Biological Cybernetics, 108(1):61–73, 2014.

Classification using sparse representations: a biologically plausible approach

M. W. Spratling
King's College London, Department of Informatics, London. UK.

Abstract

Representing signals as linear combinations of basis vectors sparsely selected from an overcomplete dictionary has proven to be advantageous for many applications in pattern recognition, machine learning, signal processing, and computer vision. While this approach was originally inspired by insights into cortical information processing, biologically-plausible approaches have been limited to exploring the functionality of early sensory processing in the brain, while more practical applications have employed non-biologically-plausible sparse-coding algorithms. Here, a biologically-plausible algorithm is proposed that can be applied to practical problems. This algorithm is evaluated using standard benchmark tasks in the domain of pattern classification, and its performance is compared to a wide range of alternative algorithms that are widely used in signal and image processing. The results show that, for the classification tasks performed here, the proposed method is competitive with the best of the alternative algorithms that have been evaluated. This demonstrates that classification using sparse representations can be performed in a neurally-plausible manner, and hence, that this mechanism of classification might be exploited by the brain.

Keywords: sparse coding; classification; predictive coding; neural networks; pattern recognition.

1 Introduction

Many methods in signal processing and pattern recognition seek to represent a signal $\mathbf{x}$ as a linear combination of basis vectors selected from a dictionary, $\mathbf{V}$, in proportion to a set of coefficients, $\mathbf{y}$, *i.e.*,:

$$\mathbf{x} \approx \mathbf{V}\mathbf{y}$$

Well-known examples include vector quantisation (VQ), principal components analysis (PCA), independent components analysis (ICA), and non-negative matrix factorisation (NMF). These methods differ in the types of constraints placed on the factors $\mathbf{V}$ and $\mathbf{y}$. In addition, there are often several different algorithms which have been proposed to implement each method, which differ in the details of how the factors are found.

Partially inspired by research on efficient coding in the primary visual cortex (Olshausen and Field, 1997, 2004), recently there has been much interest in representing signals using a *sparse* set of basis vectors selected from an *overcomplete* dictionary (see Pece, 2002; Tošić and Frossard, 2011; Wright et al., 2009a, for reviews). In vision, sparse representations have been applied to image compression (*e.g.*, Aharon et al., 2006; Fischer et al., 2006, 2007; Murray and Kreutz-Delgado, 2006; Pece and Petkov, 2000) and image restoration (*e.g.*, Elad and Aharon, 2006; Hyvarinen et al., 1998; Mairal et al., 2009). Sparse representations are also increasingly popular for classification (*e.g.*, Bociu and Pitas, 2004; Elhamifar and Vidal, 2011; Jiang et al., 2013; Kang et al., 2011; Mairal et al., 2008a; Ramirez et al., 2010; Sprechmann and Sapiro, 2010; Wright et al., 2009b; Zhang et al., 2013), often with results that are competitive with the state-of-the-art. As with the non-sparse, non-overcomplete, case there are many algorithms that have been proposed for finding sparse representations. These algorithms differ in terms of the constraints placed on the factors $\mathbf{V}$ and $\mathbf{y}$, the method used to measure and constrain the sparsity of $\mathbf{y}$, as well as in details of the algorithm used to search for appropriate factors.

The elements of the vector $\mathbf{y}$ define a set of coefficients that accurately represent the signal $\mathbf{x}$ through a mixture of basis vectors taken from the dictionary $\mathbf{V}$. For sparse coding algorithms, the vector $\mathbf{y}$ must also be sparse, meaning that each signal is represented by a small number of non-zero coefficients. Many

algorithms ("sparse solvers") have been proposed for finding appropriate values of $\mathbf{y}$ given a dictionary and a signal. Most practical methods fall into one of two classes: greedy pursuit algorithms, or convex relaxation algorithms (Tropp and Wright, 2010).

The columns of the matrix $\mathbf{V}$ comprise basis vectors (or "elementary components", or "codebook entries", or "atoms", or "dictionary elements"), that are appropriate for the efficient representation of all the signals that occur in the particular task domain under consideration. The appropriate entries in the dictionary, $\mathbf{V}$, will thus depend on the (typically large) set of signals that need to be represented. Dictionaries are typically learnt from the data or are predefined. In the case of dictionary learning for classification tasks, separate sub-dictionaries might be learnt for each class (Ramirez et al., 2010; Sprechmann and Sapiro, 2010), or a single dictionary might be learnt for all classes (Tošić and Frossard, 2011). In the case of dictionary predefinition, the dictionary might be defined using a generic set of basis vectors, like Gabor functions (Yang and Zhang, 2010), or the dictionary might be defined to consist of a set of signals taken from the training set (Wright et al., 2009b).

This article advocates a biologically plausible method for finding sparse representations. Many previous algorithms for finding sparse representations have been presented as models of cortical information processing (*e.g.*, Bell and Sejnowski, 1997; Berkes et al., 2009; Charles et al., 2012; Falconbridge et al., 2006; Földiák, 1990; Hamker and Wiltschut, 2007; Harpur, 1997; Hoyer, 2003, 2004; Hoyer and Hyvärinen, 2000; Jehee and Ballard, 2009; King et al., 2013; Liu and Jia, 2012; Lücke, 2009; Olshausen and Field, 1996; Perrinet, 2010; Rao and Ballard, 1999; Rehn and Sommer, 2007; Rozell et al., 2008; Van Hateren and van der Schaaf, 1998; Weber and Triesch, 2008; Wiltschut and Hamker, 2009; Zhu and Rozell, 2013; Zylberberg et al., 2011). Hence, several of these previous algorithms are also biologically plausible. However, in most cases these algorithms have been applied to only a single task: learning basis vectors that resemble the receptive fields (RFs) of cells in primary visual cortex (V1) when trained on patches taken from natural images. Hence, previous work on biologically-plausible sparse coding methods has concentrated on dictionary learning in a single domain. Here, the emphasis is on showing that a biologically-plausible algorithm has the potential to model a wider range of cortical processes underlying a wider range of tasks, and that a biologically-plausible algorithm can be used for practical applications.

The proposed, biologically-plausible, algorithm is called `PC/BC-DIM`. Previous work on `PC/BC-DIM` has concentrated on dictionary learning, and has shown that the `PC/BC-DIM` algorithm (with appropriate learning rules) can learn dictionaries that can be used to efficiently represent natural images (Spratling, 2012c), face images (Spratling et al., 2009), and artificial datasets used to benchmark machine learning algorithms (Spratling, 2012c; Spratling et al., 2009). Here, to allow direct comparison with other sparse solvers, `PC/BC-DIM` is used to find sparse representations defined over a predefined dictionary. It is shown that the classification performance of `PC/BC-DIM` is competitive with the best performance of the non-biological sparse solvers that were tested.

2 Methods

2.1 Sparse Solvers

Given a dictionary, $\mathbf{V}$, we wish to find a sparse set of coefficients, $\mathbf{y}$, that can accurately reconstruct a signal $\mathbf{x}$ from the dictionary entries. Numerous algorithms for solving such sparse representation problems have been proposed. The performance of the proposed algorithm (`PC/BC-DIM`, see section 2.2) was compared to a representative set of these existing algorithms taken from three publicly available toolboxes: SMALLbox[1] (Damnjanovic et al., 2010), SparseLab[2], and L1Benchmark[3] (Yang et al., 2010). The algorithms that were tested are listed in Table 1. The tested methods include a number of different implementations of greedy algorithms constraining the ℓ_0-pseudo-norm of the coefficients (`MP`,

[1] http://small-project.eu/software-data/smallbox/
[2] http://sparselab.stanford.edu/
[3] http://www.eecs.berkeley.edu/~yang/software/l1benchmark/

Toolbox	Acronym	Description
	PC/BC-DIM	The proposed algorithm
SMALLbox	MP	Matching Pursuit
	OMP	Orthogonal Matching Pursuit with Cholesky updates
	PCGP	Partial Conjugate Gradient Pursuit
SparseLab	PDBBP	Primal-Dual Barrier Method for solving Basis Pursuit
	LARS+	Least Angle Regression
	LARS-lasso+	Least Angle Regression with the Lasso modification
	PFP+	Polytope Faces Pursuit
L1Benchmark	Homotopy+	Homotopy Method for solving Basis Pursuit Denoising
	SpaRSA+	Sparse Reconstruction by Separable Approximation
	FISTA	Fast Iterative Shrinkage-Threshold Algorithm
	DALM	Dual Augmented Lagrange Multiplier

Table 1: The set of algorithms for finding sparse representations that are tested in this article.

