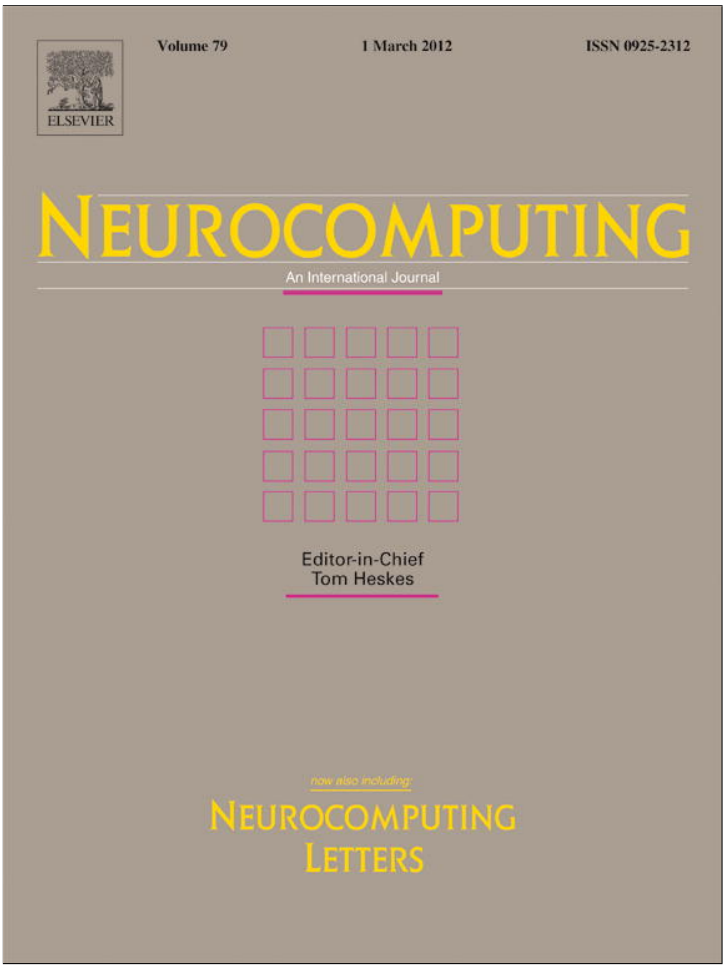

Neurocomputing 79 (2012) 115–124

Contents lists available at SciVerse ScienceDirect

Neurocomputing

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

Sparse and silent coding in neural circuits

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

[a] Eötvös Loránd University, Pázmány Péter sétány 1/C, Budapest H-1117, Hungary
[b] Sparsense Inc., USA
[c] ELTE-Soft kft., Pázmány Péter sétány 1/C, Budapest H-1117, Hungary

ARTICLE INFO

Article history:
Received 29 June 2010
Received in revised form
29 May 2011
Accepted 20 October 2011
Communicated by R. Kozma
Available online 15 November 2011

Keywords:
ℓ_1-Norm
Cross-entropy method
Sparse coding

ABSTRACT

Sparse coding algorithms find a linear basis in which signals can be represented by a small number of non-zero coefficients. Such coding may play an important role in neural information processing and metabolically efficient natural solutions serve as an inspiration for algorithms employed in various areas of computer science. In particular, finding non-zero coefficients in overcomplete sparse coding is a computationally hard problem, for which different approximate solutions have been proposed. Methods that minimize the magnitude of the coefficients ('ℓ_1-norm') instead of minimizing the size of the active subset of features ('ℓ_0-norm') may find the optimal solutions, but they do not scale well with the problem size and use centralized algorithms. Iterative, greedy methods, on the other hand are fast, but require *a priori* knowledge of the number of non-zero features, often find suboptimal solutions and they converge to the final sparse form through a series of non-sparse representations. In this article we propose a neurally plausible algorithm which efficiently integrates an ℓ_0-norm based probabilistic sparse coding model with ideas inspired by novel iterative solutions.

Furthermore, the resulting algorithm does not require an exactly defined sparseness level thus it is suitable for representing natural stimuli with a varying number of features. We demonstrate that our combined method can find optimal solutions in cases where other, ℓ_1-norm based algorithms already fail.

1. Introduction

In the last decades a large body of research has focused on the properties of sparse linear representations of signals. For a signal $\mathbf{x} \in \mathbb{R}^n$ we want to get the sparsest possible representation using a given feature dictionary $\mathbf{D} \in \mathbb{R}^{n \times m}$:

$$\min \|\mathbf{a}\|_0 \quad \text{subject to } \mathbf{Da} = \mathbf{x} \tag{1}$$

where $\mathbf{a} \in \mathbb{R}^m$ denotes the coefficients.

It should be noted that even sparser solutions can be gained if $\mathbf{D}$ is allowed to adapt to the input statistics: dictionary learning [1] currently attracts a lot attention in computer science, but in this article we focus on the subproblem of coefficient selection.

Linearity is assumed for tractability reasons, although there are non-linear coding models pursuing e.g. factorial codes [2] which – depending on the statistics of the signals to be represented – may actually yield sparse codes as well. In fact, it has been argued [3] that to fully understand input statistics (in terms of full probability distribution as opposed to marginal distributions) non-linear transformations are essential.

It is of particular interest when the number of basis vectors (features) is greater than the dimensionality of the input signal ($m \geq n$). In signal processing, statistics and machine learning overcompleteness offers enhanced resistance to small perturbations in the signal (invariance), greater flexibility in capturing data (yielding more compact forms) and ever larger sparsity.

Interestingly, neuroscience also advocates overcomplete sparse coding (OSC) as a universal representation mechanism in sensory areas [4,5], but for reasons a bit different from those mentioned above. As it was argued in e.g. [4,6] sensory systems extract second and higher order statistical dependencies of the stimuli in order to explicitly explain those stimuli by a set of independent events (in line with the principle of redundancy reduction [7]). Since most natural stimuli are inherently sparse (e.g. images can be described in terms of a small number of structural primitives [8]), searching sparse representations may actually gives rise to independent codes.

To cope with uncertainty in the observations, neural systems are supposed to employ generative models [9] in which those independent hypothesis (events) can describe the stimuli. In a linear framework transformations in early sensory areas may be characterized as

$$\mathbf{x} = \mathbf{Da} + \boldsymbol{\epsilon} \tag{2}$$

* Corresponding author.
E-mail address: andras.lorincz@elte.hu (A. Lőrincz).

0925-2312/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2011.10.017

where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0},\boldsymbol{\Sigma})$ is additive Gaussian noise with $\mathbf{0}$ mean and covariance matrix $\boldsymbol{\Sigma}$. If uncertainty is properly defined (like assuming Gaussian noise) then such representational form can be recast in a probabilistic framework [4] in which the sparsity constraint is introduced through priors of the coefficients. The maximum likelihood estimation gives rise to the following objective for OSC:

$$\mathbf{a}_\lambda \triangleq \arg\min_{\mathbf{a}} \tfrac{1}{2}\|\mathbf{x}-\mathbf{D}\mathbf{a}\|_2^2 + \lambda\|\mathbf{a}\|_0 \tag{3}$$

where sparsity is measured by the ℓ_0 quasi-norm, which counts the number of non-zero elements ($\|\mathbf{a}\|_0 \triangleq \#\{i, \text{ s.t. } a_i \neq 0\}$). Throughout this paper we will use this objective.

While various constraints can be imposed (e.g. non-negativity [10]), sparse coding models seem to get the strongest support from theory (e.g. [11]) as well as from experiments (e.g. [12,6], but see e.g. [13]). Besides efficient coding, the importance of sparsity in neural systems has been motivated by various, partially overlapping considerations (for a review, see e.g. [5]).

One important argument is about metabolic efficiency: it is required to control the metabolic cost of neural activity which to a large extent is a consequence of *spiking* [14]. Regarding representational power, sparse coding achieves a balance between distributed dense coding and local coding in terms of memory capacity and noise tolerance.

