i.e. the deviation between the original images and their reconstructions, amounted to $E_{\text{reco}} = 3.47$.

[5] For the results presented in this paper, only the left camera is used.

Fig. 4. Example outputs of the visual forward model for three different saccades (A, B, C) and two different inputs (A, B vs. C).

The visual associative network has a similar structure as the kinesthetic associative model (i.e. a 3-layer topology). We chose a relatively large hidden layer, consisting of 100 sigmod units. A training pattern is a pair $(\boldsymbol{\theta}, \boldsymbol{a})$, consisting of an arm posture $\boldsymbol{\theta} \in \mathbb{R}^6$ and the appearance vector of the corresponding gripper view $\boldsymbol{a} \in \mathbb{R}^{10}$; the total number of such patterns was $N = 1381$. For the training we used the RProp algorithm with early stopping.

Figure 3 shows the original views of the gripper (upper row) for 3 different grid position. The artifacts (e.g. the white shadow) imposed by the reconstruction from the appearance vectors (middle row) are also present in the associated views (bottom row). Furthermore, the reconstruction from the network (bottom row) output does not differ noticeably from the PCA reconstruction (middle row).

5 Visual Forward Model

The visual forward model [8] is a predictor for the visual consequences of a saccade. It receives as input the retinal images computed from the current view of the cameras and a motor command (i.e. a change in gaze direction). From these inputs, the visual forward model predicts an image according to a given change gaze direction, $\Delta\boldsymbol{v}$ and the current view. A so-called validator model indicates if a pixel can be faithfully predicted.

6 Results

We use the associated images of two example arm postures (see figure 3, 1st and 2nd column) to give a qualitative impression of the performance of the overall model. Before the application of the visual forward model, the gripper is roughly located in the center of the images. To transform the gaze direction towards the gripper (generated by the kinesthetic associative model) into the desired gaze direction, a saccade is computed and fed as input to the visual forward model. The resulting output of the visual forward model is shown in figure 4. In figure 4 A and B the same input images (generated by the visual associative model) were used, A shows the predictions of an upward tilt movement, B shows the

prediction of the same tilt movement, combined with a right pan movement. Figure 4 C uses a different input image; here we see the predicted results of a upward tilt movement, combined with a left pan movement. Note that the black regions at the image borders correspond to regions which were marked as unpredictable by the validator model (which is part of the visual forward model).

7 Conclusion and Outlook

We presented a model architecture for the association between the joint angles of a robot manipulator and corresponding views of its gripper. Furthermore, we showed how the output of the associative model can be warped according to an arbitrary gaze direction by a mechanism termed "visual forward model". The architecture is fully adaptive since it is based on artificial neural networks and subspace methods from pattern recognition. We presented some examples which suggest the association capabilities result in valid images. Nevertheless, the quality of the generalization ability has to be analyzed quantitatively.

For the experiments in this study, we used several shortcuts, e.g. reducing the number of exploration trials by defining a regular grid of spatial positions, and the extensive use of off-line learning. These shortcuts are problematic from a modelling perspective; in a more realistic setting, the agent would learn its sensory-motor associations on-line while performing exploration movements. This will be the subject of further research.

References

1. Glasbey, C.A., Mardia, K.V.: A review of image-warping methods. Journal of Applied Statistics 25(2), 155–171 (1998)
2. Jägersand, M.: Image based view synthesis of articulated agents. In: IEEE Conf. Comp. Vis. and Pat. Recog. (CVPR '97). pp. 1047–1053 (1997)
3. Jeannerod, M.: Mental imagery in the motor context. Neuropsychologia 33(11), 1419–1432 (1995)
4. Kosslyn, S.M., Alpert, N.M., Thompson, W.L., Maljkovic, V., Weise, S.B., Chabris, C.F., Hamilton, S.E., Rauch, S.L., Buonanno, F.S.: Visual mental imagery activates topographically organized visual cortex: PET investigations. Journal of Cognitive Neuroscience 5(3), 263–287 (1993)
5. Riedmiller, M., Braun, H.: A direct adaptive method for faster backpropagation learning: the RPROP algorithm. In: IEEE International Conference on Neural Networks. vol. 1, pp. 586–591 (1993)
6. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Nature 323, 533–536 (1986)
7. Schenck, W.: Adaptive Internal Models for Motor Control and Visual Prediction. MPI Series in Biological Cybernetics, Logos Verlag, Berlin (2008)
8. Schenck, W., Möller, R.: Training and application of a visual forward model for a robot camera head. In: Butz, M.V. et al. (eds.) ABiALS: From Brains to Individual and Social Behavior, pp. 153–169. No. 4520 in Lecture Notes in Artificial Intelligence, Springer, Berlin, Heidelberg, New York (2007)
9. Turk, M., Pentland, A.: Eigenfaces for recognition. Journal of Cognitive Neuroscience 3(1), 71–86 (1991)

Direct Policy Search: Intrinsic vs. Extrinsic Perturbations

Verena Heidrich-Meisner and Christian Igel*

Institut für Neuroinformatik
Ruhr-Universität Bochum, 44780 Bochum, Germany
verena.heidrich-meisner@ini.rub.de,christian.igel@ini.rub.de

Abstract. Reinforcement learning (RL) is a biological inspired learning paradigm based on trial-and-error learning. A successful RL algorithm has to balance exploration of new behavioral strategies and exploitation of already obtained knowledge. In the initial learning phase exploration is the dominant process. Exploration is realized by stochastic perturbations, which can be applied at different levels. When considering direct policy search in the space of neural network policies, exploration can be applied on the synaptic level or on the level of neuronal activity. We propose neuroevolution strategies (NeuroESs) for direct policy search in RL. Learning using NeuroESs can be interpreted as modelling of extrinsic perturbations on the level of synaptic weights. In contrast, policy gradient methods (PGMs) can be regarded as intrinsic perturbation of neuronal activity. We compare these two approaches conceptually and experimentally.

Key words: Evolution Strategies, Reinforcement Learning, Direct Policy Search, Covariance Matrix Adaptation, CMA-ES

1 Introduction

We propose neuroevolution strategies (NeuroESs) for direct policy search in reinforcement learning (RL). The algorithm gives striking results on RL benchmark problems ([4]; [9, 10, 12]). It corresponds to trial-and-error learning based on extrinsic dynamic perturbations of synaptic weights [8].

The perhaps most elaborate NeuroES is the covariance matrix adaptation NeuroES (CMA-NeuroES), which relies on the covariance matrix evolution strategy (CMA-ES, [7, 6, 17]). It is a variable-metric algorithm learning an appropriate metric for efficient adaptation. By memorizing successful learning steps, the CMA-NeuroES infers a learning direction from scalar reinforcement signals. We compare the CMA-NeuroES to state-of-the-art RL algorithms relying on perturbations in the activity of neurons encoding actions. These learning strategies rely on similar concepts (e.g., random perturbation, metric adaptation). They differ in the levels on which these concepts are applied and in the amount of

* This paper is based on [8, 13]

information that needs to be estimated from interactions with the environment. We have evaluated the different learning approaches on standard benchmark problems taken from the machine learning literature, and the CMA-NeuroES shows considerably better performance, especially in terms of robustness [9, 10]. While many related RL strategies, both in technical applications as well as in neural models, are based on gradient estimation, NeuroESs are driven by ranking policies. We argue that this decisive difference makes NeuroESs much more robust against noise or uncertainty in the learning process. Uncertainty, which may arise from various sources, is inherent in the general RL scenario, and thus, we must be able to cope with it.

2 The CMA-(Neuro)ES for RL

Evolution strategies are random search methods [1]. They iteratively sample a set of candidate solutions from a probability distribution over the search space (i.e., the space of policies), evaluate these potential solutions, and construct a new probability distribution over the search space based on the gathered information. In evolution strategies, this search distribution is parametrized by a set of μ candidate solutions and by parameters of the variation operators that are used to create new candidate solutions from these candidates. We first describe the standard CMA-ES, and then explain the application to neural networks.

In each iteration k of the CMA-ES, the lth candidate policy with parameters $\boldsymbol{x}_l^{(k+1)} \in \mathbb{R}^n$ ($l \in \{1, \ldots, \lambda\}$) is generated by multi-variate *Gaussian mutation* and *weighted global intermediate recombination*:

$$\boldsymbol{x}_l^{(k+1)} = \boldsymbol{m}^{(k)} + \sigma^{(k)} \boldsymbol{z}_l^{(k)}$$

The *mutation* $\boldsymbol{z}_l^{(k)} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{C}^{(k)})$ is the realization of a normally distributed random vector with zero mean and covariance matrix $\boldsymbol{C}^{(k)}$. The recombination is given by the weighted mean $\boldsymbol{m}^{(k)} = \sum_{l=1}^{\mu} w_l \boldsymbol{x}_{l:\lambda}^{(k)}$, where $\boldsymbol{x}_{l:\lambda}^{(k)}$ denotes the lth best individual among $\boldsymbol{x}_1^{(k)}, \ldots, \boldsymbol{x}_\lambda^{(k)}$. This corresponds to rank-based selection, in which the best μ of the λ offspring form the next parent population. A common choice for the recombination weights is $w_l \propto \ln(\mu + 1) - \ln(l)$, $\|\boldsymbol{w}\|_1 = 1$. The quality of the individuals is determined by the function `performance(`$\boldsymbol{x}$`)`, which corresponds to the evaluation of the policy with parameters $\boldsymbol{x}$.

The CMA-ES is a variable-metric algorithm adapting both the n-dimensional *covariance matrix* $\boldsymbol{C}^{(k)}$ of the normal mutation distribution as well as the *global step size* $\sigma^{(k)} \in \mathbb{R}^+$. The covariance matrix update has two parts, the rank-1 update considering the change of the population mean over time and the rank-μ update considering the successful variations in the last generation. For example, the rank-1 update is based on a low-pass filtered *evolution path* $\boldsymbol{p}^{(k)}$ of successful steps

$$\boldsymbol{p}_c^{(k+1)} \leftarrow c_1 \boldsymbol{p}_c^{(k)} + c_2 \frac{\boldsymbol{m}^{(k+1)} - \boldsymbol{m}^{(k)}}{\sigma^{(k)}}$$

and aims at changing $\boldsymbol{C}^{(k)}$ to make steps in the promising direction $\boldsymbol{p}^{(k+1)}$ more likely by morphing the covariance towards $\left[\boldsymbol{p}_c^{(k+1)}\right]\left[\boldsymbol{p}_c^{(k+1)}\right]^{\mathrm{T}}$. For details of the CMA-ES (the choice of the constants $c_1, c_2 \in \mathbb{R}^+$, the rank-$\mu$ update, the update of σ, etc.) we refer to the original articles by Hansen et al. [7, 6].

Neural networks (NNs) are flexible function approximators and a good choice for modeling policies if the shape of the optimal policy is unknown. The NN properties depend on both their particular structure and their weights. The CMA-ES as a very robust state-of-art optimization method is very well suited to perform the task of adapting the weights. The structure of the NN needs to be defined a priori, the CMA-NeuroES then explores the policy space spanned by all possible weight combinations. Applying the CMA-ES to weight optimization in neural networks directly models extrinsic perturbations on the level of synaptic weights. The experiments on the pole balancing task (see Table 1) illustrate that extrinsic perturbations can be more successful –and in particular more robust– than intrinsic perturbations (as implemented by RPG in Table 1).

Table 1. Mean number of episodes required for different RL algorithms to solve the partially observable double pole balancing problem (i.e., pole and cart velocities are not observed), using the standard performance function and using the damping performance function, respectively, see [5]. The CMA-ES adapts standard recurrent neural network representing policies. The CMA-NeuroES results are taken from the paper by [12] and the other results were compiled by [4]. The abbreviation RWG stands for Random Weight Guessing, and RPG for Recurrent Policy Gradients. The other methods are evolutionary approaches, SANE stands for Symbiotic Adaptive Neuro-Evolution, CNE for Conventional Neuroevolution, ESP for Enforced Sub-Population (ESP), NEAT for NeuroEvolution of Augmenting Topologies, and CoSyNE for Cooperative Synapse Neuroevolution (CoSyNE) (see [4, 12], for references).

method	reward function	
	standard	damping
RWG	415,209	1,232,296
SANE	262,700	451,612
CNE	76,906	87,623
ESP	7,374	26,342
NEAT	–	6,929
RPG	(5,649)	–
CoSyNE	1,249	3,416
CMA-ES	860	1,141

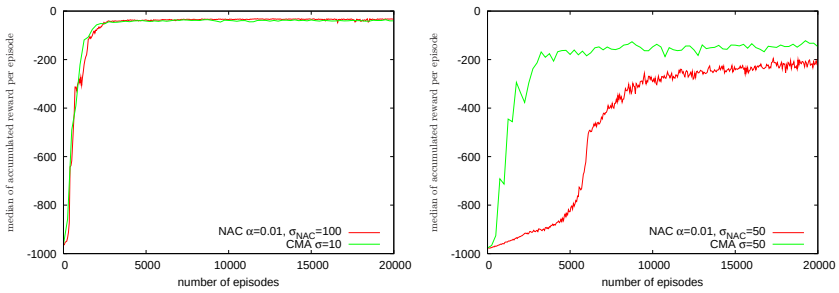


Fig. 1. Performance of natural actor-critic (NAC) and CMA-ES on the single-pole balancing task as described in [16]. The observed accumulated reward averaged over 20 trials is given as performance measure. Only linear policies are considered in this case. We show CMA-ES and NAC results for the best respective hyper-parameter configuration $\sigma, \sigma_{\text{NAC}}$, and α_{NAC}. In the left plot, the start states are initialized form the region $x \in [-0.2, 0.2]$, $\zeta \in [-0.2, 0.2]$. In the right plot, the starting region is $x \in [-2.0, 2.0]$, $\zeta \in [-0.6, 0.6]$. For the first (easier) starting region both methods perform comparatively. In the case of the larger staring region, the CMA-ES clearly outperforms the NAC, because the rank-based approach is more robust against noise induced by different starting positions. These results are taken from [9].

3 NeuroES and policy gradient methods for direct policy search

Direct policy search may be one way biological systems solve RL problems [3, 2]. In contrast to temporal difference learning, this approach does not require the prediction of future rewards.

