Learning Data Representations with Sparse Coding Neural Gas

Kai Labusch and Erhardt Barth and Thomas Martinetz

University of Lübeck - Institute for Neuro- and Bioinformatics
Ratzeburger Alle 160 23538 Lübeck - Germany

Abstract. We consider the problem of learning an unknown (overcomplete) basis from an unknown sparse linear combination. Introducing the "sparse coding neural gas" algorithm, we show how to employ a combination of the original neural gas algorithm and Oja's rule in order to learn a simple sparse code that represents each training sample by a multiple of one basis vector. We generalise this algorithm using orthogonal matching pursuit in order to learn a sparse code where each training sample is represented by a linear combination of k basis elements. We show that this method can be used to learn artificial sparse overcomplete codes.

1 Introduction

In the last years there has been a lot of interest in sparse coding. Sparse coding is closely connected to independent component analysis, in particular to overcomplete and noisy ICA [1]. Furthermore, favorable properties of sparse codes with respect to noise resistance have been studied [2]. Mathematically the problem is to estimate an overcomplete basis of given training samples $X = (x_1, \ldots, x_L)$, $x_i \in \Re^N$ that were generated from a sparse linear combination. Without loss of generality we require X to have zero mean. We measure the quality of the basis by the mean square of the representation error:

$$E = \frac{1}{L} \sum_{i=1}^{L} \|x_i - Ca(i)\|_2^2 \tag{1}$$

$C = (c_1, \ldots, c_M)$, $c_j \in \Re^N$ denotes a matrix containing the basis elements. $a(i)$ denotes a set of sparse coefficients that was chosen optimally for given x_i and C. The number of basis elements M is a free model parameter. In case of overcomplete bases $M > N$ holds. Imposing different constraints on the basis C or the choice of the coefficients $a(i)$ allows to control the structure of the learned basis.

2 Vector quantization

A well-known approach for data representation is vector quantization. Vector quantization is based on a set of so-called codebook vectors. Each sample is encoded by the closest codebook vector. Therefore, for the coefficients $a(i)$ holds

$$a(i)_k = 1,\ a(i)_j = 0\ \forall j \neq k \quad \text{where} \quad k = \arg\min_j \|c_j - x_i\|_2^2 \tag{2}$$

Vector quantization finds a set of codebook vectors that minimize (1) under the constraints posed by (2). The well-known k-means algorithm is among the methods that try to solve this optimization problem. But k-means can lead to a sub-optimal utilization of the codebook vectors with respect to (1) due to the hard-competitive nature of its learning scheme. The neural gas algorithm introduced in [3] remedies this deficiency by using a soft-competitive learning scheme that facilitates an optimal distribution of the codebook vectors over the data manifold to be learned.

3 Learning one-dimensional representations

In a step towards a more flexible coding scheme, i.e., a coding scheme that in some cases may better resemble the structure of the data, we drop one constraint on the coefficients $a(i)$ allowing a representation in terms of an arbitrary multiple of one codebook vector. Due to the added flexibility of the coefficients, we require $\|c_j\|_2^2 = 1$ without loss of generality. This leads to the following optimization problem, which can be understood as a model of maximum sparseness:

$$\min \sum_{i}^{L} \|x_i - Ca(i)\|_2^2 \quad \text{subject to} \quad \|a(i)\|_0 \leq 1 \quad \text{and} \quad \|c_j\|_2^2 = 1 \tag{3}$$

Here $\|a(i)\|_0$ denotes the number of non-zero coefficients in $a(i)$. First consider the marginal case of (3), where only one codebook vector is available, i.e, $M = 1$. Now (3) becomes:

$$\min \sum_{i=1}^{L} \|x_i - ca(i)\|_2^2 = \sum_{i=1}^{L} x_i^T x_i - 2a(i)c^T x_i + a(i)^2 \text{ subject to } \|c\|_2^2 = 1 \tag{4}$$

Fixing x_i and c, (4) becomes minimal by choosing $a(i) = c^T x_i$. One obtains as final optimization problem:

$$\max \sum_{i=1}^{L} (c^T x_i)^2 \quad \text{subject to} \quad \|c\|_2^2 = 1 \tag{5}$$