Another striking property of neural representations is that stimuli are often over-represented *many times* using extremely overcomplete dictionary ($m \gg n$). Such high level of overcompleteness would also be advantageous for artificial systems so computer science should benefit from understanding the mechanisms that make natural OSC so efficient (by translating metabolic cost as computational cost, for example). In turn, we search improvements over currently available solutions which can be implemented in a neurally plausible way. The underlying difficulty is that finding optimally sparse representation of a signal using an overcomplete dictionary is a so-called 'NP-hard' combinatorial problem [15]. Informally there is no guaranty that any approach would provide a solution to *all* problems in this form in *polynomial time*. An algorithm is said to be solvable in polynomial time if the number of steps required to complete the algorithm for a given input is $\mathcal{O}(n^p)$ for some non-negative integer p, where n is the dimension of the input. Polynomial algorithms are considered 'fast' in theory, but in practice there are significant differences among the solutions depending on the complexity of the problem. For sparse coding, the complexity is related to the level of overcompleteness and the required sparsity: $\kappa = k/m$, where k is the number of active components. As most algorithms either fail to find the optimal solutions in a reasonable time in problems comparable in complexity to those routinely solved by the neural system (e.g. 25- to 100-fold overcomplete sparse representations of stimuli in the sensory areas) we are searching novel solutions, which (1) excel current approaches in efficiency and (2) follow the principles of neural interactions (employing parallel, local, Hebbian interactions only).

In order to motivate our solution, first we briefly review the two popular classes of sparse coding methods: ℓ_1 norm based solutions and iterative, greedy approaches and discuss the limitations of neural implementations. In particular, we detail the so called subspace pursuit method (SP) [16], which becomes superior to other greedy methods by the incremental refinement of the basis selection. In Section 2 we present a stochastic global optimization scheme for deterministic combinatorial problems called cross-entropy method (CEM) [17] which can also be applied in sparse coding problems by explicitly minimizing the number of non-zero coefficients [18]. In this section we also show our OSC method which is essentially an integration of CEM and SP. In Section 3 we demonstrate the superiority of our method over established methods on artificial, but large-scale problems when exact solution is known. In Section 4 we discuss the relevance of our proposal to distributed, neurally plausible computations. We also discuss correspondences between the proposed algorithm and the underlying computations in sensory processing. Conclusions are drawn in Section 5. In the Appendix we detail the pseudo-code of SP and CEM for reproducibility.

1.1. Solution via linear programming

What makes the problem of Eq. (3) difficult is that the objective is not continuous because of the last term (sparsity constraint on the coefficients) and because of a potentially large number of local minima.

One approximate solution is to replace the ℓ_0 sparsity inducing norm by ℓ_1 (it measures the sum of the absolute values of the components) as it yields gradient learning rules for the coefficients (see e.g. [19]), too.

$$\mathbf{a}_\lambda \triangleq \arg\min_{\mathbf{a}} \tfrac{1}{2}\|\mathbf{x}-\mathbf{D}\mathbf{a}\|_2^2 + \lambda\|\mathbf{a}\|_1 \tag{4}$$

where λ is again a trade-off parameter controlling the balance between the reconstruction quality and sparsity. For more details, see e.g. [20].

For these methods, features (columns of $\mathbf{D}$) are required to have unit norm in order to bound the optimization problem, otherwise arbitrarily small activity could be achieved by increasing the norm of the corresponding features. Importantly, as recent results in signal processing (e.g. [21–23]) show, the sparse coefficients in the original problem of Eq. (3) can be exactly recovered by replacing ℓ_0 norm with ℓ_1 norm if 1, generating sources can indeed be assumed to be sparse in a proper basis and 2, some not too restrictive conditions about the basis are met (for a comprehensive list of conditions see [24]). In turn, *linear programming* methods can be used to solve the optimization problem, for which there exist efficient interior point-type linear programming [25] techniques. These methods have proven robust in the sense that even noisy or heavily undersampled signals can be recovered this way. This observation relaxes the combinatorially hard problem, yet the computational complexity of linear programming methods is still prohibitive for large scale problems. To address the scaling issue, other, non-global, but fast methods have also been developed.

1.2. Iterative methods

The need for faster decoders – aiming at linear time operation – has brought about improved linear programming methods (e.g. linear programming with preconditioned conjugated gradients [26]) as well as other classes not based on convex programming. Most importantly, a family of greedy algorithms (e.g. [27,28]) – following the ideas of the matching pursuit (MP) approach [29] – became dominant due to their low complexity and simple geometric interpretation.

The core idea is that reconstruction is built up iteratively: the best element of the dictionary is selected to minimize the *residual* at iteration t: $\mathbf{r}^t = \mathbf{x}-\mathbf{D}[\mathcal{I}]\mathbf{a}^t$, that is the difference between the signal and its actual estimation made of the combination of the previously chosen elements of the dictionary. $\mathcal{I}$ denotes the corresponding index series of the already selected basis functions, $\mathbf{D}[\mathcal{I}]$ denotes the "truncated" matrix of the selected basis forming a restricted top-down reconstruction matrix, whereas $\mathbf{a}^t$ represents the corresponding (optimized) coefficients (formal definitions are given in the Appendix). Although MP is fast, its approximation performance can be quite poor. Improved solutions, like orthogonal matching pursuit (OMP, also known as forward stepwise regression) methods provide better performance, but usually require longer running times.

In general the iteration of the different greedy methods consists of the following two steps:

1. Select: Find basis vector $\mathbf{d}_b$ that provides the largest projection of the residual and whose index is not yet contained in $\mathcal{I}$:

$$b = \arg\max_{j \notin \mathcal{I}} \|\mathbf{d}_j^T \mathbf{r}^t\|. \quad (5)$$

   Expand the index set with this new index: $\mathcal{I} \leftarrow \mathcal{I} \cup b$.
2. Improve basis set and update: Expand the sparse basis: $\mathbf{D}[\mathcal{I}] \leftarrow \mathbf{D}[\mathcal{I}] \cup \mathbf{d}_b$. Compute the new approximation, i.e., the coefficients $\mathbf{a}^{t+1}$ and the residual $\mathbf{r}^{t+1}$ by minimizing the approximation error, $\|\mathbf{r}^{t+1}\|_2$ according to the applied method.

Iteration may stop when the allowed number of dictionary elements is reached or when the approximation error is reduced below a predefined threshold. Note that MP and OMP methods differ in their update procedure [30]:

MP: $\mathbf{r} \leftarrow \mathbf{r} - \mathbf{d}_b \mathbf{d}_b^T \mathbf{r}$,
OMP: $\mathbf{r} \leftarrow \mathbf{r} - \mathbf{D}[\mathcal{I}]\mathbf{D}[\mathcal{I}]^{\dagger}\mathbf{r}$ $(= \mathbf{x} - \mathbf{D}[\mathcal{I}]\mathbf{D}[\mathcal{I}]^{\dagger}\mathbf{x})$,

where † stands for taking the pseudoinverse ($|\mathcal{I}| \leq n$). In MP not only the choice of the feature subset is suboptimal, but also the approximation of the coefficients. OMP improves upon MP by recomputing all coefficients at every step in order to ensure orthogonality between the residual and the columns of $\mathbf{D}[\mathcal{I}]$. While OMP methods achieve better reconstruction, they impose stronger constraints on the matrices than the original theorem for ℓ_1-norm based optimization does (see e.g. [16]). Further acceleration can be achieved if more than 1 basis functions are chosen at every iteration as e.g. in the 'stagewise' OMP [31]. Nevertheless, all of these iterative procedures are somewhat limited by the following issue. Once a verdict is made, the chosen features remain in the subset until the algorithms terminates. One mistake thus may tend to influence the reconstruction quality for many subsequent iteration steps.

1.2.1. Subspace pursuit (SP) methods

A remedy for the above mentioned problem has been independently proposed in [16,32]. These methods assume that at most k components are sufficient to represent the input. First, a candidate representation is generated using all basis, then a subset of basis is selected that corresponds to the k largest components of the representation. This initial selection is then iteratively refined: the residual (that is the difference between the input and the current approximate representation) is calculated and mapped onto the representation space using all basis again. Then – similar to the initial step – another k basis are selected based on the amplitude of the corresponding components of the mapped residual. The original input is then projected again to the representation space using a $2k$ element basis set formed by fusing the two basis subsets. Finally, k basis are selected again that corresponds to the k largest components of the projection (basis shrinkage). The iteration stops when the norm of the residual does not decrease anymore. SP has superior speed, scaling and reconstruction accuracy over other iterative methods by directly refining the subset of reconstructing (active) components at *each* iteration. Since our solution is partially motivated by SP, for completeness the corresponding pseudocode of the SP variant of [16] is provided in the Appendix (Table 6).

1.3. Limitations of neural implementation

Although the above mentioned optimization methods are getting more efficient regarding their computational time complexity, they suffer from different limitations that prevent their efficient and neurally plausible implementation. As it was stated in [33] most ℓ_1 norm based sparse coding methods share the following properties: (1) they include neurally implausible (centralized, non-local) operations, (2) they fail to produce exactly zero-valued coefficients in finite time (sub-optimal solutions), (3) for time-varying stimuli they produce non-smooth variations in the coefficients, and (4) they use only a heuristic approximation to minimizing a desired objective function (when conditions on ℓ_0–ℓ_1 norm equivalence do not hold).

