Quadratic Programming for Learning Sparse Codes

Article · December 2003

DOI: 10.1049/cp:19991174 · Source: CiteSeer

2 authors:

Dominik Endres
Philipps University of Marburg
69 PUBLICATIONS **692** CITATIONS

SEE PROFILE

Peter Földiák
43 PUBLICATIONS **2,058** CITATIONS

SEE PROFILE

Quadratic programming for learning sparse codes

D. Endres and P. Földiák
dme2@st-andrews.ac.uk, Peter.Foldiak@st-andrews.ac.uk
School of Psychology, University of St Andrews,
St Andrews KY16 9JU, U.K.

October 30, 2003

1 Introduction

Building on a framework by Daugmann [2] and Pece [9], Harpur and Prager [4, 5, 6] constructed a neural network, which they termed the REC model, capable of discovering sparsely distributed representations by using the principle of redundancy reduction. Olshausen and Field [7, 8] employed a similar technique for efficient coding of natural images and showed how the resulting response functions of the units relate to the properties of simple cells in the mammalian primary visual cortex.

In order to model the function of later stages of visual processing in mammals, the activation patterns of this network could be used as an input to another one of similar architecture. It would therefore be advantageous if these patterns could be calculated fast. Moreover, not only speed but also accuracy would be an important issue if the network was to be used in practical applications, such as image compression. We have derived an algorithm that achieves both goals with far less computational effort than gradient descent based minimizers.

2 The model

The theoretical justification for the REC model is given in [6]. Here we focus on the dynamics of the learning process. The aim is to minimise the function

$$E^\mu = \frac{1}{2}\left(\vec{X}^\mu - \sum_{i=1}^{m} a_i^\mu \, \vec{w}_i\right)^2 + \lambda \sum_{i=1}^{m} S(a_i^\mu) + \gamma \sum_{i=1}^{m} \left(\vec{w}_i^2 - 1\right)^2 \qquad (1)$$

where a_i^μ, $i = 1, \ldots, m$ are the activations of the network's units upon presentation with the μ-th example $\vec{X}^\mu$, $\vec{w}_i$ are the basis vectors of dimension n and the value of $\lambda > 0$ controls the importance of representational sparseness versus good reconstruction of the example. This function is similar to the one used in [6], but the hard constraint on the length of the weight vectors is replaced by the additional term on the right hand side which ensures that the $\vec{w}_i$ do not grow unboundedly. This is important if the sparsifier $S(a)$ is to have the desired effect, as described in [6].

The training process, which is based on [6] and [8], is carried out in two nested loops:

- *Inner loop*: For the current example $\vec{X}^\mu$, E^μ the activations which minimise E^μ are being calcuated. Using these activations the gradient of E^μ with respect to the basis vectors, $\nabla_{\vec{w}_j} E^\mu$, is being computed.

- *Outer loop*: The basis vectors are updated by gradient descent on E, the gradient is obtained by averaging over many iterations (e.g. $\mathcal{O}(n)$) of the inner loop:

$$\vec{w}_j \to \vec{w}_j - \eta \langle \nabla_{\vec{w}_j} E^\mu \rangle_\mu \qquad (2)$$

  where η is a learning rate.

The basis vectors can be expected to converge to a stable configuration after $\mathcal{O}(n)$ repetitions of the outer loop, which is typical for gradient descent algorithms [1]. In the following discussion we assume $m = n$. This is not too severe a restriction, as Olshausen and Field showed in [8] that setting the number of units equal to the input dimesion means an effectively 1.5 times

overcomplete representation for natural images. However, a generalization to the case $m > n$ is possible.

3 Optimizing the inner loop

The time-critical step is the minimization in the inner loop which has to be repeated $\mathcal{O}(n^2)$ times in the course of training. We have therefore derived an algorithm that achieves this goal fast and accurately.

In the past, gradient descent and conjugate gradient methods were used to carry out this minimization. As we consider only a linear sparsifier $S(a) = |a|$, that was found to work well in image coding, the function E^μ is quadratic in a_i^μ. Hence, *quadratic programming* techniques [3] can be employed for the minimisation. These algorithms require a continuous first derivative of the function under consideration and converge to a point where the gradient vanishes. As $S(a) = |a|$ is not differentiable at $a = 0$, the minimum will not be found if it is located at a point where some of the $a_i^\mu = 0$. This is quite likely to happen, as the error (eqn. 1) promotes sparse representations. Indeed, numerical simulations indicate that this is the case for about 90 percent of the activations. A gradient descent algorithm will oscillate around the minimum with an amplitude that depends on the chosen stepsize.