Hence, in this marginal case, the problem of finding the codebook vector that minimizes (4) boils down to finding the direction of maximum variance. A well-known learning rule that solves (5), i.e., that finds the direction of maximum variance, is called Oja's rule [4]. Now we describe how to modify the original neural gas algorithm (see Algorithm 1) to solve the general case of (3), where $M > 1$ holds. The soft-competitive learning is achieved by controlling the update of all codebook vectors by the relative distances between the codebook vectors and the winning codebook vector. These distances are computed within the sample space (see Algorithm 1, step 4,5). Replacing the distance measure, we now consider the following sequence of distances:

$$-\left(c_{j_0}^T x\right)^2 \leq \cdots \leq -\left(c_{j_k}^T x\right)^2 \leq \cdots \leq -\left(c_{j_M}^T x\right)^2 \tag{6}$$

Algorithm 1 The neural gas algorithm

initialize $C = (c_1, \ldots, c_M)$ using uniform random values
for $t = 0$ to t_{max} **do**
  select random sample x out of X
  calculate current size of neighbourhood and learning rate:

$$\lambda_t = \lambda_0 \left(\lambda_{final}/\lambda_0\right)^{t/t_{max}}$$
$$\epsilon_t = \epsilon_0 \left(\epsilon_{final}/\epsilon_0\right)^{t/t_{max}}$$

  determine the sequence $j_0, \ldots, j_M$ with:

$$\|x - c_{j_0}\| \leq \cdots \leq \|x - c_{j_k}\| \leq \cdots \leq \|x - c_{j_M}\|$$

  for $k = 1$ to M **do**
    update c_{j_k} according to $c_{j_k} = c_{j_k} + \epsilon_t e^{-k/\lambda_t} \left(c_{j_k} - x\right)$
  end for
end for

Due to the modified distance measure a new update rule is required to minimize the distances between the codebook vectors and the current training sample x. According to Oja's rule, with $y = c_{j_k}^T x$, we obtain:

$$c_{j_k} = c_{j_k} + \epsilon_t e^{-k/\lambda_t} y \left(x - y c_{j_k}\right) \tag{7}$$

Due to the optimization constraint $\|c_j\| = 1$, we normalize the codebook vectors in each learning step. The complete "sparse coding neural gas" algorithm is shown in Algorithm 2.

4 Learning sparse codes with k-coefficients

In order to generalise the "Sparse Coding Neural Gas" algorithm, i.e., allowing for a representation using a linear combination of k elements of C to represent a given sample x_i, we consider the following optimization problem:

$$\min \sum_{i}^{L} \|x_i - Ca(i)\|_2^2 \text{ subject to } \|a(i)\|_0 \leq k \text{ and } \|c_j\|_2^2 = 1 \tag{8}$$

A number of approximation methods tackling the problem of finding optimal coefficients $a(i)$ constrained by $\|a(i)\|_0 \leq k$ given fixed C and x_i were proposed. It can be shown that in well-behaved cases methods such as matching pursuit or orthogonal matching pursuit [5] provide an acceptable approximation [6, 2].

We generalize the "sparse coding neural gas" algorithm with respect to (8) by performing in each iteration of the algorithm k steps of orthogonal matching pursuit. Given a sample x that was chosen randomly out of X, we initialize $U = \emptyset$, $x^{res} = x$ and $R = (r_1, \ldots, r_M) = C = (c_1, \ldots, c_M)$. U denotes the set of indices of those codebook vectors that were already used to encode x. x^{res} denotes the residual of x to be encoded in the subsequent encoding steps, i.e., x^{res} is orthogonal to the space spanned by the codebook vectors indexed

Algorithm 2 The sparse coding neural gas algorithm.

initialize $C = (c_1, \ldots, c_M)$ using uniform random values
for $t = 0$ to t_{max} **do**
  select random sample x out of X
  set $c_1, \ldots, c_M$ to unit length
  calculate current size of neighbourhood and learning rate:

$$\begin{aligned} \lambda_t &= \lambda_0 \left(\lambda_{final}/\lambda_0\right)^{t/t_{max}} \\ \epsilon_t &= \epsilon_0 \left(\epsilon_{final}/\epsilon_0\right)^{t/t_{max}} \end{aligned}$$

  determine $j_0, \ldots, j_M$ with: $-(c_{j_0}^T x)^2 \leq \cdots \leq -(c_{j_k}^T x)^2 \leq \cdots \leq -(c_{j_M}^T x)^2$
  for $k = 1$ to M **do**
    with $y = c_{j_k}^T x$ update c_{j_k} according to $c_{j_k} = c_{j_k} + \epsilon_t e^{-k/\lambda_t} y(x - y c_{j_k})$
  end for
end for

