Neurocomputing 71 (2008) 3635–3639

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Letters

Spike-based cross-entropy method for reconstruction

András Lőrincz [a,*], Zsolt Palotai [a], Gábor Szirtes [b]

[a] Department of Information Systems, Eötvös Loránd University, Budapest H-1117, Hungary
[b] Department of Cognitive Psychology, Eötvös Loránd University, Budapest H-1117, Hungary

ARTICLE INFO

Article history:
Received 13 June 2007
Received in revised form
24 January 2008
Accepted 25 March 2008
Communicated by T. Heskes
Available online 22 April 2008

Keywords:
Cross-entropy method
Spike-based reconstruction
Reconstruction networks

ABSTRACT

Most neural optimization algorithms use either gradient tuning methods or complicated recurrent dynamics that may lead to suboptimal solutions or require huge number of iterations. Here we propose a framework based on the cross-entropy method (CEM). CEM is an efficient global optimization technique, but it requires batch access to many samples. We transcribed CEM to an *online* form and embedded it into a reconstruction network that finds optimal representations in a robust way as demonstrated by computer simulations. We argue that this framework allows for neural implementation and suggests a novel computational role for spikes in real neuronal systems.

1. Introduction

Due to the stochastic nature of both the sensory information and real neuronal system itself, it is common to view neural information processing as a probabilistic inference problem [2]. While this twofold uncertainty as a fundamental framework is usually accepted, the functioning itself is generally modeled as a dynamical system in that the system has to reach a given state while performing a task. Learning can then be seen as tuning the parameters for the best performance. On the other hand, functioning is usually defined through forming internal representations that allow the system to reconstruct the inputs. Particularly, 'sparse coding' (see e.g. [5,12]) using overcomplete bases (that is when the number of basis vectors or features is larger than the dimensionality of the input) has been suggested as an ideal candidate for such representations. Olshausen and Field [5] proposed a system that learns an overcomplete 'dictionary' from images and can reconstruct images using the learnt dictionary. Their method resulted in bases similar to the receptive fields of the simple cells in V1 suggesting biological significance. This system, however, is functionally weak: both the reconstruction and the learning of the basis dictionary follow an approximate gradient rule and an arbitrary distribution is imposed on the bases. Interestingly, sparse coding can be related [6] to a very efficient learning paradigm, the so-called support vector machine (SVM [16]) theory, but SVM lacks any real biological motivation so far. The reconstruction idea *and* sparse coding is also motivated by recent advances on compressed sensing [4] and L_1 Magic [1]. We address the issue how the sparse coding reconstruction scheme can be improved from a computational point of view subject to the constraints of biological relevance.

Here we propose a sparse coding scheme based on an efficient global optimization technique, the so-called cross-entropy method (CEM [14]). The resulting representations are sparse by construction. We tackle the task how to find the optimal sparse code in a network. In Section 2, we define the problem, outline CEM for combinatorial optimization and present its online version that suits neural implementation. In Section 3, we demonstrate some properties of the algorithm on different benchmark problems. In Section 4, we suggest a potential interpretation of the coexistence of spike-based and rate coding in biological systems. Links to generative and reconstruction networks are also briefly discussed.

2. Algorithm

The problem is the following: let $\mathbf{h} \in \mathbb{R}^n$ denote the input we want to reconstruct with a sparse representation ($\mathbf{x} \in \mathbb{R}^m$) using an overcomplete basis set $\mathbf{A} \in \mathbb{R}^{n \times m}$, where $m \gg n$. The corresponding optimization problem is

$$x^* := \arg\min_{\mathbf{x}} \varepsilon \cdot \|\mathbf{x}\|_{l_0} + \|\mathbf{h} - \mathbf{A}\mathbf{x}\|_{l_2}, \quad (1)$$

where $\|.\|_{l_n}$ denotes the l_n norm, ε is a trade-off parameter. The first term enforces low number of nonzero components, the second term minimizes the l_2 norm of the reconstruction error $\mathbf{e}(t) = \mathbf{h} - \mathbf{A}\mathbf{x}(t)$. We are solving the above problem using CEM as

* Corresponding author. Tel.: +36 209 0555/8473.
E-mail address: andras.lorincz@elte.hu (A. Lőrincz).