OMP, PCGP, LARS+), greedy algorithms constraining the ℓ_1-norm of the coefficients (LARS-lasso+, PFP+), and a number of different convex optimisation algorithms for finding solutions with the ℓ_1-norm constraint on the coefficients (PDBBP, Homotopy+, SpaRSA+, FISTA, and DALM).

In each case, the algorithm was tested using default parameter settings. Where user-defined parameter values were mandatory (MP, OMP, PCGP), these were set to those values that gave the best results for the first classification task (USPS)[4]. There parameters are therefore likely to be sub-optimal for the other tasks. However, this is also the case for all the other tested algorithms. It would be possible to tune the parameters of each algorithm to each task, however, the results would then likely favour the algorithm with the most free parameters. The motivation for using a fixed set of parameters with each algorithm, is to test how well algorithms perform across tasks. Since all the datasets used to test the algorithms were inherently non-negative, where the algorithm supported it, the coefficients were constrained to be non-negative: this is indicated by the plus-sign at the end of the algorithm name[5]. The proposed algorithm, PC/BC-DIM, is also constrained to find non-negative coefficients.

2.2 The Proposed Sparse Solver

In terms of neural networks, the coefficients, $\mathbf{y}$, can be interpreted as the activation strengths of a set of neurons which have generative (or top-down) connections with weights equal to the set of basis functions, $\mathbf{V}$ (see Fig. 1). An appropriate set of neural activations can then be used to reconstruct, via the generative weights, the input to the neural network $\mathbf{x}$. Here we define a neural network algorithm that can be used to find sparse representations, $\mathbf{y}$ that accurately represent the input. The proposed algorithm is called Divisive Input Modulation (DIM) which is the main computational building block underlying the PC/BC model of cortical function (De Meyer and Spratling, 2011, 2013; Spratling, 2008, 2010, 2011, 2012a,b,c). PC/BC-DIM is implemented using the following equations:

$$\mathbf{e} = \mathbf{x} \oslash (\epsilon_2 + \mathbf{V}\mathbf{y}) \tag{1}$$

$$\mathbf{y} \leftarrow (\epsilon_1 + \mathbf{y}) \otimes \mathbf{W}\mathbf{e} \tag{2}$$

[4] All three algorithms required the same two parameters to be defined: the maximum allowable representation error, and the maximum number of coefficients that could take non-zero values. The former was set to 1×10^{-5} for all three algorithms. The latter parameter was set to 8, 5, and 20, for MP, OMP, and PCGP respectively (only values greater than one were considered, to prevent the algorithm becoming identical to a nearest neighbour classifier, see section 2.3).

[5] Note that for LARS+ and LARS-lasso+ the coefficients that are found by the solver still contain negative values, despite the constraint that the solution should be non-negative.

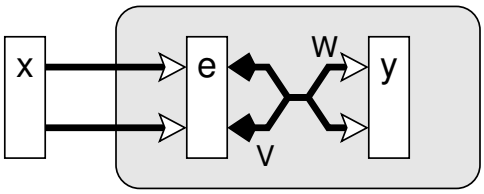


Figure 1: The PC/BC−DIM neural network architecture. Rectangles represent populations of neurons, with $\mathbf{y}$ labelling the population of prediction neurons and $\mathbf{e}$ labelling the population of error-detecting neurons. Open arrows signify excitatory connections, filled arrows indicate inhibitory connections, crossed connections signify a many-to-many connectivity pattern between the neurons in two populations, and parallel connections indicate a one-to-one mapping of connections between the neurons in two populations. The feedback (or generative) weights, labelled $\mathbf{V}$, correspond to a dictionary of basis vectors, and the activity of the prediction neurons, $\mathbf{y}$, correspond to the coefficients of the sparse representation. The feedforward (or discriminative) weights, labelled $\mathbf{W}$, are equal to a rescaled transpose of the feedback weights. Hence, if a pair of neurons in the $\mathbf{e}$ and $\mathbf{y}$ populations are connected by strong (or weak) feedforward weights, then the reciprocal feedback connection is also strong (or weak).

Where $\mathbf{e}$ is a (m by 1) vector of error-detecting neuron activations; $\mathbf{y}$ is a (n by 1) vector of prediction neuron activations; $\mathbf{W}$ is a (n by m) matrix of feedforward synaptic weight values; $\mathbf{V}$ is a (m by n) matrix of feedback synaptic weight values; ϵ_1 and ϵ_2 are parameters; and $\oslash$ and $\otimes$ indicate element-wise division and multiplication respectively (addition operations are also performed element-wise). The matrix $\mathbf{V}$ is the dictionary, hence, the feedback connections from each prediction neuron represent a single basis vector. The columns of $\mathbf{V}$ are normalised to have a maximum value of one. The matrix $\mathbf{W}$ is equal to the transpose of the $\mathbf{V}$ and has each row normalised to sum to one. Hence, the feedforward and feedback weights are simply rescaled versions of each other.

Initially the values of $\mathbf{y}$ are all set to zero, although random initialisation of the prediction node activations can also be used with little influence on the results. Equations 1 and 2 are then iteratively updated with the new values of $\mathbf{y}$ calculated by equation 2 substituted into equations 1 and 2 to recursively calculate the neural activations. This iterative process was terminated when the total absolute change in $\mathbf{e}$ between successive iterations was less than 0.04. Parameter values of $\epsilon_1 = 1 \times 10^{-9}$ and $\epsilon_2 = 1 \times 10^{-3}$ were used in all the experiments reported in this article.

The values of $\mathbf{y}$ represent predictions of the causes underlying the inputs to the network. The values of $\mathbf{e}$ represent the residual error between the input reconstructed from these predictions and the actual input, $\mathbf{x}$, to the network. This residual error is calculated, in equation 1, through the divisive comparison of the actual input with the reconstruction of the input, $\mathbf{Vy}$. An element of $\mathbf{e}$ will equal unity when the prediction of that element of the input is correct, and will have a value of less than, or greater than, unity when the prediction is incorrect. Values of $\mathbf{e}$ less than unity, corresponding to elements of the input that are over-represented by the predictions, will tend to reduce the strength of response of the prediction neurons via equation 2, see below. Conversely, values of $\mathbf{e}$ greater than unity, corresponding to elements of the input that are under-represented by the predictions, will tend to increase the strength of response of the corresponding prediction neurons. These effects will tend to drive all error-detecting neuron responses towards a value of unity. ϵ_2 is a constant that prevents division-by-zero errors. It is sufficiently small to have a negligible effect on the calculation of $\mathbf{e}$ except when an element of $\mathbf{x}$ is small. For elements of $\mathbf{x}$ that are smaller than ϵ_2, the corresponding value of $\mathbf{e}$ will always be small, and hence, these inputs are effectively ignored. Hence, the value of ϵ_2 also determines the minimum input activity that is considered significant.