RL is based on trial-and-error learning. Exploration in the space of neural networks can be modelled by stochastic perturbations, which can be applied at different levels: on the synaptic level or the level of neuronal activity. These perturbations can be *intrinsic* or *extrinsic* (e.g., see [3]). The former specifies that the neurons or synapses themselves are stochastic elements. The latter refers to some external source of perturbation. We consider the following two cases:

- Intrinsic perturbation of neuronal activity is used by policy gradient methods (PGMs). Policy gradient methods such as the natural actor-critic (NAC) perform a gradient-ascent on a predefined class of stochastic policies. PGMs explore the space of possible policies by stochastic neurons encoding actions and estimate a gradient.
- Extrinsic perturbations on the level of synaptic weights can be modelled with neuroevolution strategies (NeuroES). Extrinsic perturbations correspond to mutations. NeuroES rely on ranking candidate policies.

It is interesting to compare CMA-ES for RL and policy gradient methods. Both search directly in policy space, but ES for RL are actor-only methods while PGMs often have actor-critic architectures. In contrast to the CMA-ES, PGMs

require a differentiable structure on the search space and stochastic policies for learning. Exploration of the search space is realized by random perturbations in both ESs and PGMs. Evolutionary methods usually perturb a deterministic policy by mutation and recombination, while in PGMs the random variations are an inherent property of the stochastic policies. In ESs there is only one initial stochastic variation per episode. In contrast, the stochastic policy introduces perturbations in every time step of the episode. While the number n of parameters of the policy determines the n-dimensional random variation in ES, in PGMs the usually lower dimensionality of the action corresponds to the dimensionality of the random perturbations. In ESs the search is driven solely by ranking policies and not by the absolute values of performance estimates or even their gradients. The reduced number of random events and the rank-based evaluation are decisive differences and we hypothesize that they allow ESs to be more robust.

The adaptation of the covariance matrix in the CMA-ES is similar to learning the (Fisher) metric in natural PGMs [14, 16, 15]. Arguably one of the most elegant state-of-the-art natural PGMs is the episodic natural actor-critic algorithm (NAC, [16, 15]).

These two concepts of exploration lead to behavioral differences that become most apparent in noisy environments [10, 11]. Figure 1 shows the behavior of CMA-ES and NAC in the single-pole balancing task with linear policies. Both pole balancing benchmarks considered here illustrate that extrinsic perturbation on the level of synaptic weights as employed by the CMA-ES can indeed be more robust than intrinsic perturbations on the level of neuronal activity as employed by PGMs.

4 Conclusion: Why CMA-(Neuro)ES for RL?

As a summary, we give seven reasons why we propose the CMA-NeuroES for direct policy search. Employing the CMA-ES for RL

1. allows for direct search in policy space and does nor require learning of state-value or state-action-value functions for optimizing policies,
2. is straightforward to apply and robust w.r.t. tuning of hyper-parameters (e.g., compared to temporal difference learning algorithms or policy gradient methods),
3. is based on ranking policies, which is less susceptible to noise (e.g., caused by stochastic rewards and state transitions, random initialization, and noisy state observations) compared to estimating a value function or a gradient of a performance measure w.r.t. policy parameters (the overall number and the distribution of roll-outs among policies in the CMA-ES can be adapted as suggested by [8, 11]),
4. is a variable metric algorithm learning an appropriate metric (by means of adapting the covariance matrix and thereby considering correlations between parameters) for searching better policies,

5. can be applied if the function approximators are non-differentiable, whereas many other methods require a differentiable structure,
6. extracts a search direction, stored in the evolution path $\boldsymbol{p}_c^{(k)}$, from the scalar reward signals, and
7. efficiently optimizes weights of neural network policies, which corresponds to an implementation of extrinsic perturbations on the level of synaptic weights.

Arguably, the main drawback of the CMA-ES for RL in its current form is that the CMA-ES does not exploit intermediate rewards, just final Monte Carlo returns. This currently restricts the applicability of the CMA-ES to episodic tasks and may cause problems for tasks with long episodes.

Acknowledgements

We acknowledge support from the German Federal Ministry of Education and Research within the National Network Computational Neuroscience under grant number 01GQ0951.

References

1. H.-G. Beyer. Evolution strategies. *Scholarpedia*, 2(8):1965, 2007.
2. K. Doya and T. J. Sejnowski. A computational model of avian song learning. In M. S. Gazzaniga, editor, *The New Cognitive Neurosciences*, pages 469–482. MIT Press.
3. I. Fiete, M. Fee, and H. Seung. Model of Birdsong Learning Based on Gradient Estimation by Dynamic Perturbation of Neural Conductances. *Journal of Neurophysiology*, 98(4):2038, 2007.
4. F. Gomez, J. Schmidhuber, and R. Miikkulainen. Accelerated neural evolution through cooperatively coevolved synapses. *Journal of Machine Learning Research*, 9:937–965, 2008.
5. F. Gruau, D. Whitley, and L. Pyeatt. A comparison between cellular encoding and direct encoding for genetic neural networks. In J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 81–89. MIT Press, 1996.
6. N. Hansen. The CMA evolution strategy: A comparing review. In *Towards a new evolutionary computation. Advances on estimation of distribution algorithms*, pages 75–102. Springer-Verlag, 2006.
7. N. Hansen, S. D. Müller, and P. Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evolutionary Computation*, 11(1):1–18, 2003.
8. V. Heidrich-Meisner and C. Igel. Learning behavioral policies using extrinsic perturbations on the level of synapses. *Frontiers in Computational Neuroscience. Conference Abstract: Bernstein Symposium 2008*, page , 2008. doi: 10.3389/conf.neuro.10.2008.01.060.
9. V. Heidrich-Meisner and C. Igel. Similarities and differences between policy gradient methods and evolution strategies. In M. Verleysen, editor, *16th European Symposium on Artificial Neural Networks (ESANN)*, pages 149–154. Evere, Belgium: d-side publications, 2008.

10. V. Heidrich-Meisner and C. Igel. Variable metric reinforcement learning methods applied to the noisy mountain car problem. In S. Girgin et al., editors, *European Workshop on Reinforcement Learning (EWRL 2008)*, number 5323 in LNAI, pages 136–150. Springer-Verlag, 2008.
11. V. Heidrich-Meisner and C. Igel. Hoeffding and Bernstein races for selecting policies in evolutionary direct policy search. In L. Bottou and M. Littman, editors, *Proceedings of the 26th International Conference on Machine Learning (ICML 2009)*, pages 401–408, 2009.
12. V. Heidrich-Meisner and C. Igel. Neuroevolution strategies for episodic reinforcement learning. *Journal of Algorithms*, 64(4):152–168, 2009.
13. V. Heidrich-Meisner and C. Igel. Variable-metric evolution strategies for direct policy search. In *Multidisciplinary Symposium on Reinforcement Learning (MSRL 2009)*, 2009. http://msrl09.rl-community.org.
14. S. Kakade. A natural policy gradient. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems (NIPS14)*. MIT Press, 2002.
15. J. Peters and S. Schaal. Natural actor-critic. *Neurocomputing*, 71(7-9):1180–1190, 2008.
16. M. Riedmiller, J. Peters, and S. Schaal. Evaluation of policy gradient methods and variants on the cart-pole benchmark. In *Proc. IEEE Int'l Symposium on Approximate Dynamic Programming and Reinforcement Learning (ADPRL 2007)*, pages 254–261, 2007.
17. T. Suttorp, N. Hansen, and C. Igel. Efficient covariance matrix update for variable metric evolution strategies. *Machine Learning*, 75(2):167–197, 2009.

Attentive Stereoscopic Object Recognition

Frederik Beuth, Jan Wiltschut, and Fred H. Hamker

Chemnitz University of Technology,
Strasse der Nationen 62, 09107 Chemnitz, Germany
frederik.beuth@cs.tu-chemnitz.de,wiltschj@uni-muenster.de,
fred.hamker@cs.tu-chemnitz.de
http://www.tu-chemnitz.de/cs/KI/

Abstract. Object recognition in general is still a challenging task today. Problems arise for example from parallel segmentation and localization or the problem to detect objects invariant of position, scale and rotation. Humans can solve all these problems easily and thus neuro-computational and psychological data could be used to develop similar algorithms. In our model, attention reinforces the relevant features of the object allowing to detect it in parallel. Human vision also uses stereoscopic views to extract depth of a scene. Here, we will demonstrate the concept of attention for object recognition for stereo vision in a virtual reality, which could be applied in the future to practical use in robots.

Keywords: Object Recognition, Attention, Stereo Vision, Learning

1 Introduction

Object recognition is the task to recognize and additionally localize a searched object in an image or a scene. Many neuro-computational models, like Neocognitron [5] and HMAX [15, 19] filter the image over different stages to reduce the complexity of the filter operations. These systems are purely forward driven and do not consider the concept of attention.

Object recognition combined with attention can solve the dilemma of parallel segmentation and localization. We will first explain the concept of attention and how it solves this problem. We use a stereoscopic edge and depth detection model to achieve stereo object recognition. The object detectors are learned unsupervised and use a neuro-computational model which capture the basic principles of primate 3D perception. We will first focus on position invariant recognition and then demonstrate the ability to discriminate different objects. The model and the results are compared to neuro-computational findings.

1.1 Concept of Attention

Early concepts of visual attention define attention as to focus processing on a spatially determined part of the image, namely the spotlight of attention. The location of interest is typically determined from conspicuous or salient image features forming the saliency map [8, 10].

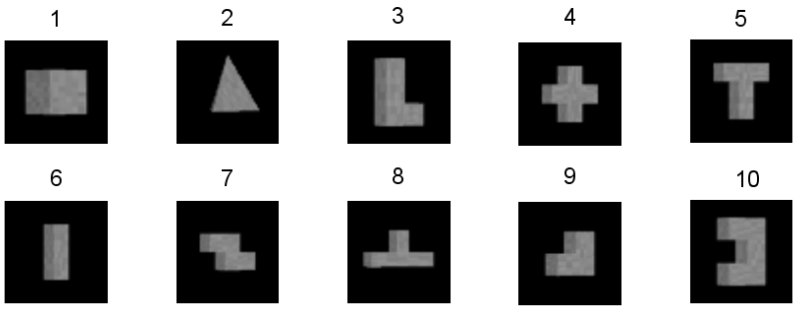


Fig. 1. The stimuli consist of 10 different 3D objects.

Recently, the "spotlight of attention" concept has been expanded to a feature-based approach [6] in which attention emerges from interactions between different brain areas. High level areas hold a template to specify the searched object and this information is propagated backwards to lower level areas. The parallel computation modifies the conspicuity of each descriptor in the system in such a way that the value represents the accumulated evidence. We implement the concept of attention as a modulating of the feed forward signals (called gain-control) dependent on the feed back from higher cortical areas. To perceive an object, a combination of several distributed visual features is required. Such binding processes can be well described by concepts of visual attention, illustrated by two continuous sub processes. The first one operates in parallel over all features and increases the conspicuity of those that are relevant for the searched object, independent of their location in the visual scene. The other subprocess is linked to action plans, e.g. eye movement plans, and combines those fragments which are consistent with the action plan, typically by their spatial location in the visual scene.

Object Selection and Segmentation For recognizing a searched object in a scene, the object must first be located and segmented, which however is only possible if the object has been recognized as such. Attention can solve this "Chicken-egg-problem" due to its parallel computation approach.[6]

2 Object Recognition System

2.1 Neuronal network architecture

We extend the concept of a population-based object representations [6] by learnable object representation based on local edge detectors. This allows to detect objects depending on their shape or texture. Additionally, we demonstrate the approach on stereoscopic images.

In our neuronal model (Fig. 2), we do not consider all the complexity of the visual stream. Rather we simulate an earlier area (V1) and a high level area

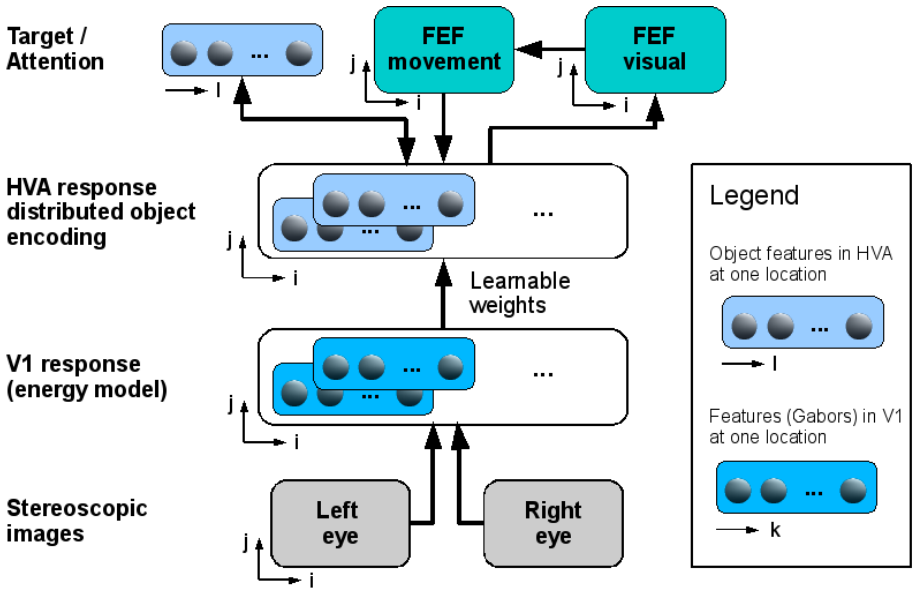


Fig. 2. Neuronal network of the stereoscopic object recognition model. The i and j indices correspondent to the spatial x and y axis of the images. The index k refers to different Gabor responses and l to different learned features in HVA.

(HVA) whose object selective cells can be mapped to area V2/V4/IT. As input stimuli we use the left and right eye view of 10 different 3D objects (Fig. 1), produced by a raytracer engine [1]. The objects are to some degree similar in their edges and thus the difficulty of the problem is comparable to cluttered scenes. The first area detects stereoscopic edges and disparities via an energy model (see [11, 13, 14]) and is comparable to area V1. This particular filter bank [17] uses 56 Gabors with 8 orientations (with a $\frac{\pi}{8}$ step size) and 7 different phase disparity shifts (with $\frac{\pi}{4}$ step size). This area builds a representation of the scene encoding edge informations, independent of the right or left view and therefore enables stereo object recognition. Overlapping receptive fields serve as input for the object selective cells of the HVA. We achieve the object selectivity by learning the feed forward weights (V1$\rightarrow$ HVA) with a biological motivated learning algorithm and a trace rule (see 2.3). The attention signal stores the features relevant for the current task. The Frontal Eye Field (FEF) consists of two areas, the saliency map (called FEFvisual) and the target of the next eye movement (called FEFmovement). One of the binding processes operates over all locations in HVA and reinforces the features of the searched object. The other is achieved by the loop over FEFvisual and FEFmovement and reinforces adjacent locations. Both processes use competition to decrease the activity of irrelevant features and location in HVA.