0925-2312/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2008.03.007

a

```
% inputs: initial dist. parameters (g(p(0)) ∈ BER^m),
% selection ratio (ρ),number of iters.(T),
% population size(N)and smoothing factor (α)
% CEM iteration main loop
for t from 1 to T,
        %draw N samples
        for i from 1 to N,
(1)         draw x^(i) from g(p(t − 1))
            % calculate cost function
(2)         f_i := f(x^(i))
        end loop
(3)     sort f_i-values in ascending order
        % update set threshold
(4)     γ(t) := f_ρ·N
        % update the set of elite samples
(5)     E(t) := {x^(i) | f(x^(i)) ≤ γ(t)}
        % get parameters of nearest distrib.
(6)     p'_j := (Σ_{x^(i)∈E(t)} χ(x_j^(i) = 1))/(ρ · N)
        % update dist. parameters
(7)     p_j(t) := (1 − α) · p_j(t − 1) + α · p'_j
(8)     t ← t + 1
end loop
```

```
% inputs: desired elite ratio(ρ̂), elite threshold (γ̂(0)),
% update rates and trade-off (α̂, ε ∈ [0, 1], β ≥ 0)
% initial distribution params (p(0) ∈ R^m),
% target (h ∈ R^n) and feature set (A ∈ R^{nm}, m ≥ n)
% online CEM main loop
for t from 1 to T, or until convergence (f(x(t))≤ θ)
(1)     generate indices ξ(t) ∈ {0, 1}^m from p(t − 1)
        % calculate internal representation
(2)     x(t) ← A(:, ξ(t))^# · h
        % calculate cost function
(3)     f(x(t)) ← ε · ||x(t)||_{l0} + ||h − Ax(t)||_{l2}
        % to check if sample belongs to the elite
(4)     if ( f(x(t)) < γ̂(t) )
            % update dist. params for nonzero features
(5)         p(t) ← (1 − α̂)p(t − 1) + α̂ · ξ(t)
            % update elite threshold
(6)         γ̂(t) ← γ̂(t − 1) − β
        else
            % update elite threshold
(7)         γ̂(t) ← γ̂(t − 1) + ρ̂ · β
        end if
(8)     t ← t + 1
end loop
```