On the other hand, iterative methods, like MP, OMP, and SP rely on matrix–vector multiplications, which are – in theory – suitable for local rule based implementations. However, in these methods the transposed form of the reconstruction matrix $\mathbf{D}$ is used in every iteration step. While reconstruction of the original signal uses only $k \times n$ channels (top-down connections or active synapses in neural networks) and thus only $k \times n$ multiplications are required, selecting or updating the coefficients requires all $n \times m$ bottom-up channels to transmit signals between the layers (for relevant overcomplete SC problems $m \gg n \geq k$). Considering the significant metabolic cost of synaptic activation (dendritic propagation, excitatory postsynaptic potentials and transmitter release) [34], the huge m/k ratio may suggest that economical computations should prefer $k \times n$ top-down reconstructions over $n \times m$ bottom-up transformations. In turn, a neurally plausible sparse coding system should also minimize the required synaptic transfer when seeking a solution. This requirement essentially implies that during the process of sparsification the transient activities should also be sparse thus maintaining lifetime sparseness (i.e., sparse activity of the *individual* neurons over time) [35]. Furthermore, they require a common predefined sparseness level for all inputs, which is quite an unlikely requirement.

We thus pursue an alternative approach for overcomplete sparse coding which (1) minimizes the number of costly bottom-up signal transmission (metabolic efficiency) (2) can be implemented in a neurally plausible fashion and (3) does not require a predefined sparsity level. Let us remark that the focus is now on forming sparse representations, so we do not discuss methods to learn the feature set (but see, for example, [4,36,37,1] on this matter). Furthermore, it should be noted that adaptation (parameter tuning of the transformations) can also contribute to the decrease of metabolic cost. LOCOCODE based methods (e.g. [38]) explicitly takes into account the complexity of the transformations (the coding/decoding process) and searches for those parameter sets that yield representations with low reconstruction error and low-complexity coding. Another similarity is that such methods do not require the a priori knowledge of features ('causes'). However, our method has been designed with the explicit goal to handle overcomplete and strictly sparse codes while the method of [38] can yield local as well as dense codes. In addition, it is not quite obvious how that model scales with overcompleteness.

2. Methods

In this section we shortly review the cross-entropy method (CEM), which is provably convergent, is very efficient in probabilistic combinatorial optimizations and can be used for explicit sparsification by minimizing in ℓ_0-norm. Finally, we propose our integrated solution, which inherits the favorable properties of both SP and CEM.

2.1. Cross-entropy method (CEM)

CEM is a randomized search based global optimization technique [39] aiming to find the solution in the following form:

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

where f is a general objective function. CEM resembles the estimation-of-distribution evolutionary methods (see e.g. [40]). As a global optimization method, it provably converges to the optimal solution [39,40]. This method has been successfully applied in different problems like optimal buffer allocation [41], DNA sequence alignment [42], reinforcement learning [43] or independent process analysis [44].

While most optimization algorithms maintain a single candidate solution $\mathbf{y}(t)$ at each iteration, CEM maintains a *distribution* over possible solutions. From this distribution, solution candidates are drawn at random. By continuous modification of the sampling distribution, random guess becomes a very efficient optimization method.

One may start by drawing many samples from a fixed distribution g and then selects the best sample as an estimation of the optimum. The efficiency of random guess depends to a great extent on the distribution g from which the samples are drawn. After drawing a moderate number of samples from distribution g, we may not be able to give an acceptable approximation of $\mathbf{y}^*$, but we may still obtain a *better sampling distribution*. The basic idea of the CE method is that it selects the best few samples (the 'elite'), and modifies g so that it becomes more similar (as measured by cross-entropy, also called Kullback–Leibler divergence) to the empirical distribution of the selected samples.

For many parameterized distribution families, the parameters of the minimum cross-entropy member can be computed easily from simple statistics of the elite samples. For completeness we provide the formulae and the corresponding pseudocode for independent Bernoulli distributions in the Appendix, as these will be needed in the sparse feature subset optimization. It may be worth noting that the independent Bernoulli distribution over the sparse features can be seen as an independent prior over the sources of a generative model with ℓ_0 norm. Derivations as well as a list of other discrete and continuous distributions with simple update rules can be found in [45].

2.2. The SP-CE method

Although SP is a promising approach, it has its own shortcomings: the heavy use of the costly, bottom-up transformation of the residuals at each iteration and the preset sparsity level κ. On the other hand, CE which is less affected by the costly bottom-up transformations, updates the probability of all active components similarly, regardless their individual contributions to the actual reconstruction error. In turn, we propose an improvement over both methods which inherits the flexibility and synaptic efficiency of CEM and the superior speed and scaling properties of SP without their shortcomings. The improved model can be realized by inserting an intermediate control step in CEM to individually update the component probabilities based on their contribution to the reconstruction error. Hence the explicit refinement of SP is replaced by an implicit modification through the component probabilities. The pseudocode of the combined method is given in Table 1.

As the resulting algorithm is not a greedy method we call it as subspace cross-entropy (SP-CE) method without the term 'Pursuit'. In comparison with the algorithms of the original online CEM [18, Fig. 1a] and the SP method there are two major differences. With regard to the online CE method, SP-CE takes advantage of the SP idea and updates the Bernoulli distribution with the help of the reconstruction error: the update of the probability of the basis functions is proportional to their contribution to the representation of the reconstruction error. With regard to the SP method, SP-CE is flexible: it is not restricted to k elements. In order to keep the ℓ_0-norm of the representation around k, probability vector $\mathbf{p}$ is normalized to k. At every iteration SP-CE draws random samples according to $\mathbf{p}$. The resulting elite samples may have more than k1s, so SP-CE is not restricted by parameter k. Furthermore, k may be adjusted through the λ trade-off parameter of cost function Eq. (4) as it controls the balance between sparseness and reconstruction quality. For simplicity such adaptive tuning is not included in the pseudocode (Table 1).

Table 1
Pseudo-code of the subspace cross-entropy (SP-CE) method for Bernoulli distributions. For more details, see technical report [46].

```
required:
  p = (p_1, ..., p_m)            % initial distribution parameters
  k                               % approximate value for k
initialize: SP and CE
for t from 1 to t_SP              % SP iteration main loop
  for τ from 0 to t_CE − 1,       % CE iteration main loop
    execute CE iteration
  output: K                       % CE optimized index set
  r ← x − D[K]D[K]^† x            % compute next residual
  if ||r^t||_2 ≥ ||r^{t−1}||_2 then   % check for improvement
quit
  else:
    stochastic update
for CE using the
residual
    e ← D^T r                     % BU step of SP
                                    (i_1, ..., i_j ..., i_m) ← MaxInd_m[e]
% ordered index set of e
    p'_{i_j} ← exp(−j/k)          % auxiliary Bernoulli distribution
                                    with ≈ k number of 1 s on average
    p' ← p + ||r||_2 p'           % weigh by residual's norm
                                    to improve distribution
  p ← k p'/||p'||_1               % normalize for k to draw
                                    k number of 1s on average
end loop
```

In the proposed setting k becomes a soft, tunable parameter that depends on the actual input and expectations of the system. In the pseudocode we suggest using an exponential function for normalization based on the projection of the residual on the components. The exponential function may be replaced by neural non-linear contrast enhancement algorithms (for an early reference, see, [47]) like soft winner-take-all methods (WTA) often used for competitive learning in neural networks in tasks like distributed decision making, pattern recognition or modeling attention [48,49]. WTA networks use mutual inhibition to select the largest (or for k-WTA the first k largest) elements of the input set. The sharp non-linearity of the k-WTA procedure corresponds to the SP expansion method (Table 6).