2.2 Neuron model

We use a rate coded neuron model which describes the firing rate r of a cell as its average spike frequency. Every cell represents a certain feature (V1: k, HVA: l) at a certain location (i, j). In the following we will omit the location indices for clarity. Consider one location in HVA, each cell in HVA gains excitation (as a weighted sum) from cells of V1 within the receptive field (here a 14x14 patch) and each cell is inhibited by all other HVA cells via Anti-Hebbian inhibition (similar as in [23]).

$$\tau_R \frac{\partial r_l}{\partial t} = \sum_i w_{kl} \cdot r_k^{\text{Input}} - \sum_{l',l' \neq l} f\left(c_{l,l'} \cdot r_{l'}\right) - r_l \quad \text{with: } f(x) = d_{nl} \cdot \log\left(\frac{1+x}{1-x}\right) \quad (1)$$

$f(x)$ gives the non-linear processing. τ_R is the time constant of the cells. The connection $w_{k,l}$ denotes the strength of the feed forward weight from input cell k to the output cell l. Lateral inhibition is given by the connection weight $c_{l,l'}$ and can differ across the cells due to the Anti-Hebbian learning.

2.3 Learning of the object descriptors

Changes in the connection strength between neurons in response to appropriate stimulation are thought to be the physiological basis for learning and memory formation [21]. In the visual system the connections between neurons (synapses) are modified according to a simple principle of joint firing, the Hebbian law [7]. According to this law synapses are strengthened if the corresponding cells are activated at the same time. Thus, over time cells "learn" to respond to and in connection with specific other cells. In our model object recognition is achieved by learning the connection weights ($w_{kl}^{\text{V1-HVA}}$) between V1 and HVA. Using a general learning algorithm, that has been shown to capture the features of early visual learning [23], cells from HVA tune themselves to specific features from the set of presented stimuli.

It has been hypothesized that the ventral pathway uses temporal continuity for the development of view-invariant representations of objects ([4, 16, 22]). This temporal continuity can be applied using a trace learning rule. The idea is that on the short time scale of stimuli presentation, the visual input is more likely to originate from different views of the same object, rather than from a different object. To combine stimuli that are presented in succession to one another, activation of a pre-synaptic cell is combined with the post-synaptic activation of the previous stimulus using the Hebbian principle. We simulate an appropriate input presentation protocol and the responses of successive stimuli are combined together to achieve a more invariant representation of an object.

During learning the connection weights $w_{k,l}^{\text{V1-HVA}}$ are changed over time according the Hebbian principle:

$$\tau_L \frac{\partial w_{kl}}{\partial t} = [r_l^{\text{HVA}} - \tilde{r}^{\text{HVA}}]^+ \left((r_k^{\text{V1}} - \tilde{r}^{\text{V1}}) - \alpha_w \left[r_l^{\text{HVA}} - \tilde{r}^{\text{HVA}}\right]^+ w_{kl}\right) \quad (2)$$

$\tilde{r}$ is the mean of the activation over the particular features $\left(e.g., \tilde{r} = \frac{1}{N}\sum_{l=1}^{N} r_l\right)$ and $[x]^+ = \max\{x, 0\}$. α_w constrains the weights analogous to the Oja learning rule [12] and τ_L is the time constant for learning. The V1-HVA weights are learned only at a single receptive field (a 14x14 patch of V1) and their values are shared with all other locations in the HVA (weight sharing approach). The learning was performed on small images containing a single stimulus before processing entire scenes (offline-learning).

Lateral connections between cells were learned by Anti-Hebbian learning. The name Anti-Hebbian implies that this strategy is the opposite of the Hebbian learning rule. Similar to the learning of the synaptic connection weights, where the connection between two cells is increased when both fire simultaneously, in the Anti-Hebbian case the inhibition between two cells is strengthened. The more frequent two cells are activated at the same time, the stronger they inhibit each other, increasing the competition among those two cells (l and l'):

$$\tau_C \frac{\partial c_{l,l'}}{\partial t} = r_{l'} \cdot r_l - \alpha_c r_{l'} \cdot c_{l,l'} \tag{3}$$

where τ_C is the learning rate of the Anti-Hebbian weights. Anti-Hebbian learning leads to decorrelated responses and a sparse code of the cell population [3].

3 Results

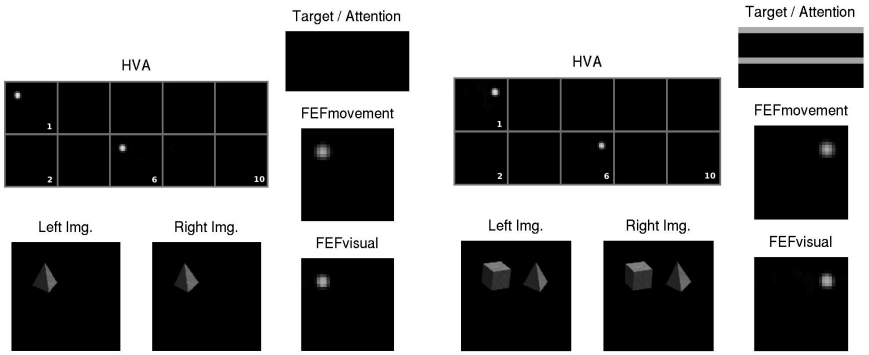
We show the ability of object recognition independent of its position within a scene containing also a distractor object. We measure the performance to recognize all objects with a discriminating value.

3.1 Object recognition independent of its position

An object must be recognized independent of its position in the image, its rotation or its relative size (for an overview see [19]). Position invariance is achieved in the cortex by pooling over a certain spatial area, which is also part of our model.

We now show an object location experiment:

1. We present an object alone in a scene without an attention signal (Fig. 3(a)). The model selects the most conspicuous region (the object) and binds the HVA activation to the working memory (which stores in our example the attention signal).
2. We present a black screen to deplete all cell activities in the system.
3. We test the ability to select the target object. We present a cluttered scene (Fig. 3(b)) (here for simplicity with only 10 features and 2 objects). The attention signal encodes the features of the object and reinforces them in HVA. By this, the system is able to locate the object again (spatial invariant recognition).



(a) (b)

Fig. 3. The figure shows the layer activities during the object location experiment. Here the stereoscopic stimuli, the responses of the feature code (with 10 features) in HVA, the attention signal (features on the y-axis) and both FEF areas are shown. Normally, the x and y axis correspondent to the spatial x and y axis of the images. **a)** The system memorizes the target object, the 'tetrahedron', and stores the HVA response as an attention signal for b). **b)** The attention signal reinforces the features which represent the 'tetrahedron' and the system detects the target object.

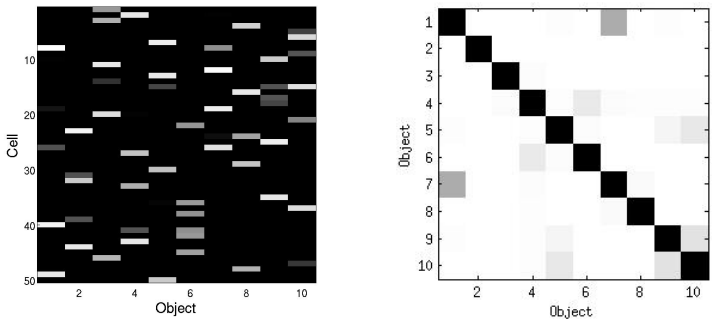
3.2 Object discrimination

To determine the similarity of two feature codes $(\mathbf{r}, \mathbf{s})$ the angle between those two vectors is considered. The lower the value of $d_{TM} \in [0; 1]$ is the more the two vectors show similar cell distributions.

$$d_{TM}(\mathbf{r}, \mathbf{s}) = 1 - \frac{\langle \mathbf{r}, \mathbf{s} \rangle}{|\mathbf{r}|\,|\mathbf{s}|} \quad \text{with: } dim(\mathbf{r}) = \dim(\mathbf{s}) \tag{4}$$

Our results show that regardless of the number of different objects and independently of the number of cells (as long as there is at least one cell per object) the model is capable to learn and discriminate all objects. It can be seen that each object is learned by several cells (Fig. 4(a)) and thus an object is characterized by a specific distributed feature code with nearly no overlap to other objects.

An analysis of whether the model is able to discriminate among the objects is shown in Figure 4(b) using the discrimination value (d_{TM}). Low values (indicated by darker areas) give clue to similar feature codes which would indicate that discrimination between those two objects is impaired. The results show that all objects are very dissimilar in their features and thus are very easy to discriminate. Only object 1 and object 7 show slightly overlapping population codes ($d_{TM} = 0.68$) but the objects can easily be discriminated (compare Figure 4(a)). Although some cells tend to code more than one object the results show that all objects can be discriminated perfectly due to the specific distributed code.



(a) Object selectivity (b) Discrimination between objects

Fig. 4. a) For each object (x-axis) the average firing response (0 dark, 1 bright) of each feature/cell (50 features on the y-axis) is plotted. The average firing response is calculated over all input stimuli that contain the same object. **b)** Using the discrimination value d_{TM}, the similarity of the average response (Fig. 4(a)) to an object is shown here (bright = dissimilar).

4 Discussion

To summarize, attention driven object recognition can solve the problem of selection and segmentation. We had successfully combined stereo vision with object recognition which requires to merge the two views of a scene. We have constructed a merged representation of the scene in low (V1) and high levels areas (HVA). Compared with the perspective of computer vision, this can be seen as a hybrid solution of two contrary approaches. One of them is to construct a merged high level scene model from both images [2], the other one is to combine both images at the level of pixels (resulting in the correspondence problem [18]), which leads to a large number of local false matches.

Our learning algorithm captures the basics of human perception, but can extend to cover complex cell dynamics like calcium traces [20]. We have shown that our system models invariances of the visual cortex. We have focused on spatial invariance and therefore we will have to extend the model and its learning algorithm to scale and rotation invariance. Most neurons in higher areas have a small rotation and scale invariance, but encode a single view to the object (called view-tuned cells [9]). In further investigations we can compare the properties of the learned cells in the HVA with the view-tuned cells.

Acknowledgments. This work has been supported by the EC Project FP7-ICT "Eyeshots: Heterogeneous 3-D Perception across Visual Fragments".

References

1. Chumerin, N.: Nikolay Chumerin's myRaytracer (2009), `OnlineResource:http://sites.google.com/site/chumerin/projects/myraytracer`

2. van Dijck, H.: Object recognition with stereo vision and geometric hashing. Ph.D. thesis, University of Twente (1999)
3. Földiàk, P.: Forming sparse representations by local anti-hebbian learning. Biol Cybern 64, 165–170 (1990)
4. Földiàk, P.: Learning invariance from transformation sequences. Neural Computation 3, 194–200 (1991)
5. Fukushima, K.: Self-organizing neural network models for visual pattern recognition. Acta Neurochir Suppl (Wien) 41, 51–67 (1987)
6. Hamker, F.H.: The emergence of attention by population-based inference and its role in distributed processing and cognitive control of vision. Comp Vis Image Understand 100, 64–106 (2005)
7. Hebb, D.O.: Organization of Behavior. John Wiley and Sons (1949)
8. Itti, L., Koch, C.: A saliency-based search mechanism for overt and covert shifts of visual attention. Vision Res 40(10-12), 1489–506 (2000)
9. Logothetis, N.K., Pauls, J., Poggio, T.: Spatial reference frames for object recognition tuning for rotations in depth. In: AI Memo 1533, Massachusetts Institute of Technology. pp. 12–0. MIT Press (1995)
10. Milanese, R.: Detecting salient regions in an image: from biological evidence to computer implementation. Ph.D. thesis, University of Geneva (1993)
11. Ohzawa, I., DeAngelis, G.C., Freeman, R.D.: Stereoscopic depth discrimination in the visual cortex: neurons ideally suited as disparity detectors. Science 249(4972), 1037–41 (Aug 1990)
12. Oja, E.: A simplified neuron model as a principal component analyzer. J Math Biol 15(3), 267–273 (1982)
13. Qian, N.: Computing stereo disparity and motion with known binocular cell properties. Neural Computation 6(3), 390–404 (1994)
14. Read, J.C.A., Cumming, B.G.: Sensors for impossible stimuli may solve the stereo correspondence problem. Nat Neurosci 10(10), 1322–8 (Oct 2007)
15. Riesenhuber, M., Poggio, T.: Hierarchical models of object recognition in cortex. Nat. Neurosci 2(11), 1019–25 (Nov 1999)
16. Rolls, E.T., Stringer, S.M.: Invariant object recognition in the visual system with error correction and temporal difference learning. Network 12(2), 111–129 (May 2001)
17. Sabatini, S., Gastaldi, G., Solari, F., Diaz, J., Ros, E., Pauwels, K., Van Hulle, M., Pugeault, N., Krüger, N.: Compact and accurate early vision processing in the harmonic space. In: International Conference on Computer Vision Theory and Applications (VISAPP), Barcelona (2007)
18. Scharstein, D., Szeliski, R.: A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. Int. J. Comput. Vision pp. 7–42 (2002)
19. Serre, T.: Learning a Dictionary of Shape-Components in Visual Cortex: Comparison with Neurons, Humans and Machines. Ph.D. thesis, Massachusetts Institute of Technology (2006)
20. Shouval, H.Z., Castellani, G.C., Blais, B.S., Yeung, L.C., Cooper, L.N.: Converging evidence for a simplified biophysical model of synaptic plasticity. Biol Cybern 87(5-6), 383–91 (Dec 2002)
21. Squire, L.R., Kandel, E.R.: Memory: From Mind to Molecules. Roberts & Co Publ (2008)
22. Wallis, G., Rolls, E.T.: Invariant face and object recognition in the visual system. Prog Neurobiol 51(2), 167–194 (Feb 1997)
23. Wiltschut, J., Hamker, F.H.: Efficient coding correlates with spatial frequency tuning in a model of v1 receptive field organization. Vis Neurosci 26, 21–34 (2009)

Clustering of Hyperspectral Image Signatures Using Neural Gas

Udo Seiffert and Felix Bollenbeck

Fraunhofer IFF Magdeburg, Germany
[Udo.Seiffert, Felix.Bollenbeck]@iff.fraunhofer.de

Abstract. This paper shows the utilization of the Neural Gas (NG) algorithm to cluster hyperspectral image signatures in the context of crop plant phenotyping. Based on nutrition experiments of different tobacco lines (genotypes) it is shown that Neural Gas can unravel relevant and valuable biological information from hyperspectral images of the plant leaves. Neural Gas is able to demonstrate its beneficial properties in terms of numerical robustness, computational speed as well as initialization and parameter tolerance again. Moreover, NG provides an excellent basis for perspective implementations of more application specific dissimilarity measures rather than using the standard Euclidean or simple correlation based distance measures.