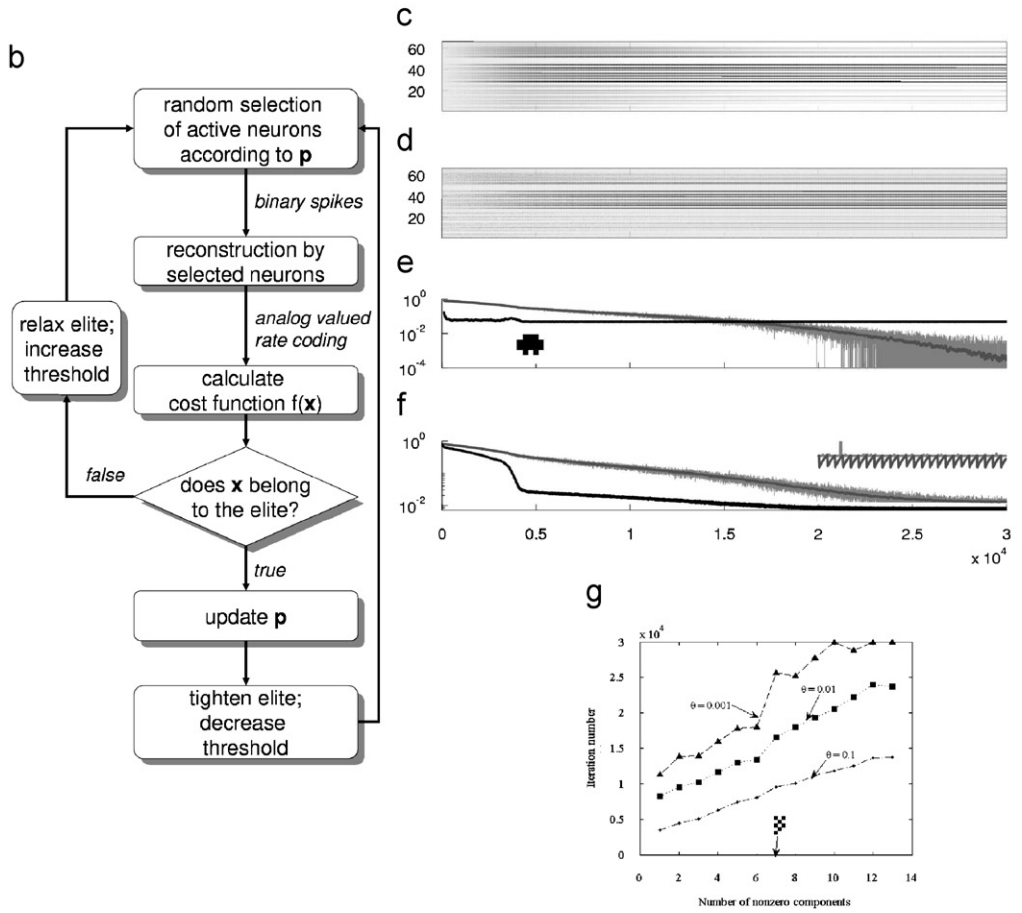


Fig. 1. (a) Left-hand side: pseudocode for CEM for binary reconstruction problem with Bernoulli distribution. Right hand side: online embedded CEM for reconstructing analog valued input. (b) Flow chart of the embedded online CEM algorithm for reconstruction. (c)–(f) Simulation results on the reconstruction task. For details, see the main text: (c) probability parameters $p^i(t) \in [0,1]$ at every 30th step; (d) components of the corresponding internal representation $x^i(t) \in 0,1$; (e) the evolution of the reconstruction error term ($\|\mathbf{e}(t)\|_{l2}$) in Eq. (1). Light gray line: average error, dark gray line: temporally smoothed average error, black line: empirical elite sample ratio, $\phi(t)$. The scale is logarithmic; (f) evolution of the overall cost function. Light gray line: average value, dark gray line: temporally smoothed average value, black line: the current elite threshold, $\hat{\gamma}(t)$. Inset: 500 steps from step no. 25001 to step no. 25500. The scale is logarithmic. (g) Iteration number as a function of complexity (# of nonzero pixels) at three different stopping criterion threshold θ values. Check board images of size 5×5 with 1–13 nonzero components were used. A sample image with seven nonzero components is depicted in the inset.

it exploits sparsity. CEM (for more details, see [3]) aims to find the (approximate) solution for global optimization tasks of the following form:

$$\mathbf{x}^* := \arg\min_{\mathbf{x}} f(\mathbf{x}),$$

where f is a general objective function. Instead of having a single candidate solution at each time step, CEM maintains a *distribution* $g(t)$ of $\mathbf{x}(t)$ for guessing solution candidates. The efficiency of this random guess is then improved by selecting the best samples—the so-called '*elite*'—and iteratively modifying $g(t)$ to be more peaked around the elite. It is known that, under mild regularity conditions, CEM converges with probability 1 and that, for a sufficiently large population, the global optimum is found with high probability [14,11]. It is also known that CEM is strikingly efficient in *combinatorial* optimization problems. Formally, CEM works as follows. Let g belong to a family of parameterized distributions, $\mathcal{G}$. Let $\mathbf{x}^{(1)},\ldots,\mathbf{x}^{(N)}$ be independent samples (N is fixed beforehand) from g. For each $\gamma \in \mathbb{R}$, the set of 'elite' samples,

$$\hat{E}_\gamma := \{\mathbf{x}^{(i)} \,|\, f(\mathbf{x}^{(i)}) \leqslant \gamma, 1 \leqslant i \leqslant N\},$$

provides an approximation to the level set $E_\gamma := \{\mathbf{x} \,|\, f(\mathbf{x}) \leqslant \gamma\}$.

Let U_γ and $\hat{U}_\gamma$ be the distribution over the level set E_γ and $\hat{E}_\gamma$, respectively. For small γ, U_γ is peaked around $\mathbf{x}^*$. CEM chooses $g(t)$ that is closest in the *cross-entropy* (or KL divergence) metric to the empirical distribution $\hat{U}_\gamma$. The algorithm iteratively modifies γ and g to reach the optimal performance value $\gamma^* = f(\mathbf{x}^*)$. CEM uses some technical tricks. It prohibits depletion of $\hat{E}_\gamma$ by maintaining the best $\rho \cdot N$ samples ($1-\rho$ quantile), where $\rho \in [0,1]$ is a fixed ratio. Also, it deals with parameterized distribution families where parameters can easily be approximated from simple statistics of the elite samples. Note also that updating the distribution parameters can be too coarse, so a smoothing factor α is applied (see step no. (7) of the CEM pseudocode in Fig. 1(a)).