3. Simulation results

We performed extensive computer simulations in order to compare the performance of the combined algorithm to some well established solutions, including direct linear programming methods ('ℓ_1-magic' [23,50]) and SP of [16]. To see the impact of SP on CEM, we also included the results of a standard batch CEM implementation as we have already shown the equivalence of the online and batch versions of CEM (in terms of global convergence and accuracy) [18]. In order to provide unbiased comparison, we tested these methods on a noise free synthetic problem for which the optimal solution is always known. For each comparison 10 reconstruction matrices were generated. Matrix elements were drawn randomly from Gaussian distribution with zero mean and unit variance then the columns were normalized to 1. For each matrix 10 distinct random binary vectors were selected as sparse internal representations (coefficients). Input signals were then generated as products of the internal representations and the

matrices. Results are averaged over the 100 ($10 \cdot 10$) runs. The task is to reconstruct the signals by estimating the sparse coefficients. The dimension of the input, the dimension of the overcomplete representation and the number of non-zero components in the representation ($\dim(\mathbf{x})$, $\dim(\mathbf{a})$ and k, respectively) are the main factors that determine the computational complexity of the test. We studied the run time and the reconstruction quality as a function of overcompleteness and sparseness. All simulations were run on a single core PC. Note that we did not exploit the parallel nature of the batch CEM, which would add an enormous speed-up in run time over serial methods (but of course at the same metabolic cost). Instead, all algorithms were implemented in Matlab (Mathworks, Natick, MA) and run in the same environment. Table 2 summarizes the parameters of the algorithms.

In Fig. 1 two graphs are shown to characterize the performance of ℓ_1-magic, SP, CEM and SP-CE algorithms as a function of overcompleteness while keeping sparsity and input size fixed ($k=8$, $n=64$). Note that CEM should find the optimal solution if allowed to run long enough. On the other hand, methods based on the ℓ_0–ℓ_1 norm equivalence are expected to fail when overcompleteness is above the breakdown point. For our synthetic problem the breakdown point, that is when the equivalence of ℓ_1 and ℓ_0 norms does not hold anymore, which is about at $n \sim 4k \log(m)$ [23].

Table 2
Parameters of the different algorithms. k is the number of non-zero components. The stopping criteria were either the maximum iteration number or the reconstruction error. We used the default setting for 'ℓ_1-magic' [50]. SP was implemented as given in [16]. Parameters of batch CEM and SP-CE methods were optimized for these tests by hand.

Algorithms	Parameters	Values
ℓ_1-Magic	Primal dual tolerance	0.001
	Max. number of primal–dual iterations	50
SP	Parameter free (except k)	–
CE	Reconstruction error when CE stops	$\epsilon = 0.1/k$
	Elite ratio	$\rho = 0.05$
	Update factor for p	$\alpha = 0.1$
	Number of batch samplings	100
	Sample size in one batch	500
SP-CE	Reconstruction error when CE stops	$\epsilon = 0.1/k$
	Elite ratio	$\rho = 0.05$
	Update factor for p	$\alpha = 0.9$
	Number of batch samplings	6
	Sample size in one batch	100

The first subplot in Fig. 1(a) shows the mean run time on logarithmic scale as a function of the size of the internal sparse representation. The logarithmic scale shows the fine structure of dependencies, but distorts deviations for which they are not plotted. Note the order differences among the methods. The second plot (Fig. 1(b)) depicts the ratio of good solutions as a function of the size of the internal sparse representation. In this case deviations are not shown as the difficulty of the different problems differ due to the different random dictionaries. For both plots some illustrative std values are presented in Table 4 of the Appendix. The methods fail to provide good solutions if they reach the maximum iteration number without converging to a solution or converge into an incorrect solution. What is interesting though is that all methods show graceful degradation of performance even for internal dimensions on the order of 1000. Nonetheless, there are clear advantages for the CE based methods for very sparse representations: SP and ℓ_1-magic methods reach 'critical sparseness' [16] much earlier. The SP-CE/CEM advantage could be further increased by allowing slower convergence. Last, but not least, both ℓ_1-magic and SP assume that the number of non-zero components is given beforehand, which is not needed for CEM and SP-CE.

The reconstruction performance of SP-CE is the same as that of CEM within measurement error. However, SP-CE runs orders of magnitudes faster than CEM and ℓ_1-magic. The speed of SP-CE is due to two factors. One factor is that SP-CE is using relatively few time consuming bottom-up transformations. The other factor is the principled introduction of mutual influence between the probability update of the individual components that incorporates the advantages of the SP method. In comparison with pure SP, SP-CE is only an order of magnitude slower than SP.

The impact of complexity on the different methods has been tested by changing the number of non-zero components in the overcomplete representations. Fig. 2 summarizes the results. Here again the running time (Fig. 2(a)) and the ratio of good solutions

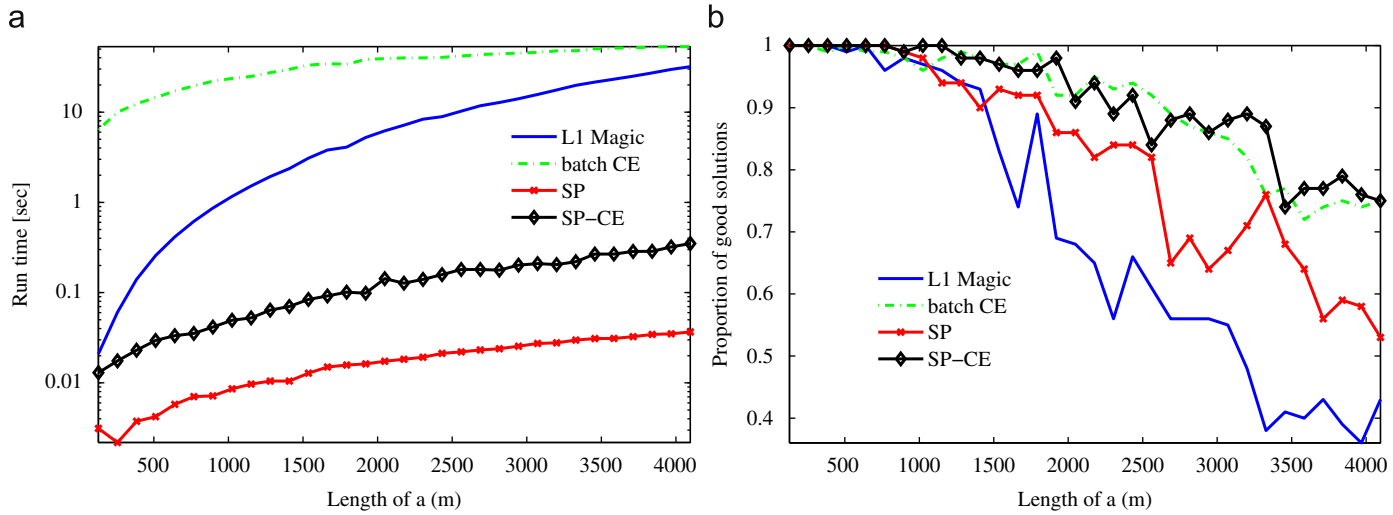

Fig. 1. Performance and run time of different sparse coding methods as a function of the size of the representation. Compared methods: ℓ_1-magic, (batch) CEM, SP and subspace CE (SP-CE). (a) Mean run time for the different methods as a function of overcompleteness, i.e., the size of the internal representation on logarithmic scale. Size of input: $\dim(\mathbf{x}) = 64$, number of active components: $k=8$. (b) Ratio of good solutions (ratio of found good components) as a function of overcompleteness. Deviations are not plotted due to the various complexity stemming from the different dictionaries. For both plots some illustrative std values are shown in Table 4 of the Appendix. Note that the true k value is needed and has been provided to ℓ_1-norm and SP methods. The strict value of this number is not needed for CEM and SP-CE methods.