1 Introduction

Hyperspectral imaging provides high-dimensional pixel-wise signatures that are extracted from an acquired image stack containing an absorption spectrum of a scenery at a rather wide range of different wavelengths typically distributed over many (≥ 100) equidistant bins. Since each (surface) material – as a mixture of many different molecules – absorbs characteristic wavelengths (the reflected light of all wavelengths is actually measured), this technique can generally be used to detect different materials.

Artificial neural networks and machine learning based paradigms [5] have generally proven their excellent abilities to process spectral and hyperspectral data [14, 15]. Hyperspectral imaging itself is particularly interesting in those cases where the human eye or standard imaging sensors are not able to recognize material differences. Pioneering applications have been geology, remote sensing, and forestry. The image acquisition was usually satellite or airplane based. More recently, applications in agriculture and plant phenotyping can be found as well.

In both traditional plant breeding and green biotechnology it is necessary to quantitatively assess the content of biologically relevant compounds, such as metabolites, proteins, etc. Traditionally this is done by biochemical digested analysis. Along with the increasing demand of high-throughput screening, these wet lab techniques become more and more unfeasible due to their high technical and financial requirements and low throughput. In turn, imaging based approaches provide a non-invasive and non-destructive way of data collection.

This makes it much easier to gather data from plant samples during their development or maturation. If the applications are extended towards agricultural production (precision agriculture) a robust and easily manageable set-up is required [10, 7].

If for example biochemical data obtained by a suitable wet lab analysis about relevant plant compounds is available, it can be used as labels for supervised analyses. A typical scenario is to match various hyperspectral signatures to particular single compounds or groups of compounds. If the label data is quantitative, instead of just containing class labels, even quantitative assessments, e.g. regression models, are possible.

However, if no labeled data is available, or the more general question arises, what plant properties are detectable and distinguishable at all by these methods, unsupervised approaches come into play. Here a typical scenario would be to know what plant features dominate the obtained hyperspectral fingerprints. Potential features in this context, spanning a multi-dimensional non-linear space, could be:

- different lines (genotypes) of the plant samples in their phenotypic expression;
- different nutrition;
- positions of a particular leaf along the plant stem (corresponding to the age of the leaf);
- positions (of a considered pixel) within one leaf.

In order to address this question, data was acquired from systematic nutrition experiments. More details about the biological background as well as the data acquisition and pre-processing can be found in [9]. The present paper focusses on neural networks related aspects of the hyperspectral signature clustering using Neural Gas (NG) [6].

2 NG Clustering: Set-up, Implementation, and Results

After the necessary pre-processing steps, such as image conversion, reflectance normalization, continuum removal (see Fig. 1), a training data matrix of 536,517 samples (pixels) and 239 features (wavelengths) was available. This matrix contains the spectral abundance (16 bit float value) of a particular pixel at a particular wavelength.

The NG algorithm was run with different parameter variations (number of prototypes, initial learning coefficient, decay parameter, number of epochs) and 10-fold validation.

Since this data set did not fit into the computer's main memory, special precautions for an out-of-core training have been taken. This slightly slows down the training, but is still much faster than to allow memory page swapping. Using an off-the-shelf PC (Intel Core2 Quad CPU Q6660 @ 2.4 GHz, 8 GB RAM) one single training epoch took about 30 minutes (10 prototypes, Euclidean distance). A total number of 25 epochs turned out to be sufficient to obtain stable results. So this sums up to about 12 hours overall training time.

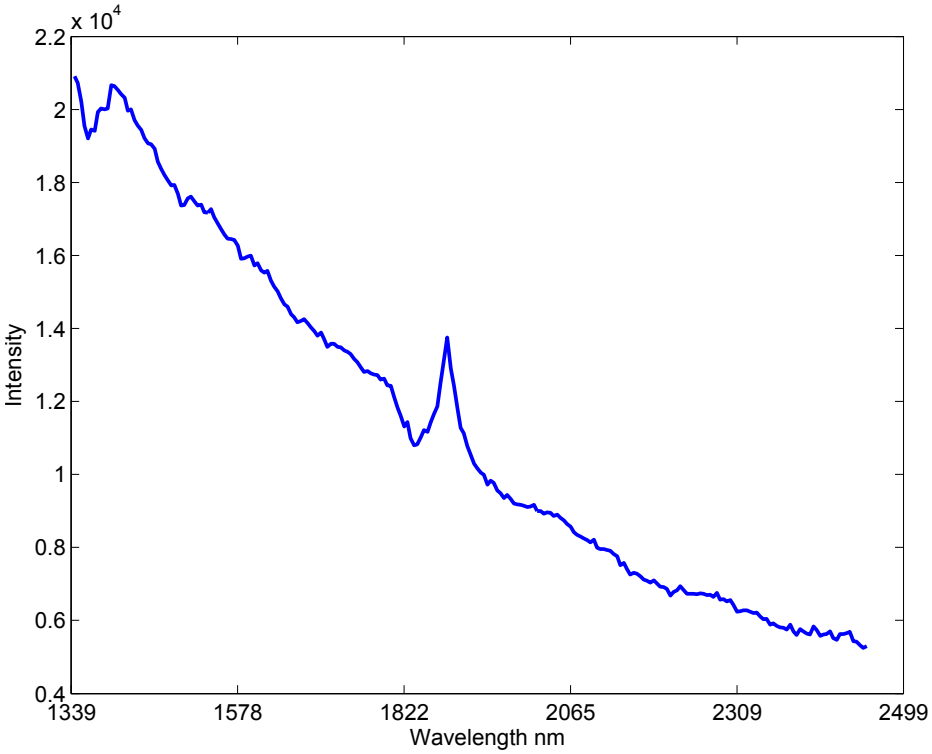


Fig. 1. Spectral intensity reflected by the used calibration normal (100%-reflection type, polytetrafluoroethylene (PTFE) pad). The typical physical effect of decreased reflected energy with increasing wavelengths can be clearly observed. The actual gradient of this curve depends on the specific characteristics of the measurement equipment, such as sensor, lens, and illumination source. This curve is used to normalize all acquired images of the scenery.

The obtained results generally turned out very stable in terms of parameter variation and prototype initialization. As could be expected, the number of prototypes is the most important training parameter. As usual, it functions as some kind of zoom into the data. A number between 5 and 10 prototypes seemed to be sensible (see Fig 2).

It also turned out that all further parameters and the prototype initialization do not significantly influence the results. Besides the fast convergence (in the light of the large data set), this robustness seems the most beneficial feature of Neural Gas – not only in this application.

The obtained prototypes for a ten-cluster-setting are shown in Fig. 4. The cluster membership of each pixel has been re-transferred into the original images as shown in Fig. 5.

With the described method a number of biologically relevant results could be obtained. In particular it can be demonstrated that concentration gradients

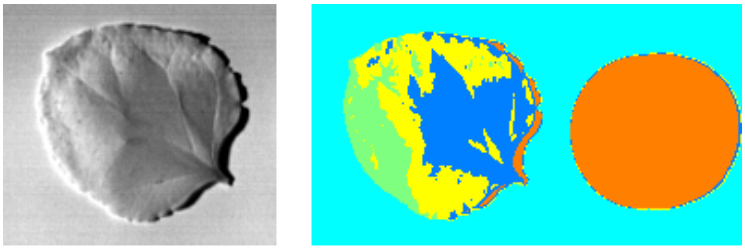


Fig. 2. Left panel: Sample leaf at arbitrarily chosen wavelength as gray-scale intensity image. The used broadband light source is located left of the leaf resulting in (realistic) inhomogeneous illumination and shadow on the right hand edge. Right panel: A total of 5 clusters obtained by NG being re-transferred into the original image using a fixed color code. The homogeneously clustered background and calibration pad can be observed.

of nutrients are quantitatively detectable depending on the intra-leaf position as well as the leaf position along the plant stem. Either nutrient concentrations correspond, besides the general nutrition and health level of the plant in terms of an offset, to the age/maturation grade of a tissue, leaf and thus finally of the whole plant. This offers novel perspectives in several related application fields, such as precision/high-throughput phenotyping and precision agriculture.

More specific results are to be expected by using supervised approaches. These require labeled data. This, in turn, typically leads to extended biochemical analyses. Extensive work into that direction is currently under way.

3 Perspectives of Application-specific Metrics

In the previous section a more or less standard Neural Gas based clustering framework was described. The results are already very promising.

However, a closer look at Fig. 2 and all further acquired hyperspectral image data lets the question arise, whether the clustering is affected by inhomogeneous illumination and unwanted reflections on the leaf surface. Whereas basic – physically non-avoidable – *spectral* illumination artifacts, for instance caused by inhomogeneous spectral energy density of the broadband light source (indoor imaging) or spectral absorptions of the sunlight by the earth's atmosphere (outdoor imaging) and wavelength-depending sensitivity of the sensor/optics, can be corrected by continuous entrainment of a calibration pad within the field-of-view, *local* illumination distortions are not covered by this normalization. These local illumination distortions typically arise because the artificial light is not perfectly diffuse (it comes just from one side) and/or the leaf is not plain and are typically leading to reflections and shadowing (as shown in Fig. 2). By using advanced equipment and special constructions on the image acquisition stage, such as for example a circular light source centered around the optical axis, these effects may be limited but not completely avoided.

This issue is addressed by a number of publications [16, 11]. Often these problems are considered as simple offset and tackled by correlation based preprocessing or correlation based dissimilarity measures within the clustering algorithm [2, 4].

At this point another advantageous property of prototype-based approaches in general and of Neural Gas in particular comes into play. Here, a 'non-standard' dissimilarity measure can be implemented easily. This offers an elegant way to incorporate application-specific knowledge directly into the clustering.

In prototype-based approaches, such as Neural Gas, the clustering algorithm needs to compare the currently processed input vector ($\mathbf{X}$) with several prototype (weight) vectors ($\mathbf{W}_i, i = 1 \ldots n$ with n being the number of prototypes). This distance or dissimilarity measure needs to be not necessarily in Euclidean space.

A possible first approach is to perform a Wavelet decomposition / transformation combined with a standard measure as shown in Fig. 6. Although a numerical cluster validation measure is not directly applicable, a visual inspection clearly shows a much better preserved leaf-specific tissue structure compared to the stand-alone Euclidean measure shown in Fig. 2.

Another possible way is to consider the individual spectral signatures (both the currently processed input and the prototypes) as densities, that are positive functions (patterns), not necessarily normalized but finite measures [12]. In general, a pairwise directed distance D between these densities is called a metric, if the following three conditions hold:

1. $D(\mathbf{X}\|\mathbf{W}) \geq 0$ (positive definiteness),
2. $D(\mathbf{X}\|\mathbf{W}) = D(\mathbf{W}\|\mathbf{X})$ (symmetry),
3. $D(\mathbf{X}\|\mathbf{Z}) \leq D(\mathbf{X}\|\mathbf{W}) + D(\mathbf{W}\|\mathbf{Z})$ (triangle inequality).

However, if only condition 1 is satisfied by a particular distance measure, it is not a metric but it is referred to as a divergence. Assuming that spectra are positive functions, which is typically the case, divergences could be applied as dissimilarity measures to compare different spectral signatures. Common divergences are for example Kullback-Leibler, Hellinger, or Jensen-Shannon.

As mentioned before, the concept of divergences is on no account restricted to the Euclidean space. A more general approach is to consider an abstract space where common divergences, such as Kullback-Leibler-, Csiszár-Morimoto-, or Bregman-divergence can be generalized towards Alpha-, Beta-, and Gamma-divergences. The fundamental properties of the underlying divergences remain present. Particularly the Gamma-divergence seems to be very robust in terms of outliers [3]. Moreover, novel divergences offering tailored properties can be developed [1].

The properties of divergences are typically controlled by parameters. This tuning can be done in an elegant way by integration of divergences as dissimilarity measures or cost functions into machine learning approaches. In return, these machine learning approaches benefit from an extended choice of of available dissimilarity measures and cost functions. As mentioned before, prototype- and vector-quantization-based paradigms are particularly suitable [13, 8].

4 Summary and Outlook

The utilization of 'non-standard' dissimilarity measures and cost functions paves the way for extended features of clustering and classification/regression tools. Moreover, there is even a chance to develop application-specific distance measures that respect relevant properties of the underlying data and application, respectively. On the other hand, there is a strong demand for application-specific distance measures in the field of analysis of hyperspectral image signatures in general as well as hyperspectral close-range plant leaf images in particular.

This paper presented a work-in-progress framework of Neural Gas clustering, as representative of the class of prototype-based cluster algorithms, applied to crop plant phenotyping by means of hyperspectral imaging. The biological data was taken from a series of nutrition experiments.

In principle, three physical phenomena in terms of hyperspectral image acquisition need to be addressed within this framework:

1. Inhomogeneous exposure,
2. Inhomogeneous spectral energy distribution,
3. Inhomogeneous absorption patterns.

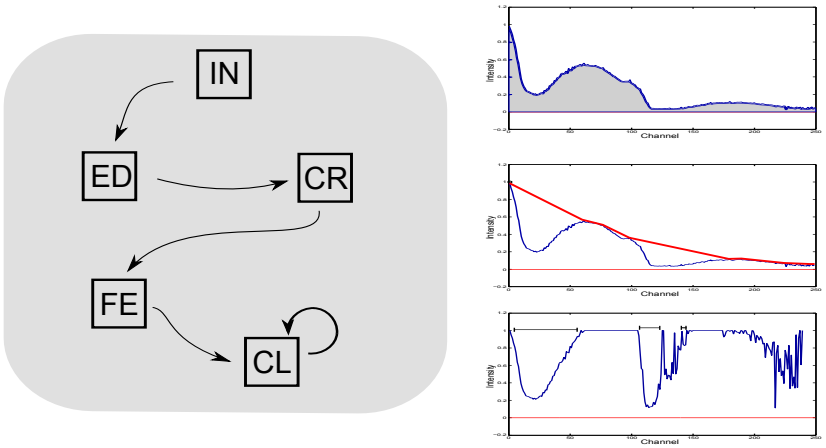


Fig. 3. Concept of integrating required individual hyperspectral data processing modules into a compact mathematical frame. Left panel: Sequence of separate processing modules *Image Normalization* (IN), *Envelope Design* (ED), *Continuum Removal* (CR), *Feature Extraction* (FE), and *Clustering* (CL) that are necessary to process hyperspectral image data and obtain suitable data clusters. These modules can be integrated into a self-contained system that is predicated on machine learning based clustering equipped with application specific dissimilarity measures (gray bounding box). Right panel: Pictographic illustration of the three major processing levels that address physical phenomena of inhomogeneous exposure, spectral energy distribution, and absorption patterns.