Informally, CEM is a maximum likelihood method, without immature decisions. Its success stems from its innovative idea about the maintenance of an *elite* and the step-by-step sharpening of the elite's distribution. Our contribution is the recognition that this *gradual sharpening* may be achieved without samples to be stored.

We use the Bernoulli distribution for our reconstruction problem: let the domain of optimization be $D = \{0,1\}^m$, and each component be drawn from *independent* Bernoulli distributions, i.e. $\mathcal{G} := BER^m$. Each distribution $g \in \mathcal{G}$ is parameterized with $\mathbf{p} = (p_1,\ldots,p_m)$. For each component x_j of $\mathbf{x} \in D$, x_j equals 1 or x_j equals 0 with probability p_j and $(1-p_j)$, respectively. In turn, the optimal distribution g^* has the following parameters:

$$p_j^* := \frac{\sum_{x^{(i)} \in E} \chi(x_j^{(i)} = 1)}{\sum_{x^{(i)} \in E} 1} = \frac{\sum_{x^{(i)} \in E} \chi(x_j^{(i)} = 1)}{\rho \cdot N}, \tag{2}$$

where $j = 1 \ldots m$ and $\chi(\cdot)$ is an indicator function. (For the derivations of Eq. (2), see [3].) Thus, the parameters of g^* are simply the componentwise empirical probabilities of 1's in the elite set. Details of the algorithm are provided in the l.h.s. of Fig. 1(a).

This algorithm can easily be applied to Eq. (1), provided that overcomplete memory is approximately independent as in [5]. The assumed independence greatly reduces the difficulty of the optimization: instead of approximating high-dimensional distributions it is sufficient to estimate one-dimensional distributions of the features. For this reason, reconstruction can be seen as a two-step task: first, we have to find the right features and second, for each choice we have to find the corresponding weights. CEM randomly draws an *index* vector, $\xi \in \{0,1\}^m$ from the distribution $g(t)$ and we need to calculate the norm of the reconstruction error $\mathbf{e}(t)$ in Eq. (1) to yield the cost of the index vector. To this end, we *embed* CEM into a *reconstruction network*, which is a particular type of generative networks (see e.g. [7,8,10] and references therein).

In short, a simple reconstruction network has three layers of neurons with firing rate-based encoding (that is neurons here are analog valued computational units). These layers maintain the target (input, $\mathbf{h}$), the internal representation ($\mathbf{x}(t)$) and the reconstruction error ($\mathbf{e}(t)$), respectively. The code words of the dictionary (feature vectors or base vectors, rows of $\mathbf{A}^{\#}$) may correspond to the synaptic weights between the internal representation and the target. Particularly, embedding CEM into this framework means that probabilities of the components converge either to 1 or to 0. The amplitude of the nonzero components is always determined by a separate procedure, the pseudoinverse method, which is feasible because of the l_0 norm that we use.