The predictions, $\mathbf{y}$, are updated via equation 2. The first term on the right-hand side of equation 2 means that the new value of a prediction neuron's response will be proportional to its value at the previous iteration. ϵ_1 is a small constant that allows a prediction neuron to become active even if its response

is initially zero. The second term on the right-hand side of equation 2 represents the feedforward input to the prediction neurons from the error-detecting neurons. The strength of this feedforward input will be influenced by the strength of the error neuron outputs (which are in turn influenced by the sensory-driven input to the network and the accuracy of the top-down predictions, as described in the preceding paragraph) and by the similarity between this activity and the synaptic weights, $\mathbf{W}$. Each element of $\mathbf{y}$ will converge to a steady-state when the corresponding element of the vector $\mathbf{We}$ has a value of one (assuming ϵ_1 is sufficiently small to be negligible). The neural responses will converge towards this steady-state since if $\mathbf{We}$ is greater than unity, $\mathbf{y}$ will increase (due to equation 2). This will subsequently decrease the values of $\mathbf{e}$ (via equation 1) which will in turn reduce the value of $\mathbf{We}$. In contrast, if $\mathbf{We}$ is less than unity, $\mathbf{y}$ will decrease (due to equation 2). This will subsequently increase the values of $\mathbf{e}$ (via equation 1) which will in turn increase the value of $\mathbf{We}$.

The activation dynamics described above result in the `PC/BC-DIM` algorithm selecting a sparse subset of active prediction neurons whose RFs (which correspond to basis functions) best explain the underlying causes of the sensory input. The strength of activation reflects the strength with which each basis function is required to be present in order to accurately reconstruct the input. This strength of response also reflects the probability with which that basis function is believed to be present, taking into account the evidence provided by the signal and the full range of alternative explanations encoded in the RFs of the prediction neurons.

Although the `PC/BC-DIM` algorithm does not explicitly enforce sparsity, the activity of the prediction neurons in response to natural image patches is highly sparse and statistically independent (Spratling, 2010, 2012c), images are therefore represented efficiently. Furthermore, although `PC/BC-DIM` does not explicitly minimise the reconstruction error using a global objective function, the activity of the prediction neurons can be used to reconstruct the input with high fidelity (Spratling, 2012c; Spratling et al., 2009), images are thus represented accurately. The results presented in section 3 demonstrate that the reconstruction error and the sparsity of `PC/BC-DIM` are competitive with the best of the other sparse solvers tested on the classification tasks considered here.

2.3 Data Preprocessing and Dictionary Definition

Comparison of the algorithms for finding sparse representations was carried out using three standard datasets: the USPS and MNIST[6] hand-written digit datasets, and the ISOLET[7] audio dataset. All three tasks define separate training and testing sets. These standard splits were used for the experiments performed here. For USPS the training set contains 7291 exemplars and the testing set contains 2007 exemplars. For ISOLET the training set contains 6238 exemplars and the testing set contains 1559 exemplars. For MNIST the training set contains 60000 exemplars and the test set contains 10000 exemplars. For each dataset, in addition to testing performance on the test set, classification performance was also tested using the training set. In each case performance was tested using the first 2000 exemplars from the training set.

For a fair comparison of different algorithms it is necessary to test each algorithm using the same dictionary. It would be possible to learn the dictionary, but this would require using a specific algorithm for finding sparse representations during the learning process. The results might therefore conceivably be biased in favour of the method used to learn the dictionary. To avoid this issue, the dictionary is created using the training data itself, following the method used by Wright et al. (2009b). Such a dictionary may not be optimal for discrimination between classes (Jiang et al., 2011; Sprechmann and Sapiro, 2010; Zhang et al., 2013), and hence, there may be scope for improving classification performance beyond the results reported here. However, the aim here is to compare the `PC/BC-DIM` algorithm with other sparse solvers, rather than produce state-of-the-art classification performance.

For each data set, as a whole, the data was rescaled to be in the range $[0, 1]$. Hence, the data contained only non-negative values. Each individual signal in the data was then rescaled to have unit ℓ_1-norm, ℓ_2-

[6] http://yann.lecun.com/exdb/mnist/

[7] http://archive.ics.uci.edu/ml/datasets/ISOLET

norm, or ℓ_∞-norm (depending on the algorithm under test, see below). The signals forming the training set were then used to define the dictionary entries used for performing the classification: each column of $\mathbf{V}$ was a (normalised) exemplar from the training set. For USPS and ISOLET all training exemplars were used to form the dictionary. To make the task more manageable, only the first 2000 training exemplars for each digit of the MNIST dataset were used to define the dictionary. Hence, for this task only 20000 of the 60000 training exemplars were employed. These dictionaries composed of exemplars from the training data, as described immediately above, were used for all the experiments described in sections 3.1 to 3.3.

In all previous work using the `PC/BC-DIM` algorithm the input $\mathbf{x}$ has been allowed to range between zero and one, and the columns of $\mathbf{V}$ have been normalised to have a maximum value of one (*i.e.*, to have unit ℓ_∞-norm). Previous work on sparse coding has constrained the magnitude of the dictionary entries and signals using the ℓ_1-norm (*e.g.*, Pece and Petkov, 2000) or, more typically, the ℓ_2-norm (*e.g.*, Elad and Aharon, 2006; Sprechmann and Sapiro, 2010; Thiagarajan and Spanias, 2011; Wright et al., 2009b). Each of the algorithms was therefore tested, using the test set for the first classification task (USPS), to determine which form of normalisation gave the best results. All subsequent experiments were performed using the same method of normalisation. Specifically, the reported results for algorithms `LARS+`, `LARS-lasso+`, `PFP+`, and `FISTA` were all generated using dictionary entries and signals normalised to have unit ℓ_1-norms, while the ℓ_2-norm was used for `MP`, `OMP`, `PCGP`, `Homotopy+`, `SpaRSA+` and `DALM`, and the remaining algorithms were tested using the ℓ_∞-norm. Note that for some algorithms (*i.e.*, `PC/BC-DIM`, `PDBBP`, `PFP+`, `Homotopy+`), results seemed to be little affected by the form of normalisation used, whereas for all other algorithms the classification performance varied widely with the normalisation method.

2.4 Methods of Classification

To compare the classification performance of each algorithm on each dataset the method of classification used by Wright et al. (2009b) was employed. Wright et al. (2009b) showed that this method produced performance comparable to the state-of-the-art in a face recognition task. A sparse representation of the signal was found using the entire dictionary. Each dictionary element is associated with a single class: the class of the training exemplar used to define that dictionary element. The sparse solver defines a coefficient for each dictionary element, hence, each coefficient can also be associated with a single class label. For each class in turn, the coefficients associated with dictionary entries for that class were selected (all other coefficients being set to zero) and the class-specific reconstruction error was calculated. The signal was then allocated to the same class as the sub-set of coefficients that produced the smallest reconstruction error. The performance of each of the tested algorithms using this method of classification are presented in section 3.1.

It is difficult to envisage how the method described above could be neurally implemented. A far simpler, and more biologically-plausible, method would involve reading out the values of $\mathbf{y}$. For example, this could be done by using an additional population of neurons, one neuron for each class. Each of these neurons would receive weighted connections from the neurons representing the $\mathbf{y}$ values, and the class of the signal would be decided by locating the maximum response within this population of classification neurons. To investigate the performance of such a method, classification was also performed by separately summing the coefficient values associated with dictionary entries for a specific class, and assigning the signal to the class that has the highest sum of coefficient values. This is equivalent to using a set of classification neurons with binary weighted connections: with a value of one for every $\mathbf{y}$ value associated with a specific class, and zero for all other $\mathbf{y}$ values. Such a linear decoding of the coefficients is similar to the method used by Jiang et al. (2011, 2013), except here the dictionary and decoding weights are predefined using the training data and class labels, rather than being learnt. The performance of each of the tested algorithms using this method of classification are presented in section 3.2.

