ESANN 2010 proceedings, European Symposium on Artificial Neural Networks - Computational Intelligence and Machine Learning. Bruges (Belgium), 28-30 April 2010, d-side publi., ISBN 2-930307-10-2.

Adaptive learning rate control for "neural gas principal component analysis"

Wolfram Schenck, Ralph Welsch, Alexander Kaiser, Ralf Möller [*]

Computer Engineering Group — Faculty of Technology
Bielefeld University — POB 100131, D-33501 Bielefeld — Germany
`wschenck(at)ti.uni-bielefeld.de`

Abstract. We propose a novel algorithm for adaptive learning rate control for Gaussian mixture models of the NGPCA type. The core idea is to introduce a unit–specific learning rate which is adjusted automatically depending on the match between the local principal component analysis of each unit (interpreted as Gaussian distribution) and the empirical distribution within the unit's data partition. In contrast to fixed annealing schemes for the learning rate, the novel algorithm is applicable to real online learning. Two experimental studies are presented which demonstrate this important property and the general performance of this algorithm.

1 Introduction

"Neural gas principal component analysis" (NGPCA) [4] belongs to the class of Gaussian mixture models which approximate data distributions by sets of multivariate Gaussian distributions. NGPCA is based on the vector quantization method "neural gas" (NG) [3]; instead of simple codebook vectors, local PCA ("principal component analysis") (e.g., [5]) units are distributed over the training data. Each PCA unit is adapted to its partition of the data. In this way, NGPCA can represent curved data distributions by combining many PCA units each of which performs local linear approximation and local dimensionality reduction.

NGPCA is an online learning method. In each training step, a data point $\mathbf{x}$ is presented to the network. Afterwards, the units are ordered according to the NG ranking scheme, and each unit's center, eigenvectors, and eigenvalues are adapted with a specific learning rate determined via the ranking process. For this reason, the PCA has to be carried out with online methods like robust recursive least squares (RRLSA) [5]. NG relies on two global parameters in every training step, the global learning rate ϵ and the neighborhood range ρ. These follow an annealing scheme and decay exponentially from initial values $\epsilon_{\max}$,$\rho_{\max}$ to final smaller values $\epsilon_{\min}$,$\rho_{\min}$ (for the last training step T_{final}). In this way, the training process allows first for quick adaptation and finally for fine–tuned (but slow) learning. However, fixed annealing like this cannot cope with non–stationary training data (for example in sensorimotor coordination tasks in adaptive robotics) once the learning rate has dropped to small values. This limits NGPCA (as presented in [4]) to applications with stationary data.

For this reason, we replaced the fixed annealing scheme for the learning rate ϵ and the neighborhood radius ρ by an adaptive process which adjusts both ϵ

[*]This work was supported by the German Science Foundation (DFG) under grants MO1037/1-2 and EXC277 (Center of Excellence "Cognitive Interaction Technologies").

and ρ depending on the match of the units to the underlying data distribution. Moreover, the learning rate ϵ becomes unit–specific. In this way, never-ending online learning becomes possible, even for non–stationary data distributions. In the following, this adaptive process is specified. Furthermore, we present the results from tests on synthetic data distributions and real world robotic data.

2 NGPCA algorithm

The training data for an NGPCA model consists of vectors $\mathbf{x}_t \in X \subset \mathbb{R}^n$. The model itself is composed from N local PCA units each of which is defined by a tupel $\{\mathbf{c}_i, \mathbf{W}_i, \mathbf{\Lambda}_i, \lambda_i^*\}$, with $i = 1, \ldots, N$. $\mathbf{W}_i$ contains the estimated eigenvectors in the m principal directions, $\mathbf{\Lambda}_i$ is a diagonal matrix with the corresponding eigenvalues $\lambda_{i,j}$ $(j = 1, \ldots, m)$. λ_i^* is the estimated residual variance for each of the $n - m$ minor directions. The codebook vector $\mathbf{c}_i \in \mathbb{R}^n$ finally defines the center of the PCA unit.