However, from a neuronal network perspective, there is a serious drawback of CEM: it requires batch access to many samples. To overcome this obstacle, we make the algorithm *online*. The idea is the following: instead of calculating the threshold level using the order statistics of the best ρ ratio of the samples, a moving average approximate level ($\hat{\gamma}(t)$) is updated according to the currently calculated cost function. Let us denote the current value of the cost function by $f(\mathbf{x}(t)) = \varepsilon \cdot \|\mathbf{x}(t)\|_{l_0} + \|\mathbf{h} - \mathbf{A}\mathbf{x}(t)\|_{l_2}$. Then the update is as follows. If $f(\mathbf{x}(t)) \leqslant \hat{\gamma}(t-1)$, i.e. it is small enough to consider $\mathbf{x}(t)$ as elite, then $\hat{\gamma}(t) \leftarrow \hat{\gamma}(t-1) - \beta$ and the level for the elite gets decreased. If, however, $f(\mathbf{x}(t)) > \hat{\gamma}(t-1)$, then the level becomes less restrictive: $\hat{\gamma}(t) \leftarrow \hat{\gamma}(t-1) + \hat{\rho} \cdot \beta$, where $\beta \geqslant 0$ is a step size parameter that scales with the range of $f(\mathbf{x}(t))$. For simplicity, in the simulations we took β as an arbitrary constant, but it can also be updated online while *the algorithm remains convergent* [15]. This modification eliminates the arbitrariness in scaling and speeds up the algorithm. If $\phi(t)$ denotes the ratio of the elite in the last n samples, then the role of the sliding parameter $\hat{\rho}$ is that it ensures that $E(\phi(t)) \approx \hat{\rho}$ ratio of the samples has a score less than or equal to the threshold preserving the concept of elite selection and elite driven sharpening of the sample generation procedure. The update of the distribution parameters $p(t)$ corresponds to the smoothing stage in the batch version (see step no. (5) of r.h.s. of Fig. 1(a)). The relation between the update parameter, $\hat{\alpha}$ and the smoothing factor in batch CEM is $\hat{\alpha} \approx \alpha/\rho N$ (that is it describes the contribution of a single sample to the distribution update, [15]). The online CEM algorithm is summarized in the r.h.s. of Fig. 1(a) and in the flow chart of Fig. 1(b).

3. Simulations

As the single modification in our algorithm is how the elite is maintained, all convergence results of CEM still hold (for proofs, see [15]). The efficiency of the online CEM is demonstrated on a benchmark clustering problem, while the overall functioning is presented on a toy problem.

3.1. Banana clustering

In this benchmark random points are scattered around a segment of a circle. In data generation we followed [9] where the batch CEM was compared to other standard methods, such as K-means clustering or linear vector quantization. It was shown that batch CEM can outperform the others regarding the accuracy. Here we only compare the online CEM to the batch version. The task was to cluster 200 points into 5, 10 or 20 Gaussian clusters (for details, see [9]). The algorithms learned the mean and the

Table 1
Simulation results on the banana clustering problem

Required number of clusters	5	10	20
Mean required sample number			
Batch	30 700 ±1140	49 700 ±2750	76 900 ±2450
Online	196 000 ±28 000	349 000 ±26 000	704 000 ±95 000
Mean relative error			
Batch	0.0031 ±0.0045	0.0064 ±0.0029	0.015 ±0.0053
Online	0.0019 ±0.0047	0.0050 ±0.0036	0.012 ±0.0048

The online and batch versions are compared using two statistics: the first is the number of required samples (±std.) averaged over 10 runs and 10 data sets for each cluster set. The second one is the average relative error (±std.) compared to the best solution (lowest cost).

deviation of the Gaussians. For statistical comparison, we randomly regenerated the datapoints 10 times for each cluster number and for each data set there were 10 runs. Overall, there were 300 runs (3 cluster number $\times 10$ data sets $\times 10$ runs) for both the batch and the online CEM. The stopping criterion was always either that (i) the difference between the cost belonging to the 10 last elite samples should be less than 0.01 or (ii) the deviation of the cluster centers would be less than 0.01. The parameters were the following: for the batch CEM, sample size $N = 800$, elite ratio $\rho = 0.025$ and the smoothing factor $\alpha = 0.7$. For the online CEM, $\hat{\alpha} = 0.01$, $\hat{\rho} = 0.025$ and $\beta = 1$. Table 1 summarizes the results. While the online CEM requires more samples (longer iterations), its accuracy is comparable with the batch results. Note also that we did not optimize the parameters to gain speed.