Sprechmann and Sapiro (2010) investigated the classification performance of a sparse dictionary learning algorithm on the same three datasets used here. This algorithm produced results comparable to the state-of-the-art for these particular tasks (Sprechmann and Sapiro, 2010). Classification was

performed by splitting the dictionary into separate sub-dictionaries each of which contained all the basis vectors corresponding to a single class. A sparse representation of the signal was found *separately* using each class-specific sub-dictionary and the signal was allocated to the same class as the sub-dictionary that minimised a measure of representation quality. The representation quality used by Sprechmann and Sapiro (2010) was a function of both the reconstruction error and the sparsity (the ℓ_1-norm of the coefficients). For simplicity, in this article the signal was allocated to the same class as the sub-dictionary that produced the smallest reconstruction error, and results for this method of classification are presented in section 3.3. When sparse representations are calculated separately for different sub-dictionaries, classification based on the strength of the response of the coefficients, as discussed in the previous paragraph, fails completely. This is because each sub-dictionary will generate strong responses to each signal in order to try to minimise the reconstruction error. However, reading out the responses of the error-detecting units associated with separate sub-dictionaries could be done in a neurally-plausible manner.

All three classification methods described above become identical to a nearest neighbour (1-NN) classifier when the following two conditions are met:

- when the dictionary is composed of all the exemplars from the training set, and
- when the sparsity of the representation generated by the solver is constrained to be maximum.

When the second condition is true, the sparse solver returns only one non-zero coefficient corresponding to the single, most similar, dictionary element. The signal is therefore assigned to the same class as this dictionary element, which if the first condition is true, is the class of the most similar training exemplar. When the sparse solver is allowed to find more than one non-zero coefficient, the classification methods described above are *not* equivalent to a k-nearest neighbour (k-NN) classifier. This is because classification with the sparse solver depends on the combination of dictionary elements that can best explain the signal, which is not necessarily the k most similar elements. For comparison with the sparse dictionary based classification results, the performance of k-NN classifiers, with various values of k, are provided in Fig. 4.

2.5 Dictionary Learning

To ensure that the results presented here are not specific to the case where the dictionary is composed of exemplars from the training data, experiments were repeated with learnt dictionaries. Many algorithms for learning sparse representations have been proposed both in the computational neuroscience literature (*e.g.*, Bell and Sejnowski, 1997; Berkes et al., 2009; Charles et al., 2012; Falconbridge et al., 2006; Földiák, 1990; Hamker and Wiltschut, 2007; Harpur, 1997; Hoyer, 2003, 2004; Hoyer and Hyvärinen, 2000; Jehee and Ballard, 2009; King et al., 2013; Liu and Jia, 2012; Lücke, 2009; Olshausen and Field, 1996; Perrinet, 2010; Rao and Ballard, 1999; Rehn and Sommer, 2007; Rozell et al., 2008; Spratling, 2012c; Van Hateren and van der Schaaf, 1998; Weber and Triesch, 2008; Wiltschut and Hamker, 2009; Zylberberg et al., 2011), and in the machine learning literature (*e.g.*, Aharon et al., 2006; Engan et al., 2007; Jiang et al., 2011, 2013; Lemme et al., 2010; Mairal et al., 2012, 2010, 2008a,b, 2007; Murray and Kreutz-Delgado, 2006; Plumbley, 2007; Ramirez et al., 2010; Sprechmann and Sapiro, 2010; Thiagarajan and Spanias, 2011; Tošić and Frossard, 2011; Wei et al., 2013; Yang et al., 2011; Zhang et al., 2013). Here the ILS-DLA algorithm (Engan et al., 2007) was used. This algorithm employs two steps which are iteratively performed to learn a dictionary. In the first step the dictionary is kept constant and a sparse solver is used to find the coefficients that are required to encode the training data using the current dictionary. In the second step, the dictionary elements are updated, as follows:

$$\mathbf{V} = (\mathbf{X}\mathbf{Y}^T)(\mathbf{Y}\mathbf{Y}^T)^{-1}$$

Where $\mathbf{X}$ is an m by p matrix of training data arranged such that each column corresponds to one exemplar, and $\mathbf{Y}$ is an n by p matrix of coefficients each column of which contains the coefficients found

by the sparse solver to represent the corresponding column of $\mathbf{X}$. For the inverse of $\mathbf{Y}\mathbf{Y}^T$ to exist, it is necessary for the coefficient corresponding to each dictionary element to be non-zero for at least one training exemplar. If this was not the case, following the method described by Engan et al. (2007), that dictionary element was replaced by the training exemplar that had the highest reconstruction error. The dictionary was initialised to consist of the first n exemplars from the training set (Engan et al., 2007), and 25 epochs of this learning procedure were performed. Separate class-specific sub-dictionaries were learnt with $n = 100$ for each sub-dictionary.

To test a particular sparse solver, that algorithm was used in the learning procedure described above, and then the same sparse solver was used to perform classification using the learnt dictionary. The classification performance was tested by assigning the signal to the same class as the sub-dictionary that produced the smallest reconstruction error, similar to the method proposed by Sprechmann and Sapiro (2010), and identical to the third classification method described in section 2.4. In the experiments performed using dictionaries defined using exemplars from the training data, each algorithm was tested using an identical dictionary. In contrast, for the experiments performed on learnt dictionaries, testing is carried out using different dictionaries for each algorithm. However, these learnt dictionaries have all been defined using the same learning procedure. The performance of each of the tested algorithms using this method of classification are presented in section 3.4.

2.6 Code

Software, written in MATLAB, which performs the experiments described in this article is available from: `http://www.corinet.org/mike/Code/sparse_classification.zip`.

3 Results

3.1 Classification based on reconstruction error with a single dictionary

Each of the algorithms under test was applied to classification tasks following the method used in (Wright et al., 2009b) for facial image recognition. Specifically, a dictionary was defined using exemplars taken from the training set (see section 2.3), and a sparse representation of a test signal was found with this dictionary using one of the sparse solvers (see sections 2.1 and 2.2). A test signal was assigned to the same class as the class-specific sub-set of coefficients that produced the smallest reconstruction error (see section 2.4). The percentage of signals that were misclassified by each algorithm is shown in the first column of Fig. 2. For easier comparison across all data sets, this same information is also shown in the first column of Fig. 3. It can be seen that the classification performance of the proposed algorithm, `PC/BC-DIM`, is competitive with that of the best of the alternative algorithms tested.

As well as testing the classification performance of each algorithm using the test set for each dataset, classification performance was also assessed using the training sets (results are shown as lighter portions of the bars in the first column of Fig. 2). When the classification performance is evaluated using signals taken from the training set, and the dictionary is defined using all exemplars from the training set, it should be expected that each algorithm will perform perfectly: the sparse solver should generate a maximally sparse representation (a single non-zero coefficient) with zero reconstruction error, by selecting the item in the dictionary that is exactly identical to the input. This perfect performance was produced by most of the tested algorithms, however, surprisingly three algorithms (`Homotopy+`, `SpaRSA+`, and `FISTA`) produced classification errors on the training data. Two further algorithms, `LARS+` and `LARS-lasso+`, crashed when calculating sparse representations for the MNIST training data.

As a baseline against which to compare the results shown in Fig. 3, results for the k-nearest neighbours (k-NN) classifier, using a range of values for k, are shown in Fig. 4. It can be seen that across all three classification tasks a value of k=1 produces the best performance for a nearest neighbours classifier. While several of the sparse coding algorithms produce worse performance than 1-NN, the performance of the best sparse coding algorithms (including the proposed algorithm) is superior to nearest neighbours.