In each training step t, one vector $\mathbf{x}_t$ from the training set is drawn at random (in the following, the index t is omitted). For every unit, the volume–normalized[1] Mahalanobis distance d_i (incl. the reconstruction error) is determined (see Eqn. (3.10) in [2]), yielding the vector $\mathbf{d}$ of all distances $d_i(\mathbf{x})$. A rank $r_i(\mathbf{d}) = 0, ..., N-1$ is assigned to each unit: A rank of 0 indicates the closest and a rank of $N - 1$ the largest distance to the vector $\mathbf{x}$. After the ranking, an effective learning rate α_i is computed for each unit:

$$\alpha_i = \epsilon \cdot h_\rho(r_i(\mathbf{d})) \tag{1}$$

This effective learning rate is used for the adaptation of all elements of the tupel $\{\mathbf{c}_i, \mathbf{W}_i, \mathbf{\Lambda}_i, \lambda_i^*\}$.[2] The function $h_\rho(r) = \exp(-r/\rho)$ ensures that not only the best–matching unit is updated, but every unit with a factor exponentially decreasing with its rank. The neighborhood range ρ and the global learning rate ϵ in training step t are determined by an annealing process as described before.

3 Adaptive learning rate control

Our algorithm for adaptive learning rate control is intended to replace the fixed annealing scheme for the neighborhood range ρ and the global learning rate ϵ. Instead, each unit i gets its own specific learning rate ϵ_i which changes during the course of training. This learning rate should be large if the unit is badly adapted to its partition of the training data, and small otherwise. For this reason, each local PCA unit is matched with the empirical distribution of data vectors $\{\mathbf{x}\}$ assigned to it. First, for a given $\mathbf{x}$ the principal components $\mathbf{y}$ are computed: $\mathbf{y} = \mathbf{W}^T(\mathbf{x} - \mathbf{c}_i)$. It is assumed that ideally — for a perfectly adapted local PCA unit — the principal components y_j $(j = 1, \ldots, m)$ are themselves drawn from one–dimensional Gaussian distributions with variances $\lambda_{i,j}$.

Given this assumption, a function $g(y_j) = \exp\left(-y_j^2/(2\lambda_{i,j})\right)$ is specified to determine the adaptation quality of the PCA unit separately for each principal

[1] Volume–normalized means here that all units are treated for the computation of the distance values as if they had the same elliptical volume.

[2] The centers are updated as specified in NG [3], the local PCAs with RRLSA [5].

ESANN 2010 proceedings, European Symposium on Artificial Neural Networks - Computational Intelligence and Machine Learning. Bruges (Belgium), 28-30 April 2010, d-side publi., ISBN 2-930307-10-2.