These are currently met by a sequence of individual processing modules that are designed heuristically (in terms of clustering / classification) or are at least very application-specific. The derived concept of integrating these modules into a compact mathematical frame as summarized in Fig. 3 is highly desirable. Along with the introduced concept of density-based divergence measures we see strong perspectives for both the development of an adapted mathematical theory and challenging applications in hyperspectral imaging at close-range level.

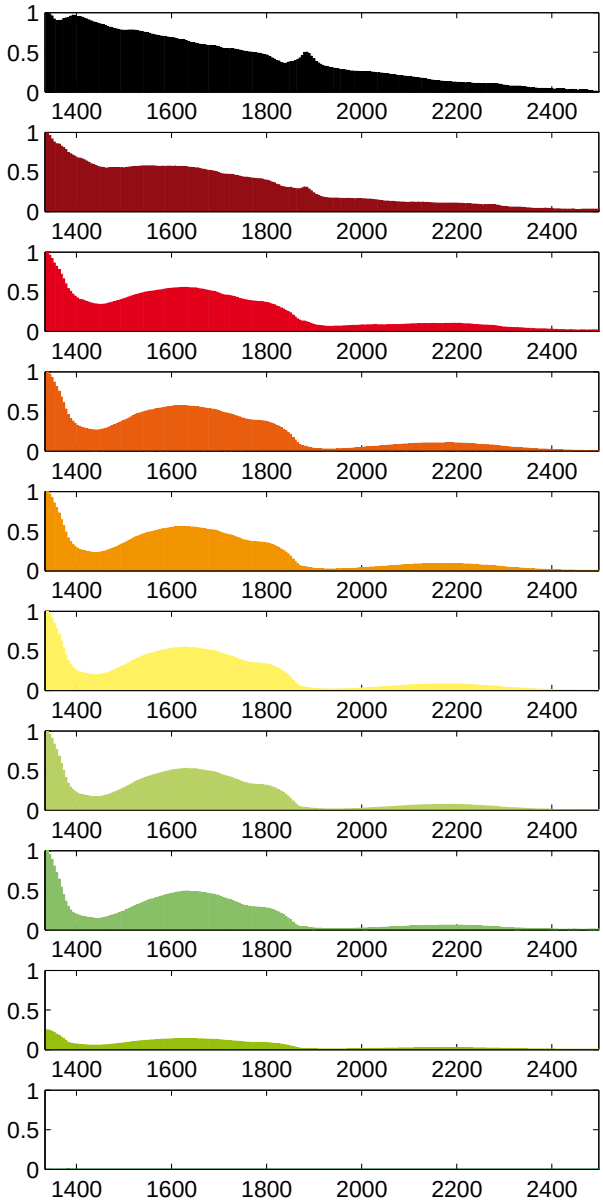


Fig. 4. Result of the Neural Gas training: normalized weight vectors of 10 neural prototypes, sorted according to their similarity (top to bottom), representing the centers of the 10 clusters. These prototypes also represent characteristic hyperspectral fingerprints formed from the data set during network training (abscissa shows the wavelength in nm). Figure borrowed from [9].

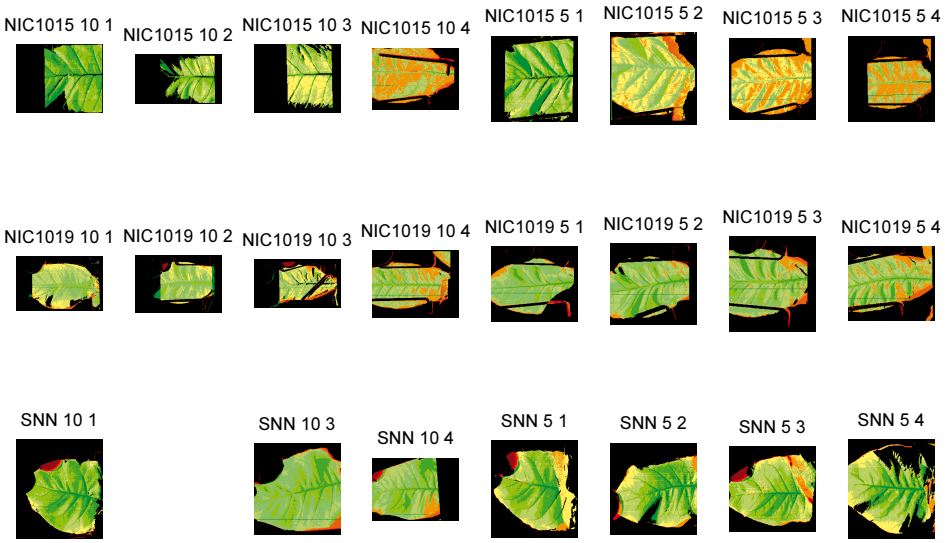


Fig. 5. Some sample images of the underlying nutrition experiment. The color code of each pixel indicates its membership of a particular cluster according to Fig. 4. The identifier of each subfigure contains the genotype (e.g. NIC1015, SNN) in separate rows, the ammonium nitrate (NH_4NO_3) nutrition treatment 10 mM (left figure half) vs. 5 mM (right figure half), and the position of the leaf along the plant stem (1 to 4 shown here). The background and some metal clips to fix the leaves on the image acquisition stage can be clearly seen in black. Green clusters generally indicate healthy parts of the leaf with a gradient from high content of ammonium nitrate mainly located in the leaf veins (dark green) to lower content (light green) in the leaf marrow. Yellow up to orange clusters are mainly located near the leaf edges and indicate a rather low content of ammonium nitrate due to beginning senescence. The blank subfigure in the bottom row is due to a highly degraded leaf at this particular position of this plant which could not be measured.

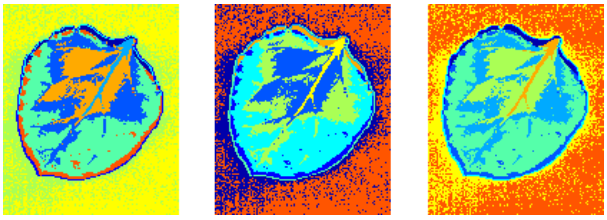


Fig. 6. Advanced dissimilarity measure: A Wavelet decomposition has been combined with standard metrics, Euclidean, correlation, and cosine (from left to right), in a ten-cluster-setting. It can be observed that leaf-specific tissue structures are better clustered than in Fig. 2.

Acknowledgements

The authors gratefully acknowledge scientific and technical support from Hans-Peter Mock and Andrea Matros, Leibniz-Institute of Plant Genetics and Crop Plant Research Gatersleben, Germany; Joachim Schiemann, Julius-Kühn Institute Quedlinburg, Germany; as well as technical support from Lars O. Lierstuen, Norsk Elektro Optikk A/S, Lørenskog, Norway.

References

1. Cichocki, A., Zdunek, R., Phan, A.H., Amari, S.I.: Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-way Data Analysis and Blind Source Separation. John Wiley and Sons, Ltd (2009)
2. Du, Y., Chang, C.I., Ren, H., Chang, C.C., Jensen, J.O., D'Amico, F.M.: New hyperspectral discrimination measure for spectral characterization. Optical Engineering 43(8), 1777–1786 (2004)
3. Fujisawa, H., Eguchi, S.: Robust parameter estimation with a small bias against heavy contamination. Journal of Multivariate Analysis 99(9), 2053 – 2081 (2008)
4. Kwon, H., Der, S.Z., Nasrabadi, N.M.: Unsupervised segmentation algorithm based on an iterative spectral dissimilarity measure for hyperspectral imagery. vol. 4310, pp. 144–152. SPIE (2000)
5. Lippmann, R.P.: An introduction to computing with neural nets. IEEE ASSP Magazine 4(87), 4–23 (1987)
6. Martinetz, T., Schulten, K.: A Neural Gas learns topologies. In: Kohonen, T., Mäkisara, K., Simula, O., Kangas, J. (eds.) Artificial Neural Networks, pp. 397–402. North-Holland, Amsterdam (1991)
7. Okamoto, H., Murata, T., Kataoka, T., Hata, S.I.: Plant classification for weed detection using hyperspectral imaging with wavelet analysis. Weed Biology and Management 7(1), 31–37 (2007)
8. Park, D.C., Nguyen, C., Lee, Y.: Content-based classification of images using centroid neural network with divergence measure. In: AI 2006: Advances in Artificial Intelligence, Lecture Notes in Computer Science, vol. 4304, pp. 729–738. Springer-Verlag, Berlin (2006)
9. Seiffert, U., Bollenbeck, F., Mock, H.P., Matros, A.: Clustering of crop phenotypes by means of hyperspectral signatures using artificial neural networks. In: Proceedings of the 2nd IEEE Workshop on Hyperspectral Imaging and Signal Processing: Evolution in Remote Sensing WHISPERS 2010, pp. 31–34. IEEE Press (2010)
10. Severson, R.F., Arrendale, R.F., Chortyk, O.T., Johnson, A.W., Jackson, D.M., Gwynn, G.R., Chaplin, J.F., Stephenson, M.G.: Quantitation of the major cuticular components from green leaf of different tobacco types. Journal of Agriculture and Food Chemistry 32, 566–570 (1984)
11. Vigneau, N., Rabatel, G., Roumet, P., Ecarnot, M.: Calibration of a chemometrical model from field hyperspectral close-range images: Taking into account leaf inclination and multiple reflection effects. In: Proceedings of the 2nd IEEE Workshop on Hyperspectral Imaging and Signal Processing: Evolution in Remote Sensing WHISPERS 2010, pp. 23–26. IEEE Press (2010)
12. Villmann, T.: Utilization of divergences to compare density functions (June 2010), personal communication on the 2nd Mittweida Workshop on Computational Intelligence MiWoCi 2010

13. Villmann, T., Haase, S., Schleif, F.M., Hammer, B.: Divergence based online learning in vector quantization. In: Artificial Intelligence and Soft Computing, Lecture Notes in Computer Science, vol. 6113, pp. 479–486. Springer-Verlag, Berlin (2010)
14. Villmann, T., Merényi, E., Hammer, B.: Neural maps in remote sensing image analysis. Neural Networks 16(3–4), 389–403 (2003)
15. Villmann, T., Merényi, E., Seiffert, U.: Machine learning approaches and pattern recognition for spectral data. In: Verleysen, M. (ed.) Proceedings of the 16th European Symposium on Artificial Neural Networks ESANN 2008. pp. 433–444. D-Side Publications, Evere, Belgium (2008)
16. Woolley, J.T.: Reflectance and transmittance of light by leaves. Plant Physiology 47, 656–662 (1971)

Interpretive Risk Assessment on GWA Data with Sparse Linear Regression

Ingrid Brænne, Kai Labusch, Thomas Martinetz and Amir Madany Mamlouk

Institute for Neuro and Bioinformatics, University of Luebeck
braenne@inb.uni-luebeck.de

Abstract. Genome-wide association (GWA) studies provide large amounts of high-dimensional data. GWA studies aim to identify variables, i.e., single nucleotide polymorphisms (SNP) that increase the risk for a given phenotype and have been successful in identifying susceptibility loci for several complex diseases. A remaining challenge is however to predict the individual risk based on the genetic pattern. Counting the number of unfavorable alleles is a standard approach to estimate the risk of a disease. However this approach limits the risk prediction by only allowing for a subset of predefined SNPs. Recent studies that apply SVM-learning have been successful in improving the risk prediction for Type I and II diabetes. However, a drawback of the SVM is the poor interpretability of the classifier. The aim is thus to classify based on only a small number of SNPs in order to also allow for a genetic interpretability of the resulting classifier. In this work we propose an algorithm that can do exactly this. We use an approximation method for sparse linear regression problems that has been recently proposed and can be applied to large data sets in order to search for the best sparse risk predicting pattern among the complete set of SNPs.

1 Introduction

There are three general aims of genome-wide association (GWA) studies: identifying genetic loci or patterns associated with common complex multifactorial diseases; understanding the complex genetic mechanisms underlying the disease; and predicting the individual risk of the disease based on the genetic patterns. In the past decade, GWA studies have been successfully employed to identify genetic loci (SNPs) associated to common complex diseases such as diabetes [13], myocardial infarction [11, 4], and Crohn's disease [10]. However, so far these findings have only limited impact on risk assessment and clinical treatment [6, 5]. A reason for the lack of feasible risk prediction by means of genetic variants can be explained by the fact that a disease effect may only come about through the interaction of multiple loci. Studies that focus on single locus effects alone are thus not likely to reveal the more complex genetic mechanisms underlying multifactorial traits [13, 12, 7]. To understand the underlying genetics of a disease, it might be feasible to identify sparse patterns that influence the risk. Such patterns can be helpful not only to assess the risk, but also to detect

possible genetic interactions. However, it is not straightforward to identify multiple SNPs that together increase the risk. Testing all possible combinations is not possible due to the typically enormous amount of SNPs in a GWA study.

The standard method for computing a genotype score (GS) that is used in order to predict the individual risk, is to count the number of unfavorable alleles (those associated to the disease) [3, 2, 1]. However, the drawback of this approach is rather obvious: if we only account for SNPs that are directly associated to the disease we will not allow for interactions between SNPs without an association. Validated susceptibility loci only explain a small proportion of the genetic risk and thus by only accounting for these it is not likely to gain a significant increase in risk prediction [6, 13]. Thus, a major improvement in risk prediction can only be achieved by training a multivariate classifier taking all SNPs into account. Since we expect the SNPs to have different influence on the risk, we need to apply weights to the SNPs in the classifier which again is not straightforward.

A classical tool for classification accounting for all features is the support vector machine (SVM) [15, 16]. The advantage of the SVM is that it is applicable to very large datasets and has already been applied successfully on GWA data [13, 17, 18]. However, the disadvantage of the SVM when it comes to GWA data is that the learned classifier is based on all SNPs, which makes it difficult to interpret the resulting classifier in a biological context. Thus, since we aim to classify based only on a small number of SNPs, we need to apply an appropriate SNP selection in advance and train the SVM only on these. A standard approach is to select the SNPs based on single significance values (p-values). However, this leads to the same issues as described for the GS approach except that a weighting of the SNPs is done by the SVM. Thus, if we aim to predict the risk of an individual as well as understanding the complex genetic mechanisms underlying the disease, the SVM might not be the appropriate choice.