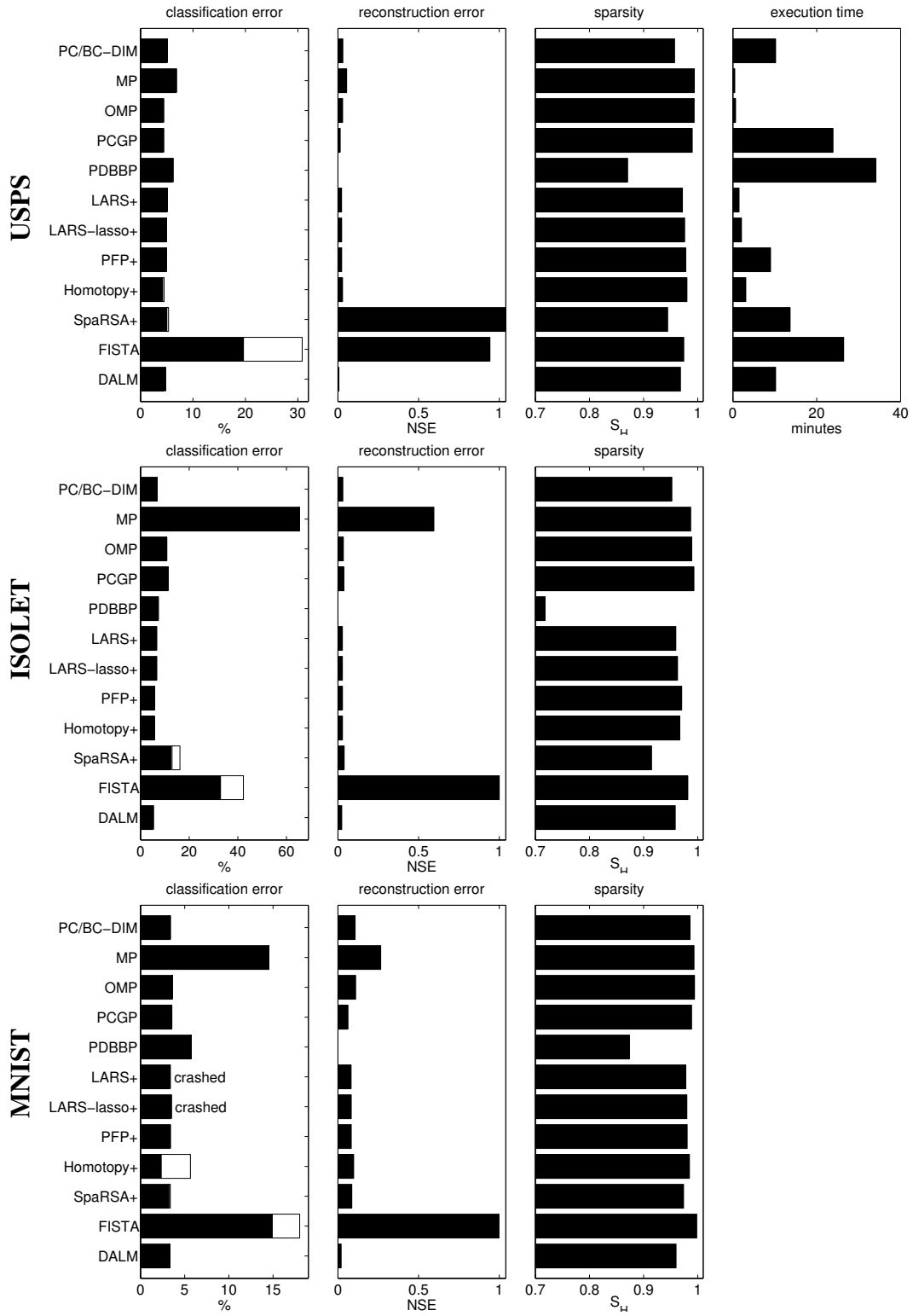


Figure 2: Comparison of the proposed, biologically-plausible, sparse coding algorithm (`PC/BC-DIM`) with 11 alternative algorithms using the USPS (top-row), ISOLET (middle row), and MNIST (bottom-row) datasets. The first column shows the percentage of signals misclassified by each algorithm: the dark part of each bar shows the classification error on the test set, the light part of each bar shows the classification error on the training set. For most algorithms there is no light part of the bar as there was no classification error on the training set. The second column shows the normalised squared error between the signal and the reconstruction of the signal generated by the sparse representation. This value has been averaged across all exemplars in the test set. The third column shows the sparsity of the representation, calculated using Hoyer's measure, averaged across all exemplars in the test set. The fourth column shows the total time taken by each algorithm to process all the exemplars in the test set. The execution time data is only provided for the USPS dataset.

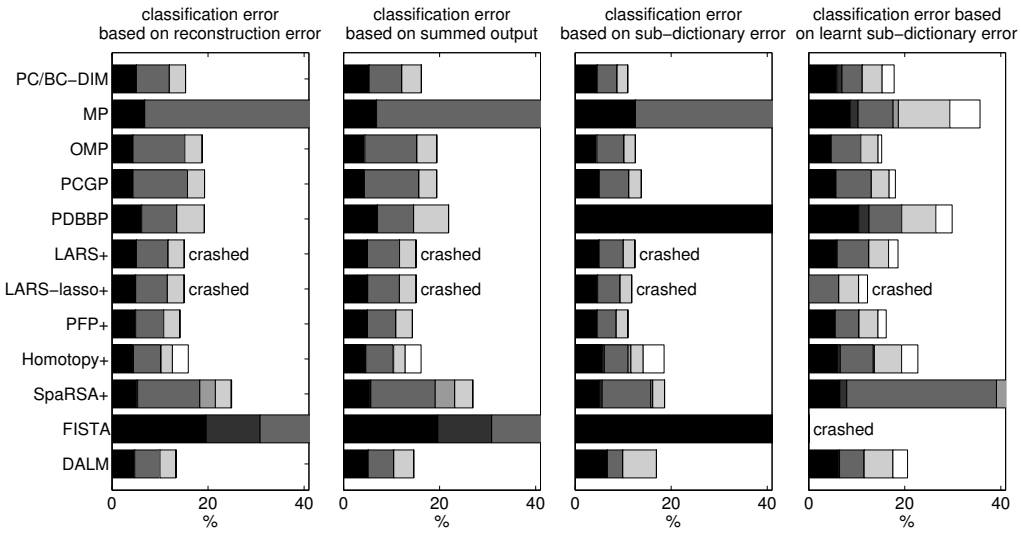


Figure 3: The total classification error over all three datasets for `PC/BC-DIM` and the 11 other algorithms under test. For each bar, the first two shaded segments correspond to the classification error for the USPS dataset, the second two segments show the the classification error for the ISOLET dataset, and the last two segments show the error for the MNIST dataset. In each of these pairs of shaded segments, the classification error on the test set is shown first, and the error on the training set is shown second. Note, however, that for many algorithms the classification error on the training set is zero so the second shaded segment in each pair is not visible. The first column shows the classification error when classification was performed by finding a sparse representation for the entire dictionary and then determining which class-specific sub-set of coefficients produced the lowest reconstruction error: similar to the method used in Wright et al. (2009b). These are the same values as shown in the first column of Fig. 2. The second column shows the classification error when classification was performed by finding a sparse representation for the entire dictionary and then determining which class-specific sub-set of coefficients had the highest sum. The third column shows the classification error when classification was performed by finding sparse representations separately for class-specific sub-dictionaries and determining which sub-dictionary produced the lowest reconstruction error: similar to the method used in Sprechmann and Sapiro (2010). The fourth column shows classification error calculated as for the third column, except in this case the algorithm under test had been used to learn the class-specific sub-dictionaries using the ILS-DLA dictionary learning method (Engan et al., 2007).

The remaining columns of Fig. 2 show additional information about each algorithm. The second column shows the reconstruction error for each algorithm, averaged over all the signals in the test set. For an individual signal the reconstruction error is measured as the normalised squared error (NSE) between the signal, $\mathbf{x}$, and the reconstruction of the signal generated by the sparse code, $\mathbf{Vy}$:

$$\text{NSE} = \frac{\sum(\mathbf{x} - \mathbf{Vy})^2}{\sum \mathbf{x}^2}$$

It can be seen that the reconstruction error for `PC/BC-DIM` is small and comparable to the best of the other algorithms.