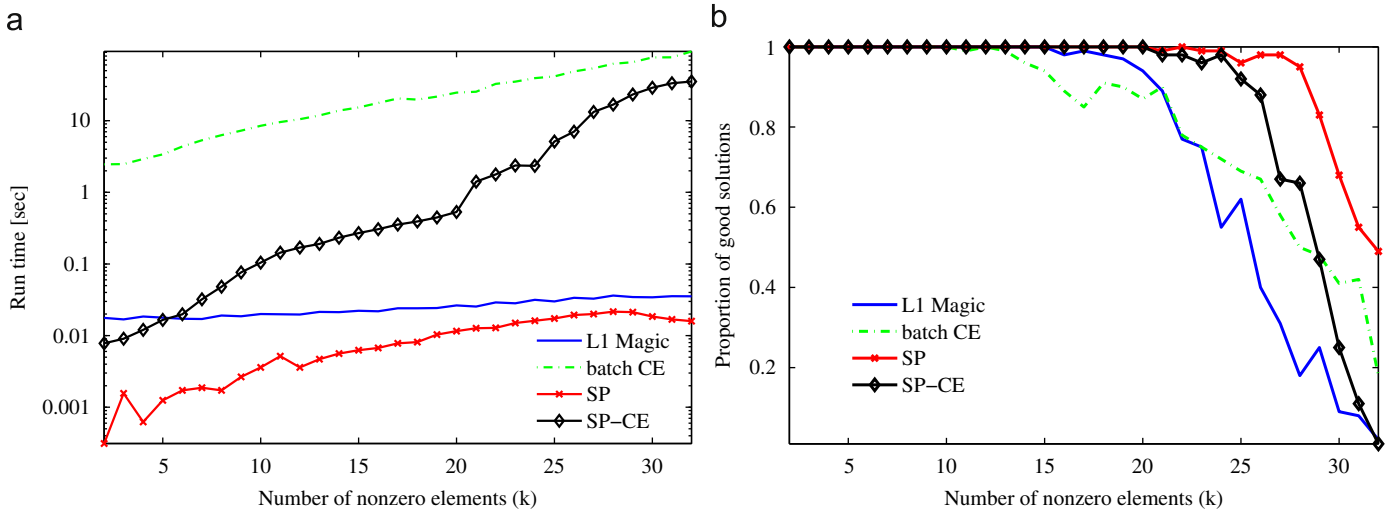

Fig. 2. Performance and run time of different sparse coding methods as a function of the number of non-zero elements of the representation. Compared methods: ℓ_1-magic, (batch) CEM, SP and subspace CE (SP-CE). (a) Run time for the different methods as a function of sparseness, i.e., the number of non-zero elements of the internal representation. Size of input: $\dim(\mathbf{x})=64$, size of the overcomplete representation: $\dim(\mathbf{a})=128$. (b) Ratio of good solutions (reconstruction quality) as a function of overcompleteness. Similar to the previous tests ℓ_1-norm and SP methods used the true value of k. The strict value of this number is not needed for CE and SP-CE methods. For both plots some illustrative std values are shown in Table 5 in the Appendix.

Table 3
Comparison of approximate complexity of the different algorithms. Complexity can be expressed as the number of elementary flops since matrix vector products and pseudoinverse calculations determine cost. For very sparse problems, CEM is the most favorable solution. Let us note that the second term comes from the pseudoinverse calculation, while the third term is for the probability update. As sparsity decreases, the number of iterations within CE also increases. While the method is convergent, its convergence speed may vary. The same applies to SP-CE, which is a CEM variant. This approximation for SP-CE assumes that for each input the representation is modified about k times by sending the rec. error.

ℓ_1-Magic	$O(n^2 \cdot m)$
SP	$O(k \cdot n \cdot m)$
CE	$O(n \cdot m)+k \cdot n+m)$
SP-CE	$O(k \cdot n \cdot m+k \cdot n+m)$

(Fig. 2(b)) are plotted, but now as a function of the number of non-zero components. In this test SP seems superior, but the combined method still performs well (increased number of iterations would allow even better performance at the price of longer run time). Furthermore, as the level of overcompleteness increases, SP requires increasingly more iteration for calculating the reconstruction error, thus resulting in an enormous number of additional synaptic transmissions. This disadvantage does not show up in run time using Matlab on a single PC since Matlab is optimized for matrix-vector computations. Let us remark that performance of SP may quickly deteriorate if the number of non-zero components is not known beforehand, which is the typical case in real scenarios. Finally, an approximate comparison of the computational complexity is given in Table 3, based on the values of [25]. As an example, a detailed derivation of the complexity of the so called interior point methods [25, Chapter 11.5.5] indicates that scaling of the best linear programming solution is still $\sim O(m^3)$.

4. Discussion

According to the expectations brain inspired computations are going to dominate and shape technology in the near future. While the most compelling features of neural systems are (1) extremely low energy cost, (2) massively parallel and distributed organization and (3) outstanding resilience (robustness against structural and functional perturbations, self-organizing mechanisms for self-repair, etc), mainstream digital solutions mostly focus on speed, are organized in a serial fashion and even small scale perturbations can make them malfunction or totally dysfunctional. In order to make a paradigm shift, we need to have a better understanding of the structural and functional constituents of the neural systems responsible for the development and maintenance of these features. In this article we presented a very efficient signal processing approach designed to form overcomplete, sparse representations. In theoretical neuroscience the usefulness of such representations has long been recognized (e.g. [51]) for its contribution to increased storage capacity, better pattern separation and higher noise tolerance.

Recent discoveries in signal processing now suggest another important factor that must be utilized by evolutionary solutions as well. For most natural signals (i.e., those that are encountered most frequently by living systems) very efficient *sampling* is possible by exploiting their intrinsic sparseness. The fact that signals can be recovered by a small set of samples has serious impact on computational speed as well as on noise tolerance since even partially observed signals can also be recovered. A few novel models [52,53,33] have already started to implement these ideas in neural computations. These studies either deal with learning efficient dictionaries (in terms of reconstruction quality) or draw parallels between sensory processing and compressed sensing [22,23]. In this article we argued that efficiency in terms of metabolic cost should also be taken into account when considering sparse coding mechanisms. Since this constraint is equally important for artificial and natural systems we wanted to improve ℓ_1-norm based solutions by explicitly minimizing the required number of signal transmission (e.g. by reducing the number of signal projections and comparisons, especially when bottom-up transmissions are needed). In contrast to ℓ_1-norm based methods, our solution, which is designed to explicitly minimize the active number of components (i.e., the ℓ_0 norm), can be efficiently implemented in a parallel, distributed system. Furthermore, CEM ensures convergence to the optimal solution [39,40], which may not be the case for ℓ_1-norm based solutions if conditions are not met. What remains to be seen whether the SP-CE algorithm can be implemented in a neurally plausible form.

4.1. Neural plausibility of the SP-CE method

To discuss whether our algorithm makes sense in neural terms, implementation and functional issues need to be considered. As the implementation is concerned, although a detailed translation of the model into equations used by computational neuroscience is missing, we conjecture that SP-CE with online CEM [18] can be expressed in neuronal form. An important property of the online variant of CEM is that it preserves the convergence properties of the original batch method [46] although it relies on threshold adaptation and needs to identify elite samples and update the probabilities at each iteration. Since thresholds of individual neurons and neuronal ensembles can be independently and locally adapted and the discrete time updates of the algorithm can be rephrased in differential forms similar to those describing the changes of membrane potential we see no obstacles to translate the whole procedure into neural form. The artificial neural network will have an input layer a layer for sparse representation that holds the activities of the features and a copy of this layer that updates the probabilities of the sparse components. We note that online CEM has two global parameters [46], which are similar to soft-thresholding.

As it was demonstrated our solution is indeed an efficient sparse coding solution with local rules supporting distributed implementations. However, SP based update, while considerably accelerating the computations, does not seem to support lifetime sparseness (at least transiently). This shortcoming may, for example, be compensated by feedforward inhibitory thresholding [33].

Soft-WTA selecting approximately k components is another algorithm that we include into our computations. In SP-CE, its role is to keep the number of active components low and to update the probabilities of the components. WTA methods have been already proposed as a computational primitive in the neural system. On a particular use of the more restrictive k-WTA in modeling the hippocampal computations, see [54], whereas for a recent paper on the computational power of neural WTA methods, see e.g. [55].

The proposed algorithm uses two distinct quantities: the probability of firing (feature selection) and the activity that reconstructs the input (coefficients of the features). Whether this dichotomy can be mapped onto assumed forms of neural computations is an open question. There are many potential solutions, ranging from local neuronal circuits, through considerations on population codes [56,57] to models that neurons themselves represent multi-layer networks exhibiting a range of linear and non-linear mechanisms [58]. Future work will focus on this issue.

It has been long debated if the nature of neocortical sensory processing is mostly feedforward or feedback, since even the first spikes after stimulus onset may carry enough information in e.g. recognition tasks [59]. While most arguments are about the expected speed of information processing, the capacity of the generative networks or the impact of anatomical and physiological constraints (see e.g. [60–64]), we find that a particular combination of feedforward (bottom-up) and feedback (top-down) processing provides the most favorable solution for sparse coding in terms of *metabolic cost*. Given that the statistics of natural signals supports sparse representations, economical solutions, like this combined process, may have evolutionary advantage over purely feedforward or feedback solutions.

In respect to the criticisms of [33] against previously proposed neural sparse coding mechanisms our solution (1) is neurally plausible (using decentralized and local interactions at low synaptic activity), (2) can generate exactly zero-valued coefficients (since it directly minimizes the number of active components) and (3) it is based on a principled approach as opposed to heuristic approximations applied in other proposals. However, the fourth issue about non-smooth variations in the coefficients for time-varying stimuli has not yet been addressed. Its importance stems from the belief that prediction is probably the most fundamental task of the neural system [65]. Hence any model claiming to explain some aspects of neural representation should provide means to support this central task. Currently we are seeking methods to make the proposed algorithm applicable for predicting time-varying sparse signals. Future work will show if the emerging probabilistic representations can be used e.g. in a belief propagation network thus supporting prediction.