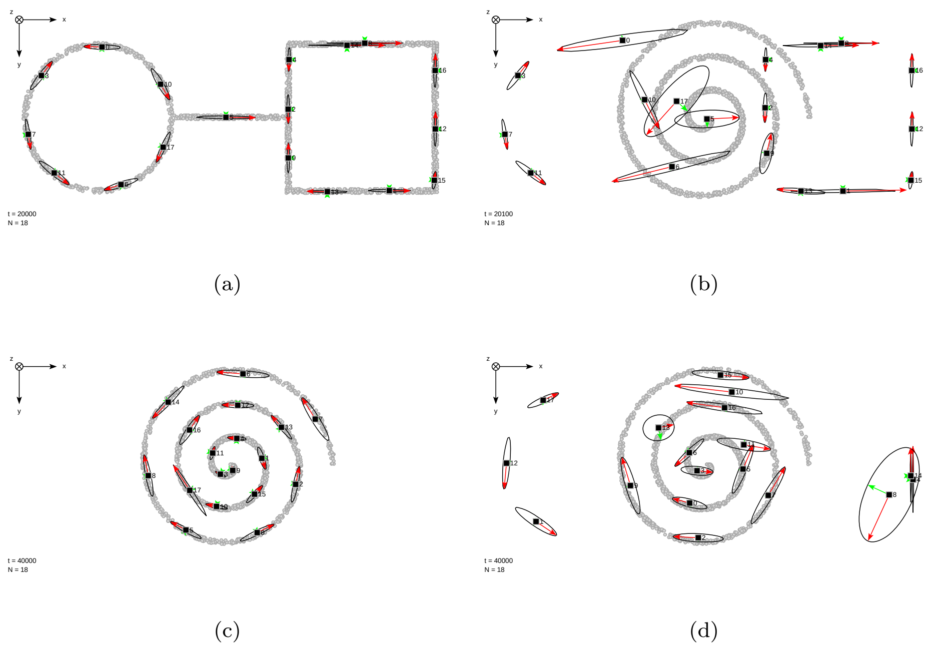

Fig. 1: Course of adaptation: (a) UDM–NGPCA after 20000 training steps on the RLS distribution (composed from the gray data points; the PCA units of the model are depicted as ellipsoids with axis half–lengths $\sqrt{\lambda_{i,j}}$); (b) at step 20100, now on the vortex distribution; (c) after 40000 training steps; (d) for comparison: classical NGPCA after 40000 training steps, the first 20000 of them on the RLS distribution, the last 20000 on the vortex.

direction j. It was shown in [7] that the expectations of $g(y_j)$ amount to $1/\sqrt{2}$ for all principal directions of a perfectly adapted unit, thus the deviations from this value indicate the adaptation quality. In our implementation, the expectations of $g(y_j)$ are computed with a low–pass filter and stored in a state vector $\mathbf{b}_i$ of size m for each unit i. Each element $b_{i,j}$ is updated by $b_{i,j} \longleftarrow (1-\beta_i)\, b_{i,j} + \beta_i\, g(y_j)$ in every training step. The low–pass parameter β_i depends on the rank of the unit to reflect the soft assignment of data vectors to units: $\beta_i = \mu\, h_\rho(r_i(\mathbf{d}))$. The parameter μ was set to 0.01 in all studies of this paper. Finally, an overall matching parameter D_i is computed for each unit based on the squared differences between the $b_{i,j}$ values and $1/\sqrt{2}$: $D_i = (2/m)\sum_{j=1}^{m}\left(b_{i,j} - (1/\sqrt{2})\right)^2$.

D_i varies between 0 (perfect match) and 1 (strong mismatch, $\mathbf{b}_i = \mathbf{0}$).[3] Finally, the unit–specific learning rate ϵ_i is determined by

$$\epsilon_i = (\epsilon_{\max} - \epsilon_{\min})\,\sqrt{D_i} + \epsilon_{\min}$$

[3]One has to note that all local data distributions with the appropriate projections will lead to a D_i value of zero, thus this measure is not unique and works only in conjunction with learning rules which align the principal directions to the data (as PCA naturally does).

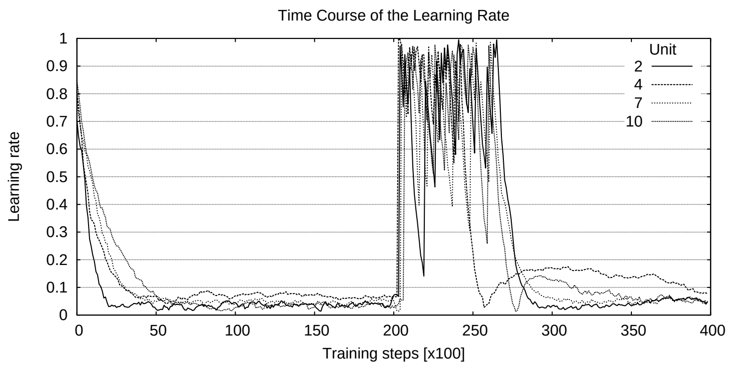

Fig. 2: Learning rate ϵ_i for four selected units (for details see text).

and the overall neighborhood range ρ by

$$\rho = (\rho_{\max} - \rho_{\min}) \sqrt{(1/N) \sum_{i=1}^{N} D_i} + \rho_{\min}.$$

ϵ_i replaces ϵ in (1), otherwise the basic NGPCA training algorithm stays unchanged. The important modification is that the fixed annealing scheme for ϵ and ρ is replaced in this way by an adaptive solution based on "unit–data matching" (this variation of NGPCA is called UDM in the following).