The third column shows the sparsity of the representations found by each algorithm, averaged over all the signals in the test set. For each exemplar in the test set the sparsity was measured using the metric proposed by Hoyer (2004):

$$S_H = \frac{\sqrt{n} - \frac{\|\mathbf{y}\|_1}{\|\mathbf{y}\|_2}}{\sqrt{n} - 1}$$

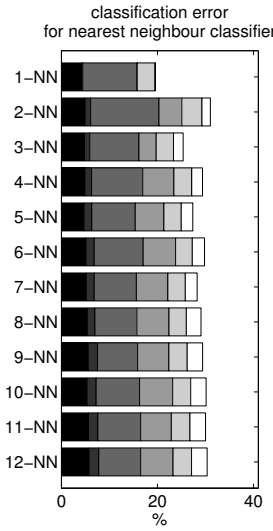


Figure 4: The total classification error over all three datasets for k-nearest neighbours classifiers, with varying values of k. The format of this figure is identical to, and described in the caption of, Fig. 3.

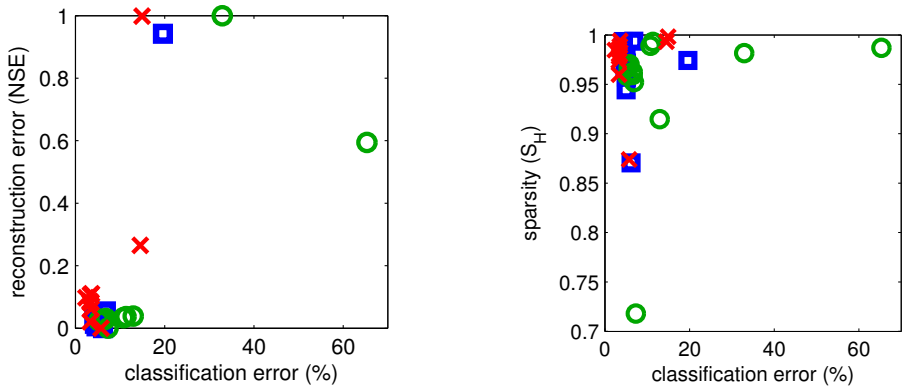
where n is the total number of coefficients (*i.e.*, the length of vector $\mathbf{y}$). S_H ranges from 0 to 1, with higher values corresponding to sparser distributions. It can be seen that the representations found by `PC/BC-DIM` are highly sparse.

The fourth column shows the total time taken by each algorithm to find the sparse coefficients for all exemplars in the test dataset. It excludes the time taken by the code to perform tasks common to testing each algorithm, such as loading the data and analysing the results. Timings were taken by running the code in MATLAB on a machine equipped with 8Gb of RAM and an Intel core i7-920 CPU running at 2.67GHz. Each algorithm was tested in isolation, and hence, timings were (as far as could be controlled) unaffected by other jobs running on the computer and each algorithm was free to utilise multiple cores of the CPU. Due to limitations in the available computational resources and the much longer run-times for the larger datasets, timings under these controlled conditions could only be taken for the first dataset. However, informal observations suggested that the relative speed of each algorithm was similar on the other datasets. The execution time for `PC/BC-DIM` is roughly average for those algorithms tested.

Attempts were also made to test other existing algorithms. Namely, the Iterative Re-Weighted Least Squares algorithm from the SparseLab toolbox and from the L1Benchmark toolbox the Primal-Dual Interior-Point method for solving Basis Pursuit and the ℓ_1-Regularized Least Squares algorithm. All three of these algorithms were at least an order of magnitude slower than any of the results shown in Fig. 2. It was therefore not practicable to fully test these algorithms with the computational resources available. Hence, compared to existing algorithms in general, `PC/BC-DIM` is faster than the results in Fig. 2 suggest.

Across all algorithms there appears to be a weak correlation between classification accuracy and reconstruction error: algorithms that produce small reconstruction error also tend to generate the most accurate classification performance (see Fig. 5a). However, there does not appear to be a correlation between classification accuracy and sparsity: both high and low values of sparsity are produced by algorithms which produce accurate classifications, whereas highly sparse representations lead to both accurate and very poor classification performance (see Fig. 5b).

Similar observations can be made for a single algorithm when the sparsity and reconstruction errors are recorded separately for correctly classified and incorrectly classified signals. Specifically, when using `PC/BC-DIM` to classify the images from the USPS test data, the mean NSE for representations that led to the correct classification was 0.029 while mean NSE for representations that led to the wrong classification was 0.064. Hence, correctly classified exemplars were generally represented more accu-



(a) (b)

Figure 5: Scatter plots showing the relationship between classification error and (a) reconstruction error, and (b) sparsity. Each point shows results from a different sparse-coding algorithm when applied to classifying the test data from each dataset: USPS (squares), ISOLET (circles), and MNIST (crosses). The same results are also shown in Fig. 2.

rately. The mean Hoyer sparsity of representations that led to the correct classification was 0.957, while the mean Hoyer sparsity of representations that led to the wrong classification was 0.973. Hence, wrong classifications were on average more sparse than correct ones. However, this latter result is due to the dictionary containing similar entries that get co-activated by the same stimulus, and hence, produce a more distributed code despite producing a correct classification. When dictionary entries from the same class that had a normalised cross-correlation (NCC) greater than 0.975 were merged into a single average basis vector (reducing dictionary size from 7291 to 5240 entries), the sparsity of test images that were correctly or incorrectly classified became very similar (0.966 versus 0.970). Consistent with previous work (Spratling, 2006; Spratling and Johnson, 2004; Zhang et al., 2011), this latter result calls into question the idea that sparsity is a useful measure of representational quality for pattern recognition.

3.2 Classification based on summed response with a single dictionary

In the results presented above, a sparse representation was found for a single dictionary, and classification was then performed by calculating class-specific reconstruction errors by using only those coefficients corresponding to each class. An alternative, simpler and more biologically-plausible, approach is to simply sum the coefficients corresponding to each class, and assign the signal to the class with the highest sum (see section 2.4). For this alternative method of classification, the percentage classification errors, across all three tasks, are shown in the second column of Fig. 3. By comparing the first two columns of Fig. 3, it can be seen that the results are very similar for these two alternative methods of performing classification using a single dictionary. This suggests that the excellent classification performance of sparse coding could be exploited by biological systems.

3.3 Classification based on reconstruction error with multiple sub-dictionaries

Sprechmann and Sapiro (2010) employed a third method of classification, which is also biologically-plausible. Specifically, they performed sparse coding independently with multiple class-specific sub-dictionaries, and assigned a signal to a class based on the quality of the sparse representation produced by each sub-dictionary (see section 2.4). The third column of Fig. 3 shows results for this classification method, when the quality of the sparse representation was determined by the reconstruction error associated with each sub-dictionary. Algorithms `LARS+` and `LARS-lasso+` again crashed when calculating sparse representations for the MNIST training data. By comparing the third column of Fig. 3 with the

preceding columns, it can be seen that the classification performance of some sparse solvers is significantly different when sparse representations are determined for a single dictionary compared to when sparse representation are found separately for multiple sub-dictionaries. Other algorithms, including PC/BC-DIM, maintain good classification performance for each method of classification. It can also be seen that the classification performance of the proposed algorithm using this method is competitive with that of the best of the alternative algorithms tested.