In this work, we propose a novel method for GWA analysis that searches for a sparse risk predictor on the complete data. Predicting the phenotype from the genotype is approached in the framework of sparse linear regression, i.e., our method determines a set of weights that generate the phenotype as a sparse linear combination of the genotype.

2 Data & Methods

2.1 Data

We simulated case/control datasets using the PLINK software [8, 9] with the `simulate` option. We simulated a dataset of 5000 cases and 5000 controls. 100 SNPs of a total of 10100 SNPs where associated to the disease phenotype. Common complex diseases have typically low effect size [6], hence we simulated the effect size i.e. odds ratio (OR) of 1.3 and 1.6 for heterozygous and a multiplicative risk of 2.6 and 3.2 respectively for the homozygous. As previously described most of the identified genetic variants have only limited impact on risk assessment and clinical treatment. This missing heritability might be caused

by the fact that the main contribution result from variants with low minor allele frequency (MAF) and such variants are difficult to detect [6]. Thus, we simulated datasets with MAFs ranging from 0.05 to 0.1 and 0.1 to 0.2. Varying the described parameters we gain a total of 4 different datasets as shown in Table 2.1.

	MAF_{min}	MAF_{max}	OR
dataset type 1	0.05	0.1	1.3
dataset type 2	0.1	0.2	1.3
dataset type 3	0.05	0.1	1.6
dataset type 4	0.1	0.2	1.6

Table 1. Simulated datasets obtained with the PLINK software.

We obtain a genotype matrix G with $G = (\mathbf{g}_1, ..., \mathbf{g}_L)^T$,$\mathbf{g}_i \in \mathbb{R}^d$ where L is the number of individuals (10000) and d the number of SNPs (10100). Furthermore, we have phenotype labels $\mathbf{p} = (p_1, .., p_L)$, $\mathbf{p} \in \mathbb{R}^L$, $p_i \in \{-1, 1\}$. For the cases $p_i = 1$ holds whereas for the controls we have $p_i = -1$. We divided the dataset into two sets of equal size, one set for training and the other one for testing.

2.2 Genotype Score

A genotype score (GS) is calculated on the basis of the number of risk alleles (those associated with the disease phenotype) that are carried by each individual for a predefined subset of SNPs S. The subset S is selected according to their p-values estimated with the Chi-square statistics. The number of selected SNPs, i.e. $|S| = k$, is varied from 1 to 20. For the genotype data $G_{ij} \in \{0, 1, 2\}$ holds. The encoding is performed such that G_{ij} corresponds to the number of risk alleles that are carried by individual i at SNP location j. The homozygous genotype for the risk allele is coded with 2, heterozygous with 1 and homozygous for the non risk allele with 0. Hence, the GS for an individual i of the test set is computed by

$$R_i = \sum_{j \in S} G_{ij} \, . \tag{1}$$

2.3 Support Vector Machine

A support vector machine (SVM) determines the hyperplane that separates two given classes with maximum margin [19]. It has been applied to a broad range of classification problems and is among the methods that are used as benchmark in many cases. In order to measure the classification performance that is obtained using the set of SNPs S (selected as for the GS approach), we train a Gaussian-kernel SVM on the genotype data of the selected SNPs of the

training set. The hyper-parameters, i.e., kernel width and softness of the margin are adjusted by 10-fold cross-validation on the training set.

2.4 Sparse Linear Regression

In contrast to the GS and SVM approaches, in the sparse linear regression (SLR) approach the set of selected SNPs is not obtained from the p-values of the chi-square statistics but the selection of a predefined number of SNPs is performed automatically by the method. We consider the following optimization problem

$$\mathbf{w}_{SP} = \arg\min_{\mathbf{w}} \|\mathbf{p} - G\mathbf{w}\| \quad \text{subject to} \|\mathbf{w}\|_0 = k \; . \tag{2}$$

We are looking for a weight vector $\mathbf{w}$ that approximates the phenotype vector $\mathbf{p}$ as a linear combination of the SNPs G where the number of non-zero entries of the weight vector is equal to k[1]. It has been shown that (2) is a NP-hard combinatorial problem. A number of approximation methods such as Optimized Orthogonal Matching Pursuit (OOMP) [20] or Basis Pursuit [21] have been proposed that provide close to optimal solutions in benign cases [22]. The bag of pursuits method (BOP) can also be applied to this optimization problem. It is derived from the OOMP and performs a tree-like search that employs a set of optimized orthogonal matching pursuits [14]. The larger the number of pursuits used in the search is, the closer BOP approximates $\mathbf{w}_{SP}$. In this work the number of pursuits was set to 100. In contrast to BP, it does not lead to a quadratic optimization problem which might become computational very demanding since nowadays very large genotype data is available ($\approx 10^4$ individuals, $\approx 10^6$ SNPs). Let $\mathbf{w}_{BOP}$ be the approximation of $\mathbf{w}_{SP}$ that has been determined using the BOP method on the training data. The decision value of a given test individual $\mathbf{g}_i$ is obtained as $d_i = \mathbf{w}_{BOP}^T \mathbf{g}_i$. Again, the number of selected SNPs, i.e., k, is varied from 1 to 20.

3 Results

We trained and tested the GS, SVM, and SLR with varying numbers of selected SNPs. We evaluated the performance of the three algorithms by means of the receiver operator characteristic (ROC) obtained on the test set. The area under this curve (AUC) is a commonly used approach to evaluate the performance of a binary classifier. We generated a total of 5 random datasets for each choice of the simulation parameters, i.e., odds ratio (OR) and minor allele frequency (MAF) as described in section 2.1. Then, for a varying number of selected SNPs, i.e., $k = 1, \ldots, 20$, we evaluated the classification performance of the three approaches by means of their mean AUC for each of the data types respectively.

[1] $\|\mathbf{w}\|_0$ is the number of non-zeros in $\mathbf{w}$

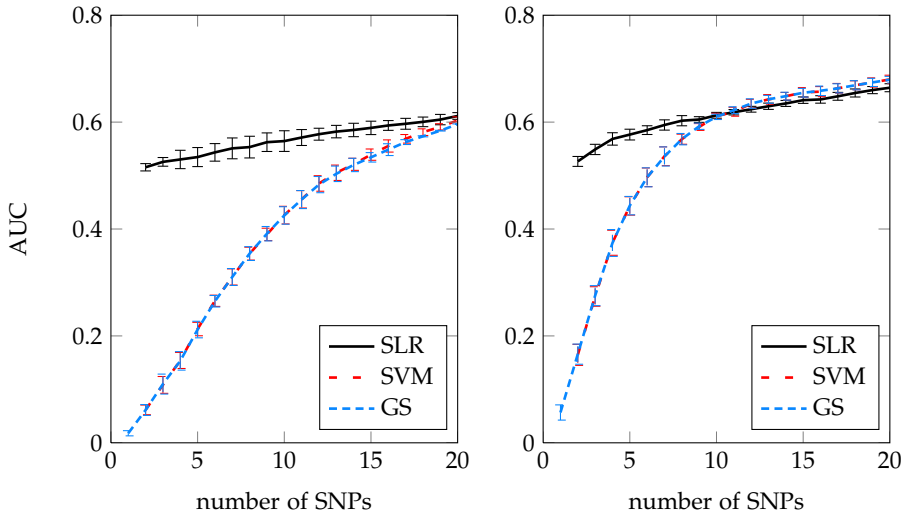


Fig. 1. OR 1.3, left: MAF 0.05-0.1, right: MAF 0.1-0.2

3.1 Dataset 1 and 2: Odds Ratio 1.3

We first compared the three algorithms on datasets with an OR of 1.3. In order to explore the effects of a higher OR on the performance of the algorithms, we then tested the three approaches on datasets with an OR of 1.6.

MAF: 0.05-0.1: As shown in Figure 1(left) the SLR approach clearly outperforms the traditional genotype score and the SVM for small numbers of selected SNPs. The performance of all three approaches improves with increasing numbers of selected SNPs and is almost the same for large numbers of selected SNPs. The performance of the GS and the SVM is very similar due to the selection of the SNPs based on the p-values that is performed for these methods. The SVM only marginally improves the performance for larger numbers of SNPs compared to the GS approach. In particular for small numbers of selected SNPs almost the same performance as for the GS is obtained.
MAF: 0.1-0.2: Figure 1(right) shows the performance of the methods on datasets with a MAF between 0.1 and 0.2. The results are qualitatively the same. However, if the number of selected SNPs is very large, the SVM and GS achieve better results than the SLR method.

3.2 Dataset 3 and 4: Odds Ratio 1.6

The performance of all three approaches on datasets with an OR of 1.6 improves compared to datasets having an OR of 1.3. This is not surprising since a higher OR implies that each single significant SNP is a stronger classifier.

MAF: 0.05-0.1: As for the results that were obtained from datasets with an OR of 1.3, the performance of the SLR method is best for small amounts of SNPs (see Figure 2,left). However, in contrast to the previous results, the GS and SVM approaches close up to the SLR and even become slightly better for larger numbers of SNPs.
MAF:0.1-0.2: Compared to the results on the dataset with a MAF of 0.05 -0.1, the performance only improves marginally as shown in Figure 2(right). In contrast to the results that were obtained on datasets with an OR of 1.3 the SLR method is not beaten by the two other approaches but rather equally good for large numbers of SNPs.

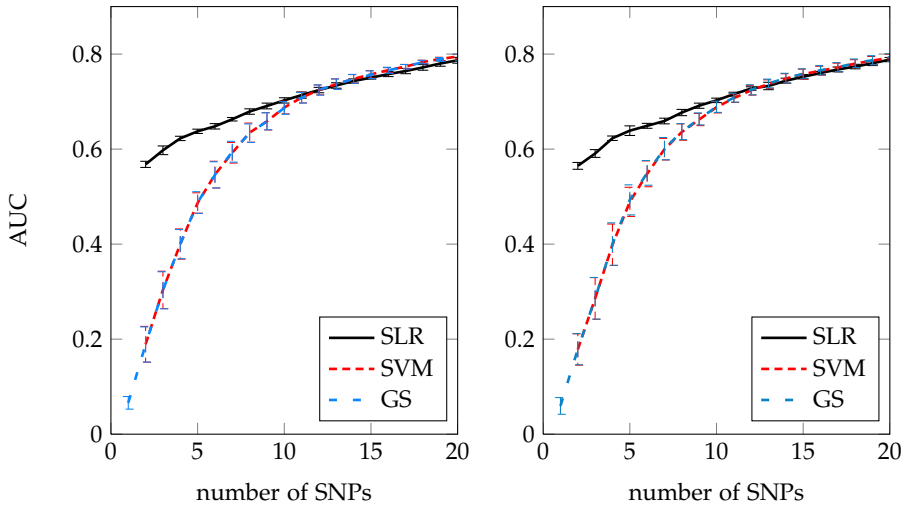


Fig. 2. OR 1.6, left: MAF 0.05-0.1, right: MAF 0.1-0.2

4 Conclusion

The aim of GWA studies is not only to perform risk predictions based on the genetic patterns but also to identify a small set of susceptibility loci in order to understand the genetic mechanisms underlying the disease. The set of susceptibility loci should be as small as possible in order to enable an analysis of the biological mechanisms that correspond to the loci that have been selected. The standard genotype score (GS) that counts the number of unfavorable alleles limits the risk prediction to be based only on SNPs associated to the disease, i.e., that are significant according to their p-value. Since we do not expect that a small number of SNPs that have been selected according to their p-values

can explain more than a small proportion of the genetic risk this approach is not sufficient. The risk prediction can be greatly improved by employing more powerful methods such as SVM-learning. However, for better interpretability of the classifier to understand the genetic mechanisms of the disease, we want to classify using only a small number of SNPs. Therefore, we again have to select SNPs on the basis of prior knowledge. In this work we ranked the SNPs by the p-values and trained the SVM on the best ranked SNPs which is a standard approach in GWA data analysis but often leads to weak performance if the number of selected SNPs is small.

In this paper, we approached the selection problem in the framework of sparse linear regression (SLR). The advantage of the SLR approach is that it does not need any a priori assumptions and thus does not have any limitations due to a preselection step. We applied the bag of pursuits method to the SLR problem which is possible even on huge data sets. We compared the three methods GS, SVM and SLR on GWA data that has been simulated using the PLINK software. If the set of selected SNPs is small, the SLR approach clearly outperforms SVM and classical GS approaches. Even though the performance of the other methods comes close to the performance of SLR if the number of SNPs is large enough, the presented results suggest that SLR should be the method of choice, in particular if the aim is not only to assess the risk of a disease but also to better understand the genetic mechanisms underlying the disease.

Acknowledgments. This work was supported by the Graduate School for Computing in Medicine and Life Sciences funded by Germany's Excellence Initiative [DFG GSC 235/1]

References

1. Sekar Kathiresan, Olle Melander, Dragi Anevski, Candace Guiducci, et al. Polymorphisms Associated with Cholesterol and Risk of Cardiovascular Events. *N Engl J Med*, 358(12):1240–1249, 2008.
2. Jianhua Zhao, Jonathan P. Bradfield, Mingyao Li, Kai Wang, et al. The role of obesity-associated loci identified in genome wide association studies in the determination of pediatric BMI. *Obesity (Silver Spring, Md.)*, 17(12):2254–2257, December 2009. PMID: 19478790 PMCID: 2860782.
3. Jianhua Zhao, Mingyao Li, Jonathan P Bradfield, Haitao Zhang, et al. The role of height-associated loci identified in genome wide association studies in the determination of pediatric stature. 11:96–96. PMID: 20546612 PMCID: 2894790.
4. Jeanette Erdmann, Anika Großhennig, Peter S. Braund, Inke R. König, et al. New susceptibility locus for coronary artery disease on chromosome 3q22.3. *Nat Genet*, 41(3):280–282, Mar 2009.
5. John P.A. Ioannidis. Prediction of Cardiovascular Disease Outcomes and Established Cardiovascular Risk Factors by Genome-Wide Association Markers. *Circ Cardiovasc Genet*, 2(1):7–15, February 2009.
6. Teri A. Manolio, Francis S. Collins, Nancy J. Cox, David B. Goldstein, et al. Finding the missing heritability of complex diseases. *Nature*, 461(7265):747–753, October 2009.