However, to identify a minimum, the gradient does not need to vanish, it is sufficient to require that it is negative in the limit $a_i^\mu \to 0^-$ and positive in the limit $a_i^\mu \to 0^+$:

$$\begin{aligned} \lim_{a_i^\mu \to 0^+} \frac{\partial}{\partial a_i^\mu} E^\mu &> 0 \quad \text{and} \\ \lim_{a_i^\mu \to 0^-} \frac{\partial}{\partial a_i^\mu} E^\mu &< 0 \end{aligned} \tag{3}$$

Lets assume it was already known that $\forall i \in I_0 = \{i_1, \ldots, i_K\} : a_i = 0$ and $\forall i \in I_1 = \{i_K + 1, \ldots, i_n\} : a_i \neq 0$, $I_0 \cap I_1 = \emptyset$, $I_0 \cup I_1 = \{1, \ldots, n\}$. The sets I_0 and I_1 contain the indices of all inactive and all active units, respectively. (Here and in the following the example index μ will be omitted for clarity). The minimum with respect to the $n - K$ non-zero activations $\hat{\vec{a}} = \left(a_{i_{K+1}}, \ldots, a_{i_n}\right)^T$ would then be determined by setting the gradient of E with respect to these activations to zero, yielding

$$\mathbf{A}\hat{\vec{a}} = \vec{b} - \lambda\, \vec{s} \tag{4}$$

where $\mathbf{A} = \left(\vec{w}_{i_{K+1}}, \ldots, \vec{w}_{i_n}\right)^T \cdot \left(\vec{w}_{i_{K+1}}, \ldots, \vec{w}_{i_n}\right)$, $\vec{b} = \left(\vec{w}_{i_{K+1}}, \ldots, \vec{w}_{i_n}\right)^T \cdot \vec{I}$ and $\vec{s} = (s_{i_{K+1}} = \text{sign}\left(a_{i_{K+1}}\right), \ldots, s_{i_n} = \text{sign}\left(a_{i_n}\right))^T$. The $\text{sign}(x)$ function is here defined as

$$\text{sign}(x) = \begin{cases} 1 & : \quad x \geq 0 \\ -1 & : \quad x < 0 \end{cases}$$

Of course, it is not *a priori* clear, which a_i will be zero and which will not. Thus, an iterative procedure has to be employed that produces configurations of I_0 and I_1 and terminates, when a configuration is consistent with conditions 3 and 4. A good inital guess is $I_1 = \emptyset$, because, as mentioned above, quite a large fraction of the a_i will be zero at the minimum. Putting all these considerations together leads to the following algorithm:

1. Set $K = n$, $I_0 = \{1, \ldots, n\}$, $I_1 = \emptyset$
2. Determine the minimum $\hat{\vec{a}} = \left(a_{i_{K+1}}, \ldots, a_{i_n}\right)^T$ of E (eqn. 1) subject to the constraints $\forall i \in I_0 : a_i = 0$ and $\forall i \in I_1 : (\text{sign}(a_i) = s_i \vee a_i = 0)$ using a simplified version of the *active set quadratic programming* technique [3].
3. Update $I_0 \to I_0 \cup \{i | a_i = 0 \wedge i \in I_1\}$ and $I_1 \to I_1 \setminus \{i | a_i = 0 \wedge i \in I_1\}$. Update $\hat{\vec{a}}$ and $\vec{s}$ by removing all the components that have become 0.
4. Calculate
$$\begin{aligned} \sigma = \max_{i \in I_0} &\left(\text{sign}\left(z_i^+\right) \cdot \text{sign}\left(z_i^-\right)\right. \\ &\left. \cdot \min(|z_i^+|, |z_i^-|)\right) \end{aligned} \tag{5}$$
where $z_i^+ = lim_{a_i \to 0^+} \frac{\partial E}{\partial a_i}$ and $z_i^- = lim_{a_i \to 0^-} \frac{\partial E}{\partial a_i}$. If $\sigma > 0$, then condition (3) is violated and the corresponding $a_i \neq 0$ at the minimum. Thus, update $I_0 \to I_0 \setminus \{a_i\}$, $I_1 \to I_1 \cup \{a_i\}$, decrement K by one, add a component to $\hat{\vec{a}}$ and $\vec{s} \to (\vec{s}| - \text{sign}\left(z_i^+\right))$ and goto step 2.
5. Return $\hat{\vec{a}}$, I_0, and I_1.
6. End.

In step 2, a simplification to the general quadratic programming algorithm was used as the a_i are decoupled with respect to the constraints.

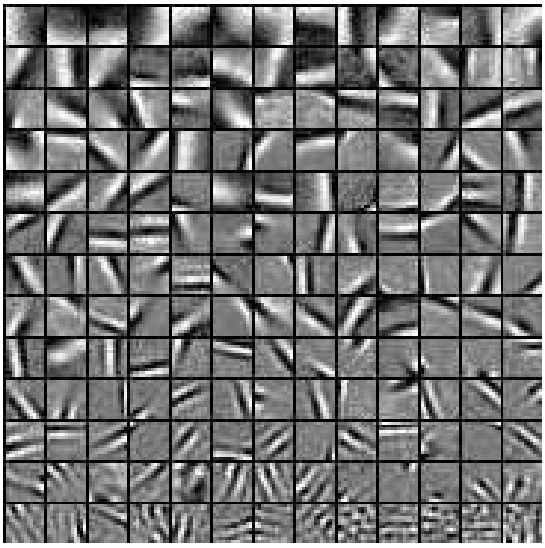

Figure 1: The basis vectors after 300 updates. The vectors are ordered from top left (greatest $\langle|a_i^\mu|\rangle_\mu$) to bottom right (smallest $\langle|a_i^\mu|\rangle_\mu$) and the grayscales are individually normalized.