Because all algorithmic steps can be realized by local interactions keeping a low energy profile, we argue that the emerging architecture might be a good model of neural sparse coding in sensory information processing. Although the generation of candidate representations and the update of the component probabilities require recurrent interactions, the constraint to minimize reentrant activity (i.e., reducing the intra-layer synaptic activity) results in apparent and approximately correct fast feed-forward working.

Beyond the low metabolic costs of the proposed algorithm, the ℓ_0-norm based optimization provides additional flexibility over simple ℓ_1-norm optimizations by not fixing the number of active components beforehand. Methods based on the ℓ_1-norm (like SP) may fail if the optimal sparsity for a given stimuli differs from the predefined value. We note that for all tests ℓ_1-norm based optimization schemes would have provided very poor approximations had they been restricted to keep a single ratio of non-zero components and the size of the representation (Fig. 1) or to keep a single number of non-zero components (Fig. 2).

5. Conclusions

We have proposed a novel solution for sparse coding by combining our previous work [18] on spike based probabilistic optimization using ℓ_0-norm with an efficient iterative method. The motivation behind this integration is that greedy methods are computationally attractive and also provide the optimal solution under certain conditions [22,50] while our ℓ_0-norm based solution using the probabilistic cross-entropy method provides a potential neural interpretation and does not require the exact knowledge of the number of non-zero components.

The combined method has several outstanding features making it an interesting candidate for neural sparse coding in the sensory systems: computations are local, distributed and 'transmission efficient' with low metabolic cost. In addition, the algorithm is less affected by constraints of other alternatives based on ℓ_1-norm optimization.

Acknowledgement

We thank Zoltán Szabó for his helpful comments on the manuscript. We are also grateful for the reviewers forhelping us presenting our ideas more clearly. This research has been partially supported by the EC NEST 'Percept' Grant under contract 043261, by the European Union, co-funded by the European Social Fund and the European Regional Development Fund. Project ID numbers: TAMOP 4.2.1./B-09/KMR-2010-0003, KMOP-1.1.2-08/1-2008-0002.

Appendix A

In this section a few selected std values—corresponding to Figs. 1 and 2 are presented first. We then give a formal description

of our notations. Finally, we provide the pseudocodes for the subspace pursuit and the cross-entropy methods.

A.1. Illustrative mean and std values for the different tests

A.2. Notations

We use calligraphic letters for index series: $\mathcal{J} \coloneqq \{(i_1,\ldots,i_j) : i_1,\ldots,i_j \in \{1,\ldots,m\}, j \le m\}$ where the corresponding plain capital letter denotes the size of the index series and indices are between 1 and m.

For any $\mathbb{Z}_+ \ni k (\le m)$ and index series $\mathcal{K} = (i_1,\ldots,i_k)$ (where $i_p \in \{1,\ldots,m\}$ and $i_p \ne i_q$ if $p \ne q$) let $\mathbf{a}[\mathcal{K}]$ denote the $\mathcal{K}$-indexed components of $\mathbf{a} \in \mathbb{R}^m$, i.e., $\mathbf{a}[\mathcal{K}] = (a_{i_1},\ldots,a_{i_k})^T$. Similarly, for matrix $\mathbf{D} \in \mathbb{R}^{n\times m}$ and for series $\mathcal{K} = (i_1,\ldots,i_k)$ ($i_p \ne i_q \in \{1,\ldots,m\}$), let $\mathbf{D}[\mathcal{K}] = (\mathbf{d}_{i_1},\ldots,\mathbf{d}_{i_k})$.

For a vector $\mathbf{a} \in \mathbb{R}^m$ let $\texttt{MaxInd}_k(\mathbf{a})$ ('ordered index series') denote the indices of the components ranked by their amplitude, i.e., $\texttt{MaxInd}_k(\mathbf{a}) = (i_1,\ldots,i_k)$ with $|a_{i_p}| \ge |a_{i_q}|$ for all $i_p < i_q$ and $p,q \in \{1,\ldots,k\}$. Accordingly, we will call $\mathbf{D}[\texttt{MaxInd}_k(\mathbf{a})]$ as 'k-truncated matrix'.

Pseudo-inverse of matrix $\mathbf{D}$ is defined as $\mathbf{D}^\dagger = (\mathbf{D}^T\mathbf{D})^{-1}\mathbf{D}^T$ where T stands for transposition. A single superscript for matrices and vectors denotes the iteration number.

A.3. Pseudocodes

One of the core algorithms in our solution is a variant of the subspace pursuit method [16]. Table 6 shows the corresponding pseudocode.

Table 4
Illustrative mean ($\pm$std) values of run time and success ratio corresponding to Fig. 1. The first line shows the mean ($\pm$std) run time, while the second one shows the mean ($\pm$std) ratio of success.

Method	m		
	512	2048	4096
ℓ_1-Magic	0.2573 ± 0.0300 0.9900 ± 0.1000	6.2003 ± 1.4352 0.6800 ± 0.4688	31.9430 ± 7.0378 0.4300 ± 0.4976
(batch) CEM	14.4706 ± 2.6599 1.0000 ± 0	39.1136 ± 8.2313 0.9200 ± 0.2727	53.1178 ± 11.2299 0.7500 ± 0.4352
SP	0.0042 ± 0.0070 1.0000 ± 0	0.0173 ± 0.0054 0.8600 ± 0.3487	0.0367 ± 0.0075 0.5300 ± 0.5016
SP-CE	0.0295 ± 0.0187 1.0000 ± 0	0.1428 ± 0.0821 0.9100 ± 0.2876	0.3498 ± 0.2018 0.7500 ± 0.4352

Table 5
Illustrative mean ($\pm$std) values of run time and success ratio corresponding to Fig. 2. The first line shows the mean ($\pm$std) run time, while the second one shows the mean ($\pm$std) ratio of success.

Method	k		
	15	21	27
ℓ_1-Magic	0.0222 ± 0.0078 1.0000 ± 0	0.0255 ± 0.0082 0.8900 ± 0.3145	0.0328 ± 0.0085 0.3100 ± 0.4648
(batch) CEM	15.3194 ± 6.9074 0.9400 ± 0.2387	25.3781 ± 11.8578 0.9000 ± 0.3015	53.9633 ± 23.3658 0.5800 ± 0.4960
SP	0.0063 ± 0.0077 1.0000 ± 0	0.0127 ± 0.0073 0.9900 ± 0.1000	0.0200 ± 0.0077 0.9800 ± 0.1407
SP-CE	0.2703 ± 0.0743 1.0000 ± 0	1.4113 ± 3.9941 0.9800 ± 0.1407	13.2244 ± 12.1605 0.6700 ± 0.4726

Table 6
The pseudocode of the subspace pursuit method of [16]. The goal is to represent the input with minimal reconstruction error using k basis only. SP differs from other iterative greedy methods in the incremental refinement of the selected basis subset. First, a representation is generated with the help of the full basis set (using pseudoinverse computations). During iteration k basis are selected based on the amplitude of the corresponding coordinates of the representation. The resulting residual (difference between the original input and the approximation obtained by projecting the representation onto the input space) is then again projected back to the representation space and another set of k basis are chosen. The two selected subsets are then fused (*expansion*) and the resulting expanded set is used again to project the original input onto the representation space. Finally, a new set of k basis are selected by the amplitude of the corresponding coordinates of the projection (*shrinkage*). Iteration stops when the norm of the residual does not decrease anymore.