7. J. H. Moore. The ubiquitous nature of epistasis in determining susceptibility to common human diseases. *Human heredity*, 56(1-3):73–82, 2003.
8. Shaun Purcell, Benjamin Neale, Kathe Todd-Brown, Lori Thomas, et al. PLINK: a tool set for whole-genome association and population-based linkage analyses. *American Journal of Human Genetics*, 81(3):559–575, September 2007. PMID: 17701901.
9. Shaun Purcell. http://pngu.mgh.harvard.edu/purcell/plink/
10. J. V. Raelson, R. D. Little, A. Ruether, H. Fournier, B. Paquin, P. Van Eerdewegh, W. E. Bradley, et al. Genome-wide association study for Crohn's disease in the Quebec Founder Population identifies multiple validated disease loci. *Proc Natl Acad Sci U S A*, 104(37):14747–14752, 2007.
11. Nilesh J. Samani, Jeanette Erdmann, Alistair S. Hall, , Christian Hengstenberg, et al. Genomewide Association Analysis of Coronary Artery Disease. *N Engl J Med*, 357(5):443–453, 2007.
12. Naomi R Wray, Michael E Goddard, and Peter M Visscher. Prediction of individual genetic risk of complex disease. *Current Opinion in Genetics and Development*, 18(73):257–263, 2008.
13. Zhi Wei, Kai Wang, Hui-Qi Q. Qu, Haitao Zhang, et al. From disease association to risk assessment: an optimistic view from genome-wide association studies on type 1 diabetes. *PLoS genetics*, 5(10):e1000678+, October 2009.
14. Kai Labusch and Thomas Martinetz. Learning Sparse Codes for Image Reconstruction. In Michel Verleysen, editor, *Proceedings of the 18th European Symposium on Artificial Neural Networks*, pages 241–246. d-side, 2010.
15. Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3:1157–1182, 2003.
16. Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene Selection for Cancer Classification using Support Vector Machines. *Machine Learning*, 46:389–422, 2002.
17. Y. Yoon, J. Song, S. H. Hong, and J. Q. Kim. Analysis of multiple single nucleotide polymorphisms of candidate genes related to coronary heart disease susceptibility by using support vector machines. *Clin Chem Lab Med*, 41(4):529–534, 2003.
18. H. J. Ban, J. Y. Heo, K. S. Oh, and K.J. Park. Identification of Type 2 Diabetes-associated combination of SNPs using Support Vector Machine. *BMC Genet.*, 11(1):26, 2010.
19. V. N. Vapnik. Statistical Learning Theory *Wiley*,1998.
20. L. Rebollo-Neira and D. Lowe. Optimized orthogonal matching pursuit approach *IEEE Signal Processing Letters*, 9(4):137-140, 2002.
21. S. S. Chen, D. L. Donoho and M. A. Saunders. Atomic Decomposition by Basis Pursuit. *SIAM Journal on Scientific Computing*, 20(1):33-61, 1998.
22. J. A. Tropp. Greed is good: algorithmic results for sparse approximation. *IEEE Transactions on Information Theory*, 50(10):2231-2242, 2004.

Ensemble methods for probability density estimation

Michael Glodek, Martin Schels, and Friedhelm Schwenker

Ulm University, Institute of Neural Information Processing
firstname.lastname@uni-ulm.de

Abstract. Estimating of probability density functions is one of the most essential techniques in machine learning and pattern recognition. In particular the Gaussian mixture model (GMM) is often used for a parametric representation of complex densities. Most of the parameters of a GMM can be adjusted using the expectation maximization (EM) algorithm. The amount of mixtures and the form of the covariance matrix however cannot be derived by the EM algorithm and therefore need to be chosen by a model selection method, which in general searches the parameter space. The presented work aims on the extension of the existing EM algorithm in order to increase the robustness and accuracy of density estimation. The basic ideas rely on the utilization of an ensemble of GMM. This can be related to model averaging: while the standard model selection method determines the best performing GMM, the ensemble approach uses a subset of best GMM which are subsequently combined such that the precision and stability of the estimated density is improved. The studies in this paper yield to first promising results which will be presented along with a detailed description of the complete approach.

1 Introduction

Estimating density is a frequent task in pattern recognition and part of many complex algorithms like hidden Markov models or Bayesian inference in general. The estimation of continuous densities can be achieved using non-parametric methods (e. g. by using kernel density estimators or nearest neighbour methods) or parametric methods. Within the class of parametric methods, the Gaussian mixture model (GMM) has been proven to be a good representation of complex densities and therefore is broadly used in conjunction with the expectation maximization (EM) algorithm which was first exhaustively studied by Dempster et al. [3]. The combination of a model averaging method, namely bagging, in conjunction with a GMM estimated by the EM algorithm, was first investigated by Ormoneit et al. in [4, 2]. One limitation of the study of Ormoneit is that the number of mixtures components remains the same over all models averaged. A related idea of ensemble creation using GMM have been proposed by Shinozaki et al. in which a modification of the EM algorithm has been proposed [6]. Due to the fusion of information between the expectation and maximization steps, the number of mixtures components is as well fixed. The presented work suggests to

combine a set of GMM having different properties chosen with regard to gain a high diversity among each other. In addition to bagging, different number of mixture components as well as changing restrictions to the covariance matrix are proposed. To the authors knowledge a study on GMM ensembles following the same approach to the one presented has not yet been conducted.

In Section 2 probability density functions and the EM algorithm are reviewed and the ensemble algorithm is introduced. Section 3 presents the experiments performed. Finally Section 4 summarizes the work of this study and gives an outlook to future work.

2 Estimating the density by an ensemble of GMM

Within this section the basic concept of probability density and the approximation using the EM algorithm are recapitulated. Later on the ensemble GMM algorithm is proposed and discussed along with a demonstrating example.

2.1 Density estimation using GMM

A probability density function $p(\mathbf{x})$ is defined over a continuous variable $\mathbf{x}$ and describes the probability that $\mathbf{x}$ lies within the interval $\mathbf{a} \leq \mathbf{x} \leq \mathbf{b}$. The probability is therefore defined as $p(\mathbf{a} \leq \mathbf{x} \leq \mathbf{b}) = \int_{\mathbf{a}}^{\mathbf{b}} p(\mathbf{x})\,\mathrm{d}\mathbf{x}$. It must hold the properties to be non-negative and to integrate up to one over the whole space of $\mathbf{x}$. The density estimation aims on deriving the probability $p(\mathbf{x})$ based on a finite number of observations $\mathbf{x}_1, \ldots, \mathbf{x}_N$. It needs to be pointed out, that the problem itself is ill-posed for there are infinitely many probability distributions that could have give rise to the observed data [1].
The Gaussian mixture model (GMM) belongs to the class of parametric representations because the GMM restricts the form of the density by a set of parameters. It is defined to be a linear superposition of Gaussians in the form

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\mu_\mathbf{k}, \mathbf{\Sigma_k}).$$

Each of the K Gaussians $\mathcal{N}$ are indexed by k and expressed by the mean vector $\mu_\mathbf{k}$ and its corresponding covariance matrix $\mathbf{\Sigma_k}$. The weights of the Gaussians $\{\pi_k\}$ are defined to be within the interval of $[0, 1]$ and to sum up to one. The presented work will focus on the GMM and therefore give a short introduction to the EM algorithm which is broadly used to estimate $\mu_\mathbf{k}$, $\mathbf{\Sigma_k}$ and $\{\pi_k\}$. The EM algorithm governs its name by the expectation and the maximization step which correspond to the second and third item of the following algorithm:

1. Initialize the means $\mu_\mathbf{k}$, the covariance matrices $\mathbf{\Sigma_k}$ and the weights π_k in an appropriate manner using for example K-means clustering or random values. Furthermore, evaluate the initial value of the log likelihood.

2. Determine the conditional probability for each mixture component k to be responsible for the observation $\mathbf{x}_n$ by

$$P(k|\mathbf{x}_n) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\mu_k, \mathbf{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\mu_j, \mathbf{\Sigma}_j)}.$$

3. Calculate the new adapted parameters for the current responsibilities for each mixture component:

$$\begin{aligned}
\mu_{\mathbf{k}}^{\text{new}} &= \frac{1}{N_k} \sum_{n=1}^{N} P(k|\mathbf{x}_n)\mathbf{x}_n \\
\mathbf{\Sigma}_{\mathbf{k}}^{\text{new}} &= \frac{1}{N_k} \sum_{n=1}^{N} P(k|\mathbf{x}_n)(\mathbf{x}_n - \mu_k^{\text{new}})(\mathbf{x}_n - \mu_k^{\text{new}})^T \\
\pi_k^{\text{new}} &= \frac{N_k}{N}
\end{aligned}$$

   where $N_k = \sum_{n=1}^{N} P(k|\mathbf{x}_n)$.
4. Update the log likelihood

$$\ln p(\{\mathbf{x}_n\}|\{\mu_k\}, \{\mathbf{\Sigma}_k\}, \{\pi_k\}) = \sum_{n=1}^{N} \ln \{\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\mu_k, \mathbf{\Sigma}_k)\}$$

   and check for stopping criterions. If no stopping criterions are met return to step 2.

The EM algorithm suffers from the appearance of singularities which may occur, if one mixture component gets responsible for solely one observation point. This circumstance leads to an infinitesimal variance and a likelihood going to infinity. These singularities can be avoided by resetting the corresponding Gaussian using an heuristic or using a regularization term. Although the EM algorithm is capable to estimate the parameters $\{\pi_k\}$, $\{\mu_k\}$ and $\{\mathbf{\Sigma}_k\}$ some model parameters like the number of mixtures K or potential restrictions to the covariance matrix (e.g. spherical or diagonal) need to be defined by experts. A common approach to determine these parameters is model selection in which different configurations are explored and the best fitting model is selected at the end. This approach has the obvious drawback that most of the models are rejected although they potentially render a good approximation of the underlying density.

2.2 Constructing an ensemble of GMM

For GMM ensembles it will be convenient to apply a variety of GMM having a high diversity. Different GMM need to focus on different parts of the data set such that they complement one another when they are combined. This can be achieved using subsets of the trainings data, random initialization of weights,

mean and covariance, different numbers of mixture components and different restrictions to the covariance matrix.

The input of the ensemble algorithm is a set of observations $\mathbf{x}_1, \ldots, \mathbf{x}_N$ and a list L of GMM configurations. Furthermore, a number M representing how many GMM shall be used for the final combination of the ensemble and parameters for bagging (subset size and number of subsets to be created) may be passed. The list L contains the models to be evaluated and may have repeated entries of the same configuration. The elements are tuples of the number of mixture components K, the restriction of the covariance matrix (e. g. diagonal, spherical) and the initialization method for the EM algorithm. Utilizing random initialization of the GMM parameters may inject the diversity of the models. Once the algorithm has trained each GMM of the list, the best M GMM with respect to the likelihood are chosen and combined using a pointwise mean or a median. The averaging of GMM retains some properties of the individual GMM such that the resulting probability density will have only positive values and the integral over the whole space of $\mathbf{x}$ will still integrate to one. Averaging using weights may also be performed by evaluating the likelihood of the different GMM. It is also considerable, that singularities occurring due to the EM algorithm can be compensated by utilizing an alternative to averaging, e. g. median. While the median is robuster to outliers in the ensemble, it is only useful in case only the mode of the density is required in the subsequent processing due to the property of integrating up to one and being differentiable will no longer hold.
Depending on the quantity M the ensemble GMM may have a notedly increased complexity. In order to circumvent this complexity, it may also be convenient to approximate the ensemble GMM by a regression method for instance a neural network. The trainings data for the regression can easily be generated by the ensemble GMM itself.

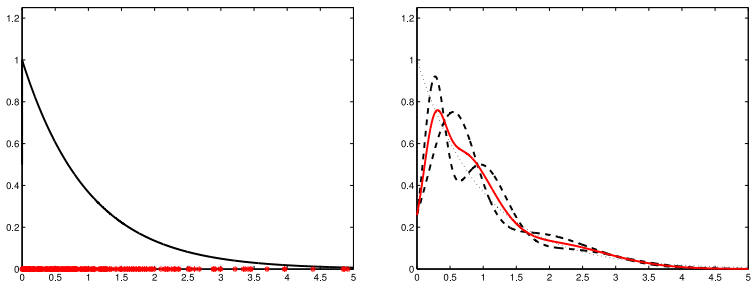


Fig. 1: Example of estimating the density of an exponential decay. The left plot shows the probability density (solid line) from which the data samples are derived (x-marks at the bottom of the plot). The data samples are used to estimate two Gaussian mixture models with two and three mixture components illustrated on the right-hand plot by the two dashed lines. While both models give a good approximation of the density, the average performs slightly better (The original function is indicated in the right plot by the dotted line).

The advantages of using ensemble GMM are demonstrated in Figure 1 with the help of a simple example. The density shown on the left-hand side of the figure is given by an exponential decay ($\lambda = 1$). The x-marks on the bottom of the same plot are the 250 data samples generated from the distribution used to train the GMM plotted on the right-hand side of the figure. A GMM having two mixture components and another one having three mixture components are illustrated on the corresponding plot by two dashed curves. The solid line corresponds to the average of both GMM. In order to compare the accuracies of the three densities, the integral of the absolute difference to the original density are taken over the interval $[0, 5]$. The GMM having two and three mixture components achieved an error of 0.29 and 0.22 respectively. The average of both functions obtained an error of 0.19 and therefore performs better. In order to gain more insight in the performance of the proposed approach, the next section will present and discuss experiments which have been conducted.

3 Experiments

This section presents two different studies conducted to investigate the benefits of ensemble GMM. The first study is a typical classification task which is based on the data set described and used by Ormoneit et al. [5]. It will support the argument that ensemble GMM gives an improved representation of the underlying density. The second study aims at the comparison of the classical approach and ensemble GMM with respect to its robustness. Both experiments have been conducted using artificial data and will be explained in detail within the following two subsection.