3.4 Classification based on reconstruction error with multiple learnt sub-dictionaries

For the experiments described above the class-specific sub-dictionaries were predefined using exemplars taken from the training data. These experiments were repeated using smaller (100-element) sub-dictionaries learnt from the training data using the ILS-DLA learning algorithm (see section 2.5). Each sparse solver was used to learn class-specific sub-dictionaries and then its classification performance was tested using those dictionaries it had learnt. Apart from using learnt dictionaries, the procedure was otherwise identical to that used in section 3.3. The results are shown in the last column of Fig. 3. Algorithm LARS-lasso+ crashed when calculating sparse representations during training on the USPS dataset, while FISTA became stuck in a seemingly infinite loop during training on all three datasets. Comparing the third and fourth column of Fig. 3 it can be seen that, with the exception of algorithms OMP and PDBBP, classification performance deteriorates. This is likely to be primarily due to the learnt dictionaries being much smaller than the predefined dictionaries. With the exception of algorithm OMP, the algorithms that perform best in this experiment are the same as those that performed best when tested with predefined dictionaries. This would suggest that the previous results generalise to other methods of dictionary definition. It can also be seen, for this experiment as for the previous experiments, that the classification performance of the proposed algorithm, PC/BC-DIM, is competitive with that of the best of the alternative algorithms tested.

4 Discussion

Each of the tested algorithms has the same objective: to find a sparse set of basis vectors that can reconstruct the input signal with minimum error. It is therefore surprising that the results are so varied. Since all other details of the evaluation process (such as the dictionary, and the method used for assigning signals to classes) are identical, these large differences in performance are due to differences in the sparse representations found by the sparse solvers themselves. This result suggests that performance of sparse coding algorithms may be heavily dependent on the choice of sparse solver. Which sparse solver is optimal may prove to be application specific. However, if the results presented here for three classification tasks generalise to other applications then it suggests that a good choice of algorithm would be PFP+, DALM, or the proposed algorithm: PC/BC-DIM.

This result demonstrates that classification using sparse representations can be performed in a biologically-plausible way. Two biologically-plausible methods were considered. In the first method, a single neural network with synaptic weights representing patterns from all classes was used. A sparse representation was calculated using the PC/BC-DIM algorithm for each input signal, and the class of the signal was determined by summing the responses of the neurons encoding the coefficients of the sparse representation. In the second method, separate neural networks were used for each category, and a signal was classified by summing the responses of error-detecting neurons once the PC/BC-DIM algorithm had found sparse representations with each sub-network. Previous work on sparse coding in the brain has concentrated on simulating early sensory cortical areas, such as V1 and V2 (*e.g.*, Lee et al., 2008; Olshausen and Field, 1997, 2004; Spratling, 2012c; Zhu and Rozell, 2013). The current work suggests that the same mechanisms might also operate in other, higher, cortical regions to perform more cognitive tasks like classification.

Conflict of interest

The authors declare that they have no conflict of interest.

References

Aharon, M., Elad, M., and Bruckstein, A. (2006). K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing*, 54(11):4311–22.

Bell, A. J. and Sejnowski, T. J. (1997). The 'independent components' of natural scenes are edge filters. *Vision Research*, 37(23):3327–38.

Berkes, P., Turner, R. E., and Sahani, M. (2009). A structured model of video rteproduces primary visual cortical organisation. *PLoS Computational Biology*, 5(9):e1000495.

Bociu, I. and Pitas, I. (2004). A new sparse image representation algorithm applied to facial expression recognition. In *Proceedings of the IEEE Signal Processing Society Workshop on Machine Learning for Signal Processing*, pages 539–48.

Charles, A. S., Garrigues, P., and Rozell, C. J. (2012). A common network architecture efficiently implements a variety of sparsity-based inference problems. *Neural Computation*, 24(12):3317–3339.

Damnjanovic, I., Davies, M. E. P., and Plumbley, M. D. (2010). Smallbox - an evaluation framework for sparse representations and dictionary learning algorithms. *Signal Processing*, page 418425.

De Meyer, K. and Spratling, M. W. (2011). Multiplicative gain modulation arises through unsupervised learning in a predictive coding model of cortical function. *Neural Computation*, 23(6):1536–67.

De Meyer, K. and Spratling, M. W. (2013). A model of partial reference frame transforms through pooling of gain-modulated responses. *Cerebral Cortex*, 23(5):1230–9.

Elad, M. and Aharon, M. (2006). Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image Processing*, 15(12):3736–45.

Elhamifar, E. and Vidal, R. (2011). Robust classification using structured sparse representation. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

Engan, K., Skretting, K., and Husøy, H. (2007). Family of iterative LS-based dictionary learning algorithms, ILS-DLA, for sparse signal representation. *Digital Signal Processing*, 17:32–49.

Falconbridge, M. S., Stamps, R. L., and Badcock, D. R. (2006). A simple Hebbian/anti-Hebbian network learns the sparse, independent components of natural images. *Neural Computation*, 18(2):415–29.

Fischer, S., Cristóbal, G., and Redondo, R. (2006). Sparse overcomplete gabor wavelet representation based on local competition. *IEEE Transactions on Image Processing*, 15(2):265–72.

Fischer, S., Redondo, R., Perrinet, L., and Cristóbal, G. (2007). Sparse approximation of images inspired from the functional architecture of the primary visual areas. *EURASIP Journal on Advances in Signal Processing*, 2007:90727.

Földiák, P. (1990). Forming sparse representations by local anti-Hebbian learning. *Biological Cybernetics*, 64:165–70.

Hamker, F. H. and Wiltschut, J. (2007). Hebbian learning in a model with dynamic rate-coded neurons: an alternative to the generative model approach for learning receptive fields from natural scenes. *Network: Computation in Neural Systems*, 18:249–66.

Harpur, G. F. (1997). *Low Entropy Coding with Unsupervised Neural Networks*. PhD thesis, Department of Engineering, University of Cambridge.

Hoyer, P. O. (2003). Modeling receptive fields with non-negative sparse coding. *Neurocomputing*, 52–54:547–52.

Hoyer, P. O. (2004). Non-negative matrix factorization with sparseness constraints. *Journal of Machine Learning Research*, 5:1457–69.

Hoyer, P. O. and Hyvärinen, A. (2000). Independent component analysis applied to feature extraction from colour and stereo images. *Network: Computation in Neural Systems*, 11(3):191–210.

Hyvarinen, A., Hoyer, P., and Oja, E. (1998). Sparse code shrinkage for image denoising. In *Proceedings of the International Joint Conference on Neural Networks*, volume 2, pages 859–64.

Jehee, J. F. M. and Ballard, D. H. (2009). Predictive feedback can account for biphasic responses in the lateral geniculate nucleus. *PLoS Computational Biology*, 5(5):e1000373.

Jiang, Z., Lin, Z., and Davis, L. S. (2011). Learning a discriminative dictionary for sparse coding via label consistent K-SVD. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

Jiang, Z., Lin, Z., and Davis, L. S. (2013). Label consistent K-SVD: Learning a discriminative dictionary for recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(11).

Kang, L. W., Hsu, C. Y., Chen, H. W., Lu, C. S., Lin, C. Y., and Pei, S. C. (2011). Feature-based sparse representation for image similarity assessment. *IEEE Trans. on Multimedia*, 13(5):1019–30.

King, P. D., Zylberberg, J., and DeWeese, M. R. (2013). Inhibitory interneurons decorrelate excitatory cells to drive sparse code formation in a spiking model of V1. *The Journal of Neuroscience*, 33(13):5475–85.

Lee, H., Ekanadham, C., and Ng, A. Y. (2008). Sparse deep belief net model for visual area V2. In Platt, J. C., Koller, D., Singer, Y., and Roweis, S., editors, *Advances in Neural Information Processing Systems*, volume 20, pages 873–80, Cambridge, MA. MIT Press.

Lemme, A., Reinhart, R. F., and Steil, J. J. (2010). Efficient online learning of a non-negative sparse autoencoder. In *Proceedings of the European Symposium on Artificial Neural Networks*.