`input:`	
$\kappa = k/m$, $\mathbf{x} \in \mathbf{R}^n$	% sparsity and signal
t_{SP}	% max iteration number
$\mathbf{D} \in \mathbb{R}^{n\times m}$	% m column dictionary
`initialization:`	
$\mathcal{K} = \texttt{MaxInd}_k(\mathbf{D}^T\mathbf{x})$	% index selection (see text)
$\mathbf{D} = \mathbf{D}[\mathcal{K}]$	% truncated basis set
$\mathbf{r} \leftarrow \mathbf{x} - \mathbf{D}\mathbf{D}^\dagger\mathbf{x}$	% compute residual
`optimization:`	
`for` t `from` 1 `to` t_{SP}	% iteration loop
$\texttt{computeMaxInd}_k(\mathbf{D}^T\mathbf{r})$	% index set for expansion
$\mathcal{K} \leftarrow \mathcal{K} \cup \texttt{MaxInd}_k(\mathbf{D}^T\mathbf{r})$	% increase set size to $2k$)
$\mathbf{e} \leftarrow \mathbf{D}[\mathcal{K}]^\dagger\mathbf{x}$	% compute projections
$\mathcal{K} \leftarrow \texttt{MaxInd}_k(\mathbf{e})$	% new index set of size k
$\mathbf{D} \leftarrow \mathbf{D}[\mathcal{K}]$	% k-truncated matrix
$\mathbf{r}^t \leftarrow \mathbf{x} - \mathbf{D}\mathbf{D}^\dagger\mathbf{x}$	% compute residual
`if` $\mathbf{r}^t = 0$ `then quit`	% finish is residual is zero
`if` $\|\mathbf{r}^t\|_2 \ge \|\mathbf{r}^{t-1}\|_2$ `then`	% check for improvement
$t = t_{SP}$	% no new iteration
$\mathcal{K}^{t_{SP}} = \mathcal{K}^{t-1}$	% use previous index set
`quit`	
`end loop`	
`output:`	
$\mathcal{K}^{t_{SP}}$	% optimized sparse index set

Table 7
Pseudo-code of CEM for Bernoulli distributions.

`required:`	
$\mathbf{p} = (p_1,\ldots,p_M)$	% initial distribution parameters
T,	% max iteration number
I	% population size
ρ	% selection ratio
`for` t `from` 1 `to` T	% CE iteration main loop
`for` i `from` 1 `to` I	
`draw` $\mathbf{y}^{(i)}$ `from` $BER^M(\mathbf{p}_t)$	% draw I samples
`compute` $f^i \coloneqq f(\mathbf{y}^{(i)})$	% evaluate them
`sort` f^i `-values`	% order by decreasing magnitude
$\gamma \leftarrow f_{\rho\cdot I}$	% level set threshold
$L_\gamma \leftarrow \{\mathbf{y}^{(i)} \mid f(\mathbf{y}^{(i)}) \le \gamma\}$	% get elite samples
$p'_j \leftarrow (\sum_{\mathbf{y}^{(i)} \in L_\gamma} \chi(y_j^{(i)} = 1))/(\rho \cdot I)$	% get parameters of nearest distrib
$p_j \leftarrow \alpha \cdot p'_j + (1-\alpha) \cdot p_j$	% update with step-size α
`end loop`	

The other building block of our combined heuristics is CEM for Bernoulli distribution. It works as follows. Let the domain of optimization be $\mathbf{y} \in \mathcal{Y} = \{0,1\}^M$, and each component be drawn from independent Bernoulli distributions, i.e., $\mathcal{G} = BER^M$. Each distribution $g \in \mathcal{G}$ is parameterized with an M-dimensional vector $\mathbf{p} = (p_1,\ldots,p_M)$. Component j of the sample $\mathbf{y} \in \mathbf{y} \in \mathcal{Y}$ drawn from distribution g thus will be

$$y_j = \begin{cases} 1 & \text{with probability } p_j \\ 0 & \text{with probability } 1-p_j \end{cases}$$

Having drawn I samples $\mathbf{y}^{(1)},\ldots,\mathbf{y}^{(I)}$, let L_γ denote the level set of the elite samples, i.e.,

$$L_\gamma := \{\mathbf{y}^{(i)} | f(\mathbf{y}^{(i)}) \le \gamma\}$$

where γ denotes a fixed threshold value. The distribution g' with minimum cross-entropy distance from the uniform distribution over the elite set has the following parameters [45]:

$$\mathbf{p}' := (p'_1,\ldots,p'_M)$$

where

$$p'_j := \frac{\sum_{\mathbf{y}^{(i)} \in L_\gamma} \chi(y_j^{(i)} = 1)}{\sum_{\mathbf{y}^{(i)} \in L_\gamma} 1} = \frac{\sum_{\mathbf{y}^{(i)} \in L_\gamma} \chi(y_j^{(i)} = 1)}{\rho \cdot I} \tag{6}$$

where $\chi(\cdot)$ is an indicator function. In other words, the parameters of g' are simply the componentwise empirical probabilities of 1s in the elite set. Changing the distribution parameters from $\mathbf{p}$ to $\mathbf{p}'$ can be too coarse, so in some cases, applying a step-size parameter α is preferable. The resulting algorithm is summarized in Table 7.