3.2. Toy problem

Figs. 1(c)–(f) show the simulation results on a reconstruction task defined by Eq. (1). The 'car' input shown on the inset of Fig. 1(e) is an image vector of size 5×5 to be reconstructed using an overcomplete feature set of 65 vectors with one or two nonzero elements (pixels) only. There are 20 vertical and 20 horizontal base vectors with two nonzero pixels set either vertically or horizontally, respectively. Further, there are 25 base vectors with exactly one nonzero pixel. The problem is ill-posed, so regularization (for example, sparsity requirement) is needed. The sample image can be reconstructed with different combinations of at most eight active features. The results are averaged over 100 trials, each trial was 30 000 step long. The following parameters were used: $\hat{\alpha} = 0.01, \beta = 0.01, \varepsilon = 0.005, \hat{\rho} = 0.05$. Figs. 1(c) and (d) show the probability parameters $p^i(t)$ and the components of the corresponding internal representation $x^i(t)$, respectively. Ambiguity in sparse representation results in more than eight active components in both vectors and due to the averaging the final component values are not 0's (white) or 1's (black), but are spread over $[0, 1]$. Fig. 1(e) plots the reconstruction error and the corresponding empirical elite ratio, $\phi(t)$ as a function of time. It can be seen that $\phi(t)$ is more or less constant thus maintaining the correspondence to the batch CEM. The evolution of the overall cost function ($f(\mathbf{x}(t))$) and the elite threshold $\hat{\gamma}(t)$ is depicted in Fig. 1(f). Comparing Figs. 1(e) and (f) reveals that 0 reconstruction error can early be reached, but finding the sparsest solution takes longer. The inset in Fig. 1(f) is a magnification of 500 steps. It shows that, when all solutions are already optimal, the oscillation of $\hat{\gamma}(t)$ is caused by the asymmetric update. At about every $(1/\hat{\rho})$th step there is a drop of size β, otherwise $\hat{\gamma}(t)$ gets increased by $\hat{\rho} \cdot \beta$. To demonstrate the scaling properties of the algorithm, we reconstructed a series of check board images of increasing complexity (that is, with increasing number of nonzero pixels). In Fig. 1(g) the required iteration number as a function of complexity is depicted at three different θ values (iteration stops when $f(\mathbf{x}(t)) \leqslant \theta$). The required time does not increase as fast as the number of possible combinations increases; scaling is approximately linear in the range studied.

4. Discussion

In this paper we proposed a network architecture for sparse input reconstruction using an overcomplete basis set. Sparse overcompleteness has some interesting consequences. Components or features are either present or absent and thus their *presence* can be *detected* independently.

Independence and sparseness may recast the reconstruction task as a combinatorial optimization problem for which we suggested a modification of the very efficient and simple CEM. CEM is robust, fast and essentially parallel, but requires batch access to samples. Our modification resulted in an online version that may enable one to find mapping onto real neuronal networks. Another extension is that for analog valued inputs and/or representations, we embedded the online CEM into a reconstruction network architecture that can directly calculate the reconstruction error term of the cost function through feedforward and feedback connections. (See, e.g. [10] about the potential role of such networks in real neuronal systems.) The advantage of such generative networks is that if both the upstream and downstream connections are well tuned then the system can provide an almost optimal initial distribution parameter guess that speeds up CE convergence considerably. In this case, the loop operates almost like a pure feedforward system and may serve many seemingly feedforward computational tasks.

Another important aspect of the online CEM is that it is essentially built on the *concurrent* use of digital (spike-based feature selection) and analog valued coding (like the input and its representation). This particular feature may contribute to the long standing debate (see e.g. [13] and references therein) about the different neuronal coding schemes by suggesting a new computational role for spikes *and* maintaining rate code as well. The novel interpretation is that spikes are used to improve the probability of the compositional representation, whereas rate code is about the magnitude of the selected components. Thus, beyond the advantages of the algorithm featuring global optimization and parallel sampling in distributed networks, it might have biological significance and resolve some of the puzzles of neuronal networks.

Acknowledgments

G.S. is supported by the Zoltán Magyary fellowship of the Hungarian Ministry of Education. We are grateful to one of our referees for his valuable comments. This research has been supported by the EC NEST grant 'PERCEPT' (FP6-043261). Opinions

and errors in this manuscript are the author's responsibility, they do not necessarily reflect those of the EC or other project members.