Liu, J. and Jia, Y. (2012). A lateral inhibitory spiking neural network for sparse representation in visual cortex. In *Advances in Brain Inspired Cognitive Systems*, pages 259–67. Springer.

Lücke, J. (2009). Receptive field self-organization in a model of the fine structure in V1 cortical columns. *Neural Computation*, 21(10):2805–45.

Mairal, J., Bach, F., and Ponce, J. (2012). Task-driven dictionary learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(4).

Mairal, J., Bach, F., Ponce, J., and Sapiro, G. (2010). Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research*, 11:19–60.

Mairal, J., Bach, F., Ponce, J., Sapiro, G., and Zisserman, A. (2008a). Discriminative learned dictionaries for local image analysis. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1–8.

Mairal, J., Bach, F., Ponce, J., Sapiro, G., and Zisserman, A. (2008b). Supervised dictionary learning. In *Advances in Neural Information Processing Systems*.

Mairal, J., Bach, F., Ponce, J., Sapiro, G., and Zisserman, A. (2009). Non-local sparse models for image restoration. In *Proceedings of the International Conference on Computer Vision*, pages 2272–9.

Mairal, J., Sapiro, G., and Elad, M. (2007). Multiscale sparse image representation with learned dictionaries. In *IEEE International Conference on Image Processing (ICIP)*.

Murray, J. F. and Kreutz-Delgado, K. (2006). Learning sparse overcomplete codes for images. *Journal of VLSI Signal Processing*, 45:97–110.

Olshausen, B. A. and Field, D. J. (1996). Emergence of simple-cell receptive properties by learning sparse code for natural images. *Nature*, 381:607–9.

Olshausen, B. A. and Field, D. J. (1997). Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Research*, 37(23):3311–25.

Olshausen, B. A. and Field, D. J. (2004). Sparse coding of sensory inputs. *Current Opinion in Neurobiology*, 14:481–7.

Pece, A. E. C. (2002). The problem of sparse image coding. *Journal of Mathematical Imaging and Vision*, 17(2):89–108.

Pece, A. E. C. and Petkov, N. (2000). Fast atomic decomposition by the inhibition method. In *Proceedings of the International Conference on Pattern Recognition*, pages 215–8.

Perrinet, L. U. (2010). Role of homeostasis in learning sparse representations. *Neural Computation*, 22(7):1812–36.

Plumbley, M. D. (2007). Dictionary learning for L1-exact sparse coding. In *Independent Component Analysis and Signal Separation*, pages 406–13. Springer.

Ramirez, I., Sprechmann, P., and Sapiro, G. (2010). Classification and clustering via dictionary learn-

ing with structured incoherence and shared features. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3501–8.

Rao, R. P. N. and Ballard, D. H. (1999). Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature Neuroscience*, 2(1):79–87.

Rehn, M. and Sommer, F. T. (2007). A network that uses few active neurons to code visual input predicts the diverse shapes of cortical receptive fields. *Journal of Computational Neuroscience*, 22:135–46.

Rozell, C. J., Johnson, D., Baraniuk, R., and Olshausen, B. A. (2008). Sparse coding via thresholding and local competition in neural circuits. *Neural Computation*, 20(10):2526–63.

Spratling, M. W. (2006). Learning image components for object recognition. *Journal of Machine Learning Research*, 7:793–815.

Spratling, M. W. (2008). Predictive coding as a model of biased competition in visual selective attention. *Vision Research*, 48(12):1391–408.

Spratling, M. W. (2010). Predictive coding as a model of response properties in cortical area V1. *The Journal of Neuroscience*, 30(9):3531–43.

Spratling, M. W. (2011). A single functional model accounts for the distinct properties of suppression in cortical area V1. *Vision Research*, 51(6):563–76.

Spratling, M. W. (2012a). Predictive coding accounts for V1 response properties recorded using reverse correlation. *Biological Cybernetics*, 106(1):37–49.

Spratling, M. W. (2012b). Predictive coding as a model of the V1 saliency map hypothesis. *Neural Networks*, 26:7–28.

Spratling, M. W. (2012c). Unsupervised learning of generative and discriminative weights encoding elementary image components in a predictive coding model of cortical function. *Neural Computation*, 24(1):60–103.

Spratling, M. W., De Meyer, K., and Kompass, R. (2009). Unsupervised learning of overlapping image components using divisive input modulation. *Computational Intelligence and Neuroscience*, 2009(381457):1–19.

Spratling, M. W. and Johnson, M. H. (2004). Neural coding strategies and mechanisms of competition. *Cognitive Systems Research*, 5(2):93–117.

Sprechmann, P. and Sapiro, G. (2010). Dictionary learning and sparse coding for unsupervised clustering. In *IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)*, pages 2042–5.

Thiagarajan, J. J. and Spanias, A. (2011). Learning dictionaries for local sparse coding in image classification. In *Asilomar Conference on Signals, Systems, and Computers*.

Tošić, I. and Frossard, P. (2011). Dictionary learning. *IEEE Signal Processing Magazine*, 28(2):27–38.

Tropp, J. A. and Wright, S. J. (2010). Computational methods for sparse solution of linear inverse problems. *Proceedings of the IEEE*, 98(6):948–58.

Van Hateren, J. H. and van der Schaaf, A. (1998). Independent component filters of natural images compared with simple cells in primary visual cortex. *Proceedings of the Royal Society of London. Series B, Biological Sciences*, 265:359–66.

Weber, C. and Triesch, J. (2008). A sparse generative model of V1 simple cells with intrinsic plasticity. *Neural Computation*, 20:1261–84.

Wei, C.-P., Chao, Y.-W., Yeh, Y.-R., and Wang, Y.-C. F. (2013). Locally-sensitive dictionary learning for sparse representation based classification. *Pattern Recognition*, 46:1277–87.

Wiltschut, J. and Hamker, F. H. (2009). Efficient coding correlates with spatial frequency tuning in a model of V1 receptive field organization. *Visual Neuroscience*, 26:21–34.

Wright, J., Ma, Y., Mairal, J., Sapiro, G., Huang, T., and Yan, S. (2009a). Sparse representation for computer vision and pattern recognition. *Proceedings of the IEEE*, 98(6):1031–44.

Wright, J., Yang, A., Ganesh, A., Sastry, S., and Ma, Y. (2009b). Robust face recognition via sparse representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(2):210–27.

Yang, A., Ganesh, A., Zhou, Z., Sastry, S., and Ma, Y. (2010). A review of fast ℓ_1-minimization algorithms for robust face recognition. *arXiv*, 1007.3753.

Yang, M. and Zhang, L. (2010). Gabor feature based sparse representations for face recognition with gabor occlusion dictionary. In *Proceedings of the European Conference on Computer Vision*.

Yang, M., Zhang, L., Feng, X., and Zhang, D. (2011). Fisher discrimination dictionary learning for sparse representation. In *Proceedings of the International Conference on Computer Vision*, pages 543–50.

Zhang, H., Zhang, Y., and Huang, T. S. (2013). Simultansous discriminative projection and dictionary learning for sparse represntation based classification. *Pattern Recognition*, 46:346–54.

Zhang, L., Yang, M., and Feng, X. (2011). Sparse representation or collaborative representation: Which helps face recognition? In *Proceedings of the International Conference on Computer Vision*, pages 471–8.

Zhu, M. and Rozell, C. J. (2013). Visual nonclassical receptive field effects emerge from sparse coding in a dynamical system. *PLoS Computational Biology*, 9(8):e1003191.

Zylberberg, J., Murphy, J. T., and DeWeese, M. R. (2011). A sparse coding model with synaptically local plasticity and spiking neurons can account for the diverse shapes of V1 simple cell receptive fields. *PLoS Computational Biology*, 7(10):e1002250.