References

[1] J. Mairal, F. Bach, J. Ponce, G. Sapiro, Online learning for matrix factorization and sparse coding, J. Mach. Learn. Res. 11 (2010) 19–60.
[2] J. Schmidhuber, Learning factorial codes by predictability minimization, Neural Comput. 4 (1992) 863–879.
[3] A.B. Lee, K.S. Pedersen, D.B. Mumford, The nonlinear statistics of high-contrast patches in natural images, Int. J. Comput. Vision 54 (2003) 83–103.
[4] B.A. Olshausen, D.J. Field, Sparse coding with an overcomplete basis set: a strategy employed by V1? Vision Res. 37 (1997) 3311–3325.
[5] B.A. Olshausen, D.J. Field, Sparse coding of sensory inputs, Curr. Opinion Neurobiol. 14 (2004) 481–487.
[6] E.C. Smith, M.S. Lewicki, Efficient auditory coding, Nature 439 (2006) 978–982.
[7] H.B. Barlow, Possible principles underlying the transformation of sensory messages, in: W.A. Rosenblith (Ed.), Sensory Communication, MIT Press, Cambridge, MA, 1961, pp. 217–234.
[8] D.J. Field, What is the goal of sensory coding, Neural Comput. 6 (1994) 559–601.
[9] D. Mumford, Neuronal architectures for pattern-theoretic problems, in: C. Koch, J.L. Davis (Eds.), Large Scale Neuronal Theories of the Brain, MIT Press, 1994, pp. 125–152.
[10] D.D. Lee, H.S. Seung, Learning the parts of objects by non-negative matrix factorization, Nature 401 (1999) 788–791.
[11] B.A. Olshausen, Sparse codes and spikes, in: R.P.N. Rao, B.A. Olshausen, M.S. Lewicki (Eds.), Probabilistic Models of the Brain, MIT Press, 2002, pp. 257–272.
[12] W. Vinje, J. Gallant, Natural stimulation of the nonclassical receptive field increases information transmission efficiency in v1, J. Neurosci. 22 (2002) 2904–2915.
[13] P. Berkes, B. White, J. Fiser, No evidence for active sparsification in the visual cortex, in: Y. Bengio, D. Schuurmans, J. Lafferty, C.K.I. Williams, A. Culotta (Eds.), Advances in Neural Information Processing Systems, vol. 22, 2009, pp. 108–116.
[14] D.J. Graham, D.J. Field, Sparse coding in the neocortex, in: J.H. Kaas, L.A. Krubitzer (Eds.), Evolution of Nervous Systems, vol. III, Elsevier, 2006, pp. 181–187.
[15] B. Natarajan, Sparse approximate solutions to linear systems, SIAM J. Comput. 24 (1995) 227–234.
[16] W. Dai, O. Milenkovic, Subspace pursuit for compressive sensing signal reconstruction, IEEE Trans. Inf. Theory 55 (2009) 2230–2249.
[17] R. Rubinstein, D.P. Kroese, The Cross-Entropy Method, Springer-Verlag, New York, 2004.
[18] A. Lőrincz, Z. Palotai, G. Szirtes, Spike-based cross-entropy method for reconstruction, Neurocomputing 71 (2008) 3635–3639.
[19] R. Tibshirani, Regression shrinkage and selection via the lasso, J. R. Statist. Soc. B 58 (1996) 267–288.
[20] S. Chen, D. Donoho, M. Saunders, Atomic decomposition by basis pursuit, SIAM J. Sci. Comput. 43 (2001) 129–159.
[21] D.L. Donoho, M. Elad, Optimally sparse representation in general nonorthogonal dictionaries via l1 minimization, Proc. Natl. Acad. Sci. USA 100 (2003) 2197–2202.
[22] D.L. Donoho, Compressed sensing, IEEE Trans. Inf. Theory 52 (2006) 1289–1306.
[23] E. Candès, J. Romberg, T. Tao, Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information, IEEE Trans. Inf. Theory 52 (2006) 489–509.
[24] T. Tao, Preprints in Sparse Recovery/Compressed Sensing, 2008 ⟨http://www.math.ucla.edu/~tao/preprints/sparse.html⟩.
[25] S. Boyd, L. Vandenberghe, Convex Optimization, Cambridge University Press, 2004.
[26] S. Kim, K. Koh, M. Lustig, S. Boyd, D. Gorinevsky, An interior-point method for large-scale l1-regularized logistic regression, J. Mach. Learn. Res. 8 (2007) 1519–1555.
[27] J.A. Tropp, Greed is good: algorithmic results for sparse approximation, IEEE Trans. Inf. Theory 50 (2004) 2231–2242.
[28] D. Needell, R. Vershynin, Uniform uncertainty principle and signal recovery via regularized orthogonal matching pursuit, Found. Comput. Math. 9 (2009) 317–334.
[29] S. Mallat, Z. Zhang, Matching pursuits with time-frequency dictionaries, IEEE Trans. Signal Process. 41 (1993) 3397–3415.
[30] B. Mailhe, R. Gribonval, F. Bimbot, P. Vandergheynst, A low complexity orthogonal matching pursuit for sparse signal approximation with shift-invariant dictionaries, in: ICASSP'09, IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 3445–3448.
[31] D. Donoho, Y. Tsaig, I. Drori, F. Starck, Sparse Solution of Underdetermined Linear Equations by Stagewise Orthogonal Matching Pursuit, Technical Report 2006-02, Stanford University, Stanford, CA, 2006.
[32] D. Needell, J.A. Tropp, CoSaMP: iterative signal recovery from incomplete and inaccurate samples, Appl. Comput. Harmonic Anal. 26 (2008) 301–321.
[33] C.J. Rozell, D.H. Johnson, E.G. Baraniuk, B.A. Olshausen, Sparse coding via thresholding and local competition in neural circuits, Neural Comput. 20 (2008) 2526–2563.
[34] P. Lennie, The cost of cortical computation, Curr. Biol. 13 (2003) 493–497.
[35] B. Willmore, D. Tolhurst, Characterising the sparseness of neural codes, Network: Comput. Neural Syst. 12 (2001) 255–270.
[36] H. Lee, A. Battle, R. Raina, A.Y. Ng, Efficient sparse coding algorithms, in: Advances in Neural Information Processing Systems, NIPS, 2007, pp. 801–808.
[37] Y. Weiss, H. Chang, W. Freeman, Learning Compressed Sensing, 2007.
[38] S. Hochreiter, J. Schmidhuber, Feature extraction through LOCOCODE, Neural Comput. 11 (1999) 679–714.
[39] R.Y. Rubinstein, The cross-entropy method for combinatorial and continuous optimization, Methodol. Comput. Appl. Probab. 2 (1999) 127–190.
[40] H. Muehlenbein, The equation for response to selection and its use for prediction, Evol. Comput. 5 (1998) 303–346.
[41] G. Allon, D.P. Kroese, T. Raviv, R.Y. Rubinstein, Application of the cross-entropy method to the buffer allocation problem in a simulation-based environment, Ann. Oper. Res. 134 (2005) 137–151.
[42] J. Keith, D.P. Kroese, Sequence alignment by rare event simulation, in: Proceedings of the 2002 Winter Simulation Conference, pp. 320–327.
[43] I. Szita, A. Lőrincz, Learning to play using low-complexity rule-based policies: illustrations through Ms. Pac-Man, J. Artif. Intell. Res. 30 (2007) 659–684.
[44] Z. Szabó, B. Póczos, A. Lőrincz, Cross-entropy optimization for independent process analysis. in: Independent Component Analysis and Blind Signal Separation, Lecture Notes in Computer Science, vol. 3889, Springer, 2006, pp. 909–916.
[45] P.-T. de Boer, D.P. Kroese, S. Mannor, R.Y. Rubinstein, A tutorial on the cross-entropy method, Ann. Oper. Res. 134 (2004) 19–67.
[46] I. Szita, A. Lőrincz, Online Variants of the Cross-Entropy Method, 2008 ⟨http://arxiv.org/abs/0801.1988⟩.
[47] G.A. Carpenter, S. Grossberg, ART2: self-organization of stable category recognition codes for analog input patterns, Appl. Opt. 26 (1987) 4919–4930.
[48] G. Carpenter, S. Grossberg, A massively parallel architecture for a self-organizing neural pattern recognition machine, Comput. Vision Graphics Image Process. 37 (1987) 54–115.
[49] M. Riesenhuber, T. Poggio, Hierarchical models of object recognition in cortex, Nat. Neurosci. 2 (1999) 1019–1025.
[50] W. Candès, J. Romberg, l_1-Magic: A Collection of Matlab Routines for Solving the Convex Optimization Programs Central to Compressive Sampling, online, 2006 ⟨http://www.acm.caltech.edu/l1magic/⟩.
[51] B.A. Olshausen, D.J. Field, Emergence of simple-cell receptive field properties by learning a sparse code for natural images, Nature 381 (1996) 607–609.
[52] M. Rehn, F.T. Sommer, A network that uses few active neurones to code visual input predicts the diverse shapes of cortical receptive fields, J. Comput. Neurosci. 22 (2007) 135–146.
[53] W.K. Coulter, C.J. Hillar, G. Isley, F.T. Sommer. Adaptive compressed sensing—a new class of self-organizing coding models for neuroscience, In: Proceedings of ICASSP'2010, pp. 5494–5497, doi:10.1109/ICASSP.2010.5495209.
[54] R. O'Reilly, J. McClelland, Hippocampal conjunctive encoding, storage, and recall: avoiding a tradeoff, Hippocampus 4 (1994) 661–682.
[55] W. Maass, On the computational power of winner-take-all, Neural Comput. 12 (2000) 2519–2535.
[56] A. Pouget, P. Dayan, R. Zemel, Information processing with population codes, Nat. Rev. Neurosci. 1 (2000) 125–132.
[57] W.J. Ma, J.M. Beck, P.E. Latham, A. Pouget, Bayesian inference with probabilistic population codes, Nat. Neurosci. 9 (2006) 1432–1438.
[58] M. London, M. Häusser, Dendritic computation, Ann. Rev. Neurosci. 28 (2005) 503–532.
[59] S.J. Thorpe, How can the human visual system process a natural scene in under 150ms? Experiments and neural network models, in: Eurorean Symposium on Artificial Neural Networks, D-Facto public, 1997.

[60] R.P.N. Rao, D.H. Ballard, Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects, Nat. Neurosci. 2 (1999) 79–87.
[61] C. Koch, T. Poggio, Predicting the visual world: silence is golden, Nat. Neurosci. 2 (1999) 9–10.
[62] A.M. Sillito, J. Cudeiro, H.E. Jones, Always returning: feedback and sensory processing in visual cortex and thalamus, Trends Neurosci. 29 (2006) 307–316.
[63] T. Serre, A. Oliva, T. Poggio, A feedforward architecture accounts for rapid categorization, Proc. Natl. Acad. Sci. 104 (2007) 6424–6429.
[64] H. Liu, Y. Agam, J.R. Madsen, G. Kreiman, Timing, timing, timing: fast decoding of object information from intracranial field potentials in human visual cortex, Neuron 62 (2009) 281–290.
[65] R. Llinas, I of the Vortex: From Neurons to Self, The MIT Press, 2002.

András Lőrincz is the head of the Neural Information Processing Group at Eötvös Loránd University, Budapest, Hungary. He is a Fellow of the European Coordinating Committee for Artificial Intelligence. His research interests are in artificial general intelligence, cognition and neuroscience. In particular he is working on the development of intelligent systems (human–computer interaction), machine learning and robotics, social cognition and modeling of neural systems (entorhinal–hippocampal complex and basal ganglia). He has published more than 200 peer-reviewed scientific papers and the number of independent references to his works is above 800.

Zsolt Palotai obtained his M.Sc. in Technical Informatics in 2003 at Budapest University of Technology and Economics. He has won twice the second and once the third prize in the Hungarian Student Research Competition in Informatics between 2001 and 2003. He obtained his Ph.D. degree in Computer Sciences in 2008 in the group of András Lőrincz at Eötvös Loránd University, Budapest. His recent research interests are in distributed computation, sparse representation learning, machine learning and artificial neural networks.

Gábor Szirtres received M.Sc. in Chemistry in 1999 and Ph.D. in Computer Sciences in 2005 at Eötvös Loránd University, Budapest, Hungary. He has worked with Kenneth R. Miller at UCSF, San Francisco and Columbia University, New York on theoretical neuroscience. He is currently working with Andras Lorincz on a functional model of the entorhinal–hippocampal complex.