In addition, a wake–up heuristic has been incorporated by which the vector $\mathbf{b}_i$ is set to zero (which implies a strong increase of ϵ_i) whenever a unit i has not been the winner of the ranking process for a certain number of training steps. Throughout the studies in this paper, this number always amounts to 25 multiplied with N. This wake–up heuristic is necessary to speed up the training process whenever the distribution of the training data is abruptly changed.

4 Experimental studies and results

4.1 Study 1: Non–stationary low–dimensional data

In this experiment, a UDM model[4] with $N = 18$ units was first trained on a two–dimensional synthetic data distribution ("ring–line–square"/RLS, see Fig. 1a) for 20000 training steps ($m = n = 2$). After this period, the UDM model approximated the RLS distribution very well. At step 20001, the data distribution was completely changed to a distribution with vortex shape and smaller range in x-direction. In this way, the fit of the model to the data became suddenly very bad, and the local PCA units started to move and to change their shape (see Fig. 1b for step 20100). Training on the vortex continued until step 40000. Fig. 1c shows that finally the UDM model arrived again at a very good fit with the (changed) data distribution. For comparison, a classical NGPCA model (termed CLASSIC in the following) was trained with the same parameters (for an annealing period of $T_{\text{final}} = 40000$). The final result is depicted in Fig. 1d. Although the change of the data distribution took place in the middle of the

[4]Parameters: $\epsilon_{\max} = 1.0$, $\epsilon_{\min} = 0.01$, $\rho_{\max} = 1.5$, $\rho_{\min} = 0.02$.

Table 1: Grasping error E_{grasp}, average results (std. dev. in brackets)

	$N = 50$			$N = 100$		
	$m = 3$	$m = 5$	$m = 9$	$m = 3$	$m = 5$	$m = 9$
CLASSIC	1.7 (0.28)	2.1 (0.47)	6.9 (1.7)	1.3 (0.14)	1.5 (0.19)	4.0 (0.98)
UDM	2.0 (0.34)	2.0 (0.45)	6.6 (2.2)	1.3 (0.14)	1.7 (0.21)	4.0 (0.93)

Table 2: Grasping error E_{grasp}, best results (std. dev. in brackets)

CLASSIC	1.3 (0.06)	1.2 (0.11)	1.9 (0.28)	0.96 (0.06)	1.0 (0.05)	1.4 (0.18)
UDM	1.4 (0.09)	1.3 (0.08)	2.2 (0.33)	**0.93** (0.05)	1.0 (0.16)	1.7 (0.15)

annealing process, the CLASSIC model did not manage to rearrange its units for a good match with the vortex distribution. These courses of training are typical and can be easily reproduced. They illustrate that UDM is far superior to CLASSIC when abrupt changes of the training data distribution occur.

Fig. 2 shows the time course of the learning rates ϵ_i for four selected units of the UDM model for the training on the RLS/vortex distributions. After 5000 training steps on the RLS distribution, the learning rates settled down at a low level. Soon after the change to the vortex distribution at step 20000, the learning rates started to oscillate around large values for 7000 steps until all units had approached the vortex distribution. Finally, the learning rates converged again to small values. Thus, the adaptive learning rate control worked as intended.

4.2 Study 2: Stationary medium–dimensional data

In the second experiment, we tested if UDM models reach the same approximation quality as CLASSIC models. For this purpose, we used a stationary data set from a real–world robotic application. This data set contains around 3200 learning examples[5] for a kinematic control task: A robot arm with six rotational joints has to grasp a block which has been fixated before by a camera head. Each data vector has $n = 68$ dimensions, 20 of which encode the input (position and orientation of the block) and 48 of which the output (a pre–grasping and a grasping posture for the robot arm) (for a detailed specification see [6]).