Algorithm 3 The generalised sparse coding neural gas algorithm.

initialize $C = (c_1, \ldots, c_M)$ using uniform random values
for $t = 0$ to t_{max} **do**
  select random sample x out of X
  set $c_1, \ldots, c_M$ to unit length
  calculate current size of neighbourhood: $\lambda_t = \lambda_0 \left(\lambda_{final}/\lambda_0\right)^{t/t_{max}}$
  calculate current learning rate: $\epsilon_t = \epsilon_0 \left(\epsilon_{final}/\epsilon_0\right)^{t/t_{max}}$
  set $U = \emptyset$, $x^{res} = x$ and $R = (r_1, \ldots, r_M) = C = (c_1, \ldots, c_M)$
  for $h = 0$ to $K - 1$ **do**
    determine $j_0, \ldots, j_k, \ldots, j_{M-h}$ with $j_k \notin U$:

$$-(r_{j_0}^T x^{res})^2 \leq \cdots \leq -(r_{j_k}^T x^{res})^2 \leq \cdots \leq -(r_{j_{M-h}}^T x^{res})^2$$

    for $k = 1$ to $M - h$ **do**
a   with $y = r_{j_k}^T x^{res}$ update $c_{j_k} = c_{j_k} + \Delta_{j_k}$ and $r_{j_k} = r_{j_k} + \Delta_{j_k}$ with

$$\Delta_{j_k} = \epsilon_t e^{-k/\lambda_t} y(x^{res} - y r_{j_k})$$

      set r_{j_k} to unit length
    end for
b   determine $j_{win} = \arg \max_{j \notin U} (r_j^T x^{res})^2$
c   remove projection to $r_{j_{win}}$ from x^{res} and R:

$$\begin{aligned} x^{res} &= x^{res} - (r_{j_{win}}^T x^{res}) r_{j_{win}} \\ r_j &= r_j - (r_{j_{win}}^T r_j) r_{j_{win}} \quad , j = 1, \ldots, M \wedge\ j \notin U \end{aligned}$$

d   set $U = U \cup j_{win}$
  end for
end for

by U. R denotes the residual of the current codebook C with respect to the codebook vectors indexed by U. Each of the k encoding steps now adds the index of the codebook vector that was used in the current encoding step to U. The subsequent encoding steps do not consider and update those codebook vectors whose indices are already elements of U. An encoding step involves: (a) updating C and R according to (7), (b) finding the element $r_{j_{win}}$ of R that has maximum overlap with respect to x^{res}, (c) subtracting the projection of x^{res} to $r_{j_{win}}$ from x^{res}, subtracting the projection of R to $r_{j_{win}}$ from R, and (d) adding j_{win} to U. The entire generalised "sparse coding neural gas" method is shown in Algorithm 3.