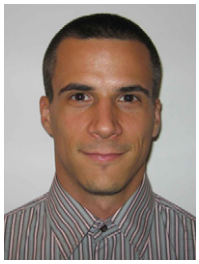
References

[1] E. Candes, J. Romberg, T. Tao, Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information, IEEE Trans. Inf. Theory 52 (2006) 489–509.
[2] P. Dayan, L.F. Abbott, Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems, The MIT Press, Cambridge, MA, 2001.
[3] P. de Boer, D. Kroese, S. Mannor, R. Rubinstein, A tutorial on the cross-entropy method, Ann. Oper. Res. 134 (2005) 19–67.
[4] D.L. Donoho, Compressed sensing, IEEE Trans. Inf. Theory 52 (2006) 1289–1306.
[5] D.J. Field, What is the goal of sensory coding?, Neural Comput. 6 (1994) 559–601.
[6] F. Girosi, An equivalence between sparse approximation and support vector machines, Neural Comput. 10 (1998) 1455–1480.
[7] G. Hinton, Z. Ghahramani, Generative models for discovering sparse distributed representations, Philos. Trans. R. Soc. B 352 (1997) 1177–1190.
[8] B.K.P. Horn, Understanding image intensities, Artif. Intell. 8 (1977) 201–231.
[9] D. Kroese, R. Rubinstein, T. Taimre, Application of the cross-entropy method to clustering and vector quantization, J. Global Optim. 37 (2007) 137–157.
[10] A. Lőrincz, B. Szatmáry, G. Szirtes, The mystery of structure and function of sensory processing areas of the neocortex: a resolution, J. Comput. Neurosci. 13 (2002) 187–205.
[11] L. Margolin, On the convergence of the cross-entropy method, Ann. Oper. Res. 134 (2005) 201–214.
[12] B.A. Olshausen, Sparse codes and spikes, Probabilistic Models of the Brain: Perception and Neural Function, MIT Press, Cambridge, MA, 2002.
[13] P. Reinagel, R.C. Reid, Temporal coding of visual information in the thalamus, J. Neurosci. 20 (2000) 5392–5400.
[14] R. Rubinstein, D.P. Kroese, The Cross-Entropy Method, Springer, New York, 2004.
[15] I. Szita, A. Lőrincz, Online variants of the cross-entropy method, Technical Report ⟨http:/nipg.info/⟩, 2007.
[16] V. Vapnik, The Nature of Statistical Learning Theory, Springer, New York, 1995.

András Lőrincz, professor, senior researcher has been teaching at the Faculty of Informatics of the Eötvös Loránd University, Budapest since 1998. His research focuses on distributed intelligent systems and their applications in neurobiological and cognitive modeling, as well as medicine. He has founded the Neural Information Processing Group of Eötvös University and he directs a multidisciplinary team of mathematicians, programmers, computer scientists and physicists. He has acted as the PI of several successful international projects in collaboration with Panasonic, Honda Future Technology Research and the Information Directorate of the US Air Force in the fields of hardware–software co-synthesis, image processing and human–computer collaboration. He is a member of two EU FP6 research projects; the NEW TIES FET project and the PERCEPT NEST project.

He graduated in physics and received his C.Sc. degree in 1986 in molecular physics. He worked on experimental and theoretical solid-state physics, chemical physics, quantum control, and artificial intelligence at the Hungarian Academy of Sciences, University of Chicago, Brown University, Princeton University, and the Illinois Institute of Technology. He authored about 200 peer reviewed scientific publications on different disciplines with about 700 independent citations. He also has several patents.

He has received the Széchenyi Professor Award, Master Professor Award and the Széchenyi István Award in 2000, 2001, and 2004, respectively. Four of his students won the prestigious Pro Scientia Gold Medal in the field of information science over the last 6 years. In 2004, he was awarded the Kalmár Prize of the John von Neumann Computer Society of Hungary. He has become an elected Fellow of the European Coordinating Committee for Artificial Intelligence in 2006.



Zsolt Palotai obtained his M.Sc. degree in technical informatics at the Budapest University of Technology in 2003. He was won second and third prizes in the biannual Hungarian student research competition in Informatics on three occasions in 2001 and 2003. He has finished his Ph.D. studies in Computer Sciences in the group of András Lőrincz at the Eötvös Loránd University, Budapest where he worked on distributed computation, sparse representation learning, machine learning, and artificial neural networks, among other things.

Gábor Szirtes received an M.Sc. in chemistry (1999) and Ph.D. in computer sciences (2005) at Eötvös Loránd University, Budapest, Hungary. He has worked on theoretical neuroscience with Kenneth R. Miller at UCSF, San Francisco and Columbia University, New York. He is currently working with Andras Lorincz (modeling the entorhinal-hippocampal circuitry) and with Valeria Csepe (development of new methods in EEG-analysis). He was awarded the Magyary Zoltán postdoctoral fellowship in 2006.