We varied the number of principal components ($m = 3$, 5, or 9) and the number of units ($N = 50$ or 100). For each of these task conditions, we tested 81 different combinations of the maximum and minimum ϵ and ρ values[6] and trained 10 models for each of these combinations. After model adaptation, we determined the resulting grasping error E_{grasp} of the model when used as kinematic controller in the robotic task (see [2] on how to use an NGPCA model as feedforward controller). The grasping error is a weighted sum of different translational and rotational deviations from the ideal grasping posture; lower values are better; values around 1.0 indicate a very good performance.

Table 1 reports the average results over all parameter combinations, Table 2 the respective best result obtained among all the combinations. Both methods

[5] 85% training data, 15% test data for the generation of the presented results.

[6] $\epsilon_{\text{max}} = 1.0/0.7/0.4$, $\epsilon_{\text{min}} = 0.07/0.04/0.01$, $\rho_{\text{max}} = 2.0/1.0/0.5$, $\rho_{\text{min}} = 0.05/0.03/0.01$, $T_{\text{final}} = 30000$.

excel for 100 units with 3 principal components (3 is also the intrinsic local dimensionality of the arm data distribution). Overall, UDM and CLASSIC show a similar grasping performance with regard to the average values (sometimes UDM is fractionally better, sometimes CLASSIC). This picture changes slightly in favor of CLASSIC with regard to the optimum achievable results in Table 2. CLASSIC outperforms UDM in nearly all task conditions by a very small margin; however, UDM manages to achieve the overall lowest grasping error (0.93) and thus might be superior to CLASSIC if the right training parameter combination (incl. N and m) is selected.

5 Conclusions

We proposed a novel algorithm for adaptive learning rate control which replaces the fixed annealing scheme of NG. It was applied to Gaussian mixture models of the NGPCA type. Its core idea is to introduce a unit–specific learning rate which is adjusted automatically depending on the match between the local PCA of each unit (interpreted as Gaussian distribution) and the empirical distribution of the training data vectors assigned to the unit. In addition, the global neighborhood range is adjusted in a similar way considering the total match of all units.

In contrast to classical NGPCA with its fixed annealing scheme, the novel algorithm (called UDM) is applicable to non–stationary data distributions. This was shown for synthetic data distributions which abruptly changed their shape. Furthermore, UDM performed overall as well as classical NGPCA on stationary real–world robotic data. A related algorithm is the PLSOM approach originally developed for self–organizing maps [1] which we adapted to NGPCA in a previous study [7]. However, the results in [7] show that the UDM approach is overall superior to our PLSOM adaptation.

In conclusion, these results suggest that the proposed algorithm can be successfully applied to a broad variety of data approximation tasks which require real online learning on non–stationary data distributions.

References

[1] E. Berglund and J. Sitte. The parameter–less self–organizing map algorithm. *IEEE Transactions on Neural Networks*, 17(2):305–316, 2008.

[2] H. Hoffmann. *Unsupervised Learning of Visuomotor Associations*. MPI Series in Biological Cybernetics. Logos Verlag, Berlin, 2004.

[3] T. M. Martinetz, S. G. Berkovich, and K. J. Schulten. “Neural-Gas” network for vector quantization and its application to time–series prediction. *IEEE Transactions on Neural Networks*, 4(4):558–569, July 1993.

[4] R. Möller and H. Hoffmann. An extension of neural gas to local PCA. *Neurocomputing*, 62(1):305–326, 2004.

[5] S. Ouyang, Z. Bao, and G.-S. Liao. Robust recursive least squares learning algorithm for principal component analysis. *IEEE Trans. on Neur. Netw.*, 11(1):215–221, 2000.

[6] W. Schenck, H. Hoffmann, and R. Möller. Grasping to extrafoveal targets: A robotic model. *New Ideas in Psychology*, 2009 (online).

[7] R. Welsch. Adaptive Lernratensteuerung für Neural Gas Principal Component Analysis. B.Sc. thesis, Comp. Eng. Group, Faculty of Technology, Bielefeld University, 2009.