5 Experiments

We test the "sparse coding neural gas" algorithm on artificially generated sparse linear combinations. We do not consider the task of determining M, i.e., the size of the basis that was used to generate the samples; instead we assume M to be known. The basis vectors and coefficients used to generate training samples are chosen from an uniform distribution. The mean variance of the training samples is set to 1. A certain amount of uniformly distributed noise is added to the training samples. First, we consider a two-dimensional toy example, where each training sample is a multiple of one in five basis vectors, i.e., $M = 5, k = 1, N = 2$. The variance of the additive noise is set to 0.01. Figure 1 (a) shows the training samples, the original basis C^{orig} (dashed line) and the basis C^{learn} that was learned from the data (solid line). Note that the original basis is obtained except for the sign of the basis vectors. In a second experiment, a basis $C^{orig} \in \Re^{20\times 50}$ is generated, consisting of $M = 50$ basis vectors within 20 dimensions. Linear combinations $x_1, \ldots, x_{5000}$ of k basis vectors are computed using uniform distributed coefficients. The learned basis C^{learn} is compared to the original basis C^{orig} that was used to generate the samples. This is done by taking the maximum overlap of each original basis vector c_j^{orig} and the learned basis vectors, i.e., $\max_i |c_i^{learn} c_j^{orig}|$. We repeated the experiment 10 times. Figure 1 (b) shows the mean maximum overlap for $k = 1, \ldots, 15$ for different noise levels. To assess how many of the learned basis vectors can be assigned unambiguously to the original basis, we consider $unique(\hat{C}^{learn})$, which is the size of the set $\hat{C}^{learn} = \{c_k^{learn} : k = \arg\max_i |c_i^{learn} c_j^{orig}|,\ j = 1, \ldots, M\}$ without repetitions. Figure 1 (c) shows the mean of $unique(\hat{C}^{learn})$. Increasing noise level leads to decreasing performance as expected. The less sparse the coefficients are (the larger k) the lower the quality of the dictionary reconstruction is (see also [6, 2]). Finally, we repeat the second experiment by fixing $k = 7$ and evaluating the reconstruction error (1) during the learning process. Note that the coefficients used for reconstruction are determined by orthogonal matching pursuit with k steps. Figure 1 (d) shows that the reconstruction error decreases over time.

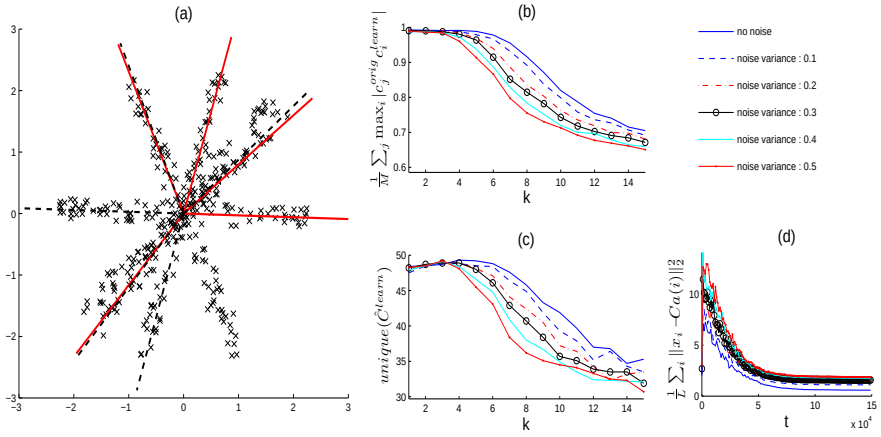

Fig. 1: (a) two-dimensional toy example where each sample is a multiple of one in five basis vectors plus additive noise (b) mean maximum overlap between original and learned basis (c) mean size of $\hat{C}^{learn}$ without repetitions (d) mean reconstruction error; Sparse Coding Neural Gas parameters used: $\lambda_0 = M/2, \lambda_{final} = 0.01, \epsilon_0 = 0.1, \epsilon_{final} = 0.0001, t_{max} = 20 * 5000$.

6 Conclusion

We described a new method that learns an overcomplete basis from unknown sparse linear combinations. In experiments we have shown how to use this method to learn a given artificial overcomplete basis from artificial linear combinations with additive noise present. Our experiments show that the obtained performance depends on the sparsity of the coefficients and on the strength of the additive noise.

References

[1] Aapo Hyvarinen, Juha Karhunen, and Erkki Oja. *Independent Component Analysis.* Wiley-Interscience, May 2001.

[2] David L. Donoho, Michael Elad, and Vladimir N. Temlyakov. Stable recovery of sparse overcomplete representations in the presence of noise. *IEEE Transactions on Information Theory*, 52(1):6–18, 2006.

[3] T. Martinetz and K. Schulten. A "Neural-Gas Network" Learns Topologies. *Artificial Neural Networks*, I:397–402, 1991.

[4] E. Oja. A simplified neuron model as a principal component analyzer. *J. Math. Biol.*, 15:267–273, 1982.

[5] Y. Pati, R. Rezaiifar, and P. Krishnaprasad. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. *Proceedings of the 27 th Annual Asilomar Conference on Signals, Systems,*, November 1993.

[6] J. A. Tropp. Greed is good: algorithmic results for sparse approximation. *IEEE Transactions on Information Theory*, 50(10):2231–2242, 2004.