3.1 Ensemble GMM Classification Performance

In order to compare the ensemble GMM to the other extensions of the EM-algorithms, an artificial classification problem which was used by Ormoneit et al. is studied [4]. Ormoneit et al. proposed two data sets which can be generated using the algorithm described in the corresponding article. The first artificial problem utilizes 200 data points per class in a two-dimensional space which are equally divided into training and test data set. GMM bagging has been performed by Ormoneit et al. using 20 mixture components which are trained with 50 replications and resampling by drawing 70% subsamples from the original data set without replacement. Furthermore, two model penalization approaches have been evaluated which favor covariance matrix with a certain scale. The second artificial problem utilizes 400 data points per class in a ten-dimensional space which are as well equally divided into training and test data set. It differs from the first artificial problem in using a smaller offset which results in a larger overlap between the two classes. The same algorithms used in the first artificial problem have also been evaluated on the higher-dimensional data set. Unfortunately, the EM-algorithm used in this study was unable to estimate valid GMM without singularities in the second artificial example although only a much

smaller number of mixture component have been utilized. This might be related to the small training data set and the high dimensionality used. Hence, only a comparison using the first artificial problem can be rendered. The ensemble GMM was evaluated in two modi: using all configurations and using only the 10 best GMM according to the likelihood. The estimated numbers of mixture components have ranged from 5 to 15 with 2 replications and resampling by drawing 90% subsamples from the original data set without replacement. The results are averaged over 20 simulations and are solely based on the test data set. The classification performs is shown in Table 1 along with the variance over the 20 simulations. The accuracy achieved by Ormoneit et al. could be improved

Method	Artificial problem I
Max. Likelihood†	79.03%(2.6)
Simple averaging†	80.43%(2)
Subset averaging†	80.73%(1.8)
Bagging†	81.2%(1.4)
Penalized likelihood†	82.28%(1.4)
Bayesian†	82.7%(1.3)
Ensemble	84.27%(0.4)
Ensemble ($M = 10$)	84.17%(0.5)

Table 1: Comparsion to the first artificial data set which Ormoneit et al. proposed. Results marked by a dagger (†) indicate the best performance achieved within the study by Ormoneit et al. for the respective method. The method *Ensemble* combined all 20 GMM, while *Ensemble ($M = 10$)* combined only the ten best GMM.

by 1.5% while averaging fewer GMM than Ormoneit et al. required. The ensemble GMM utilizing only 10 GMM with the highest likelihood does not perform better than the complete ensemble GMM.

3.2 Ensemble GMM Performance depending on Density Complexity

It can be assumed that a classical GMM performs optimal if the underlying density consists out of a superposition of Gaussians distributions. This means as well that more complex densities which for instance have a super Gaussian or sub Gaussian shape cannot be properly represented by a GMM (e.g. the Student's t-distribution which is composed out of an infinite number of Gaussian distributions). In order to study GMM and ensemble GMM different types of densities ranging sub Gaussian to almost Gaussian shape are evaluated. Three beta distribution with hyperparameters $a = b = 1$, $a = b = 2$ and $a = b = 10$ have been estimated by classical GMM and ensemble GMM, see Figure 2. The beta distribution converges to a Gaussian distribution the higher the hyperparameters a and b are chosen.

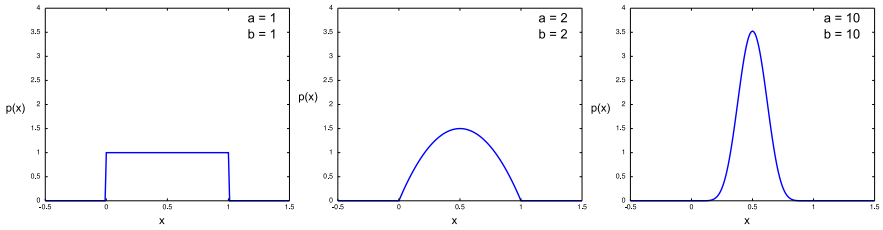


Fig. 2: Beta distribution used to study the accuracy and robustness of the classical and the ensemble GMM. The beta distribution takes the form of a uniform distribution in case the two parameters are set to $a = b = 1$ (left plot). Parameters set to $a = b = 2$ result in a concave shape with a rough change on the boundary of its support (middle plot). A shape close to a Gaussian distribution is obtained setting the parameters to $a = b = 10$ (right plot).

Beside the distribution to be estimated, a meaningful error rate shall as well be used to derive as much information from the study as possible. Therefore, the well-known bias-variance decomposition of the squared loss function with respect to models has been considered. The loss function is given by

$$E(L) = \sum_{n \in N} \{y(\mathbf{x}_n) - h(\mathbf{x}_n)\}^2$$

where the assumption has been made, that no noise is contained within the target function $h(\cdot)$. To get an impression of the uncertainty associated with one estimate, the average over one certain model $\mathcal{M}$ is introduced into the loss function by adding and substracting $\mathbb{E}_{\mathcal{M}}[y(\mathbf{x}; \mathcal{M})]$:

$$\begin{aligned} \{y(\mathbf{x}) - h(\mathbf{x})\}^2 &= \{y(\mathbf{x}; \mathcal{M}) - \mathbb{E}_{\mathcal{M}}[y(\mathbf{x}; \mathcal{M})] + \mathbb{E}_{\mathcal{M}}[y(\mathbf{x}; \mathcal{M})] - h(\mathbf{x})\}^2 \\ &= \{y(\mathbf{x}; \mathcal{M}) - \mathbb{E}_{\mathcal{M}}[y(\mathbf{x}; \mathcal{M})]\}^2 + \{\mathbb{E}_{\mathcal{M}}[y(\mathbf{x}; \mathcal{M})] - h(\mathbf{x})\}^2 \\ &\quad 2 \cdot \{y(\mathbf{x}; \mathcal{M}) - \mathbb{E}_{\mathcal{M}}[y(\mathbf{x}; \mathcal{M})]\}\{\mathbb{E}_{\mathcal{M}}[y(\mathbf{x}; \mathcal{M})] - h(\mathbf{x})\} \end{aligned}$$

Once the term has been expanded, the expectation of this expression is taken with the respect to $\mathcal{M}$. As a consequence the cross-term vanishes and we obtain

$$\mathbb{E}_{\mathcal{M}}[\{y(\mathbf{x}) - h(\mathbf{x})\}^2] = \{\mathbb{E}_{\mathcal{M}}[y(\mathbf{x}; \mathcal{M})] - h(\mathbf{x})\}^2 + \mathbb{E}_{\mathcal{M}}[\{y(\mathbf{x}; \mathcal{M}) - \mathbb{E}_{\mathcal{M}}[y(\mathbf{x}; \mathcal{M})]\}^2].$$

The first term, called the squared bias, represents the extent to which the average prediction differs from the desired density function. The second term which is called variance measures the extent to which the solution for individual models vary around their average [1].

As already mentioned this study compares the classical (single) GMM with ensemble GMM. For both 16 models have been evaluated for each beta distribution. In case of the classical GMM the models have only different number of mixture components, namely ranging from 1 to 16. For the ensemble GMM

an increasing complexity have been chosen. The amount of mixture components always ranging from one to the index of the current model, such that the ensembles $\{\{1\}, \{1,2\}, \{1,2,3\}, \ldots, \{1 \ldots 16\}\}$ are evaluated. The studied data distributions will clearly favor a simple GMM with a number of mixtures close to one. Furthermore it can be assumed that the classical GMM will perform notedly worse the more mixtures are used while the ensemble GMM will show a robuster behaviour with less variance. The error rates are shown in Figure 3 and based on equidistant data samples generated within the interval $[-0.5, 1.5]$ with a step size of 10^{-3}. For each value the model has been estimated 20 times to derive the squared bias and the variance. The results of the classical application of the GMM is illustrated in Figure 3a. Figure 3b shows the associated results for the ensemble GMM. The squared bias (dashed curve), which represents the average prediction difference from the desired density function, appears to be very stable for the different models. The variance (dotted curve) however has a notedly higher contribution to the overall test error (solid line) for the classical GMM and is steadily increasing. Note that the plots showing the Beta distributions with $a = b = 10$ have different scales on the y-axis.

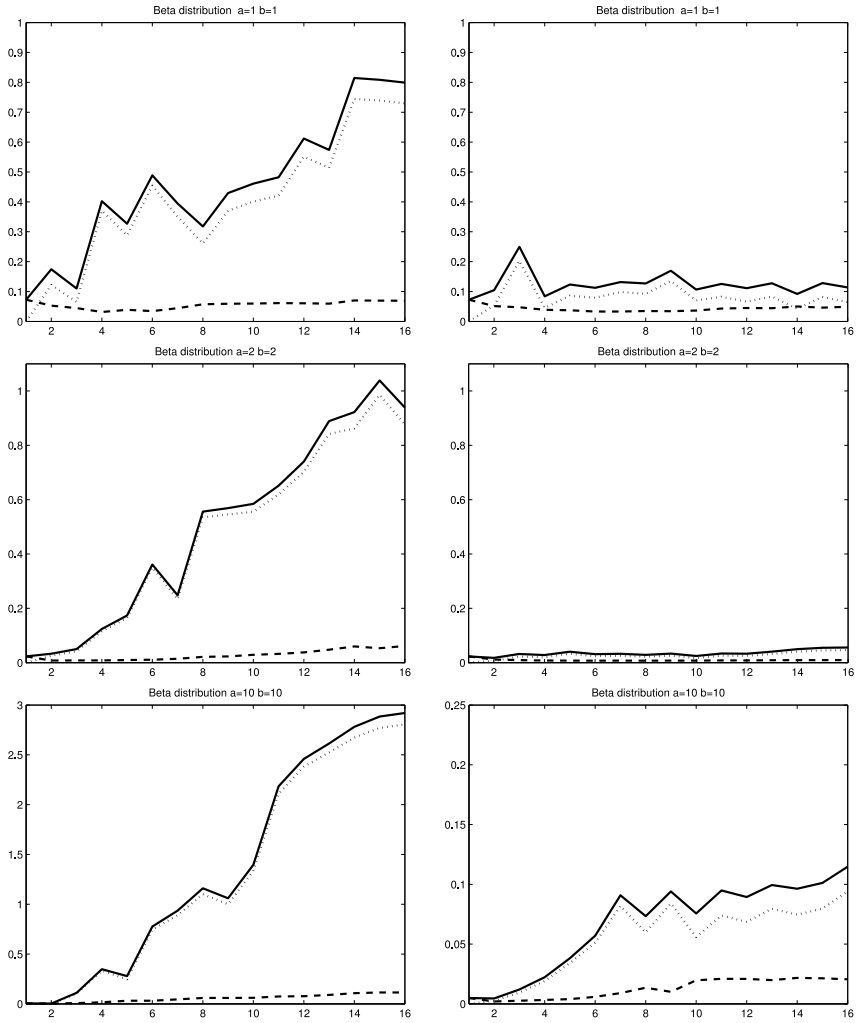
Like expected both models favor small number of mixtures components. Considering that the best performing number was not evident, the ensemble GMM achieved for all three beta distribution very robust results, which was intended to be shown.

4 Conclusion

Within this paper the extension of density estimation using ensemble GMM has been studied. The ensemble GMM combines a set of GMM being as diverse as possible such that they complement each other. A classical approach is bagging which subdivides the training data set into subsets, such that different models are obtained which are subsequently averaged. Ensemble GMM generalizes this approach by utilizing different number of mixture components and restriction to the covariance matrix. A first study demonstrated the performance achievement within a classification task. The ensemble GMM obtained better accuracies than the competing approaches. The second study proved that the robustness of ensemble GMM clearly outperforms comparable single GMM. Future work will aim on further studying the ensemble GMM approach and the extension of existing algorithms like HMM to improve their performance.

Acknowledgment

The presented work was developed within the Transregional Collaborative Research Centre SFB/TRR 62 “Companion-Technology for Cognitive Technical Systems” funded by the German Research Foundation (DFG).
The work of Martin Schels is supported by a scholarship of the Carl-Zeiss Foundation.



(a) Classical GMM. Number of mixture components are given on the x-axis.

(b) Ensemble GMM with increasing complexity. Number of mixture components are ranging from one to the given value on the x-axis.

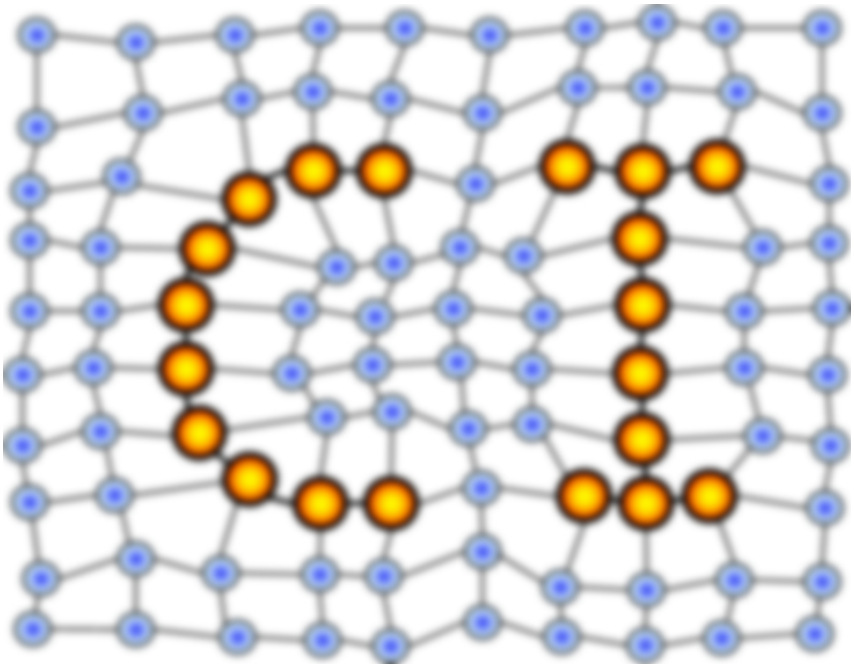
Fig. 3: Bias-variance decomposition for classical and ensemble GMM. Squared bias shown in dashed curves, variance shown in dotted curves and the sum of squared bias and variance (test error) shown in solid curves.

References

1. C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.
2. L. Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.
3. A.P. Dempster, N.M. Laird, D.B. Rubin, et al. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.
4. D. Ormoneit and V. Tresp. Improved Gaussian mixture density estimates using Bayesian penalty terms and network averaging. *Advances in neural information processing systems*, pages 542–548, 1996.
5. D. Ormoneit and V. Tresp. Averaging, maximum penalized likelihood and Bayesian estimation for improving Gaussian mixture probability density estimates. *IEEE Transactions on Neural Networks*, 9(4):639–650, 1998.
6. T. Shinozaki and T. Kawahara. GMM and HMM training by aggregated EM algorithm with increased ensemble sizes for robust parameter estimation. In *IEEE International Conference on Acoustics, Speech and Signal Processing, 2008. ICASSP 2008*, pages 4405–4408, 2008.

MACHINE LEARNING REPORTS

Report 04/2010



Impressum

Machine Learning Reports ISSN: 1865-3960

▽ Publisher/Editors

Prof. Dr. rer. nat. Thomas Villmann
University of Applied Sciences Mittweida
Technikumplatz 17, 09648 Mittweida, Germany
• http://www.mni.hs-mittweida.de/

Dr. rer. nat. Frank-Michael Schleif
University of Bielefeld
Universitätsstrasse 21-23, 33615 Bielefeld, Germany
• http://www.uni-bielefeld.de/

▽ Acknowledgments

We would like to thank the reviewers for their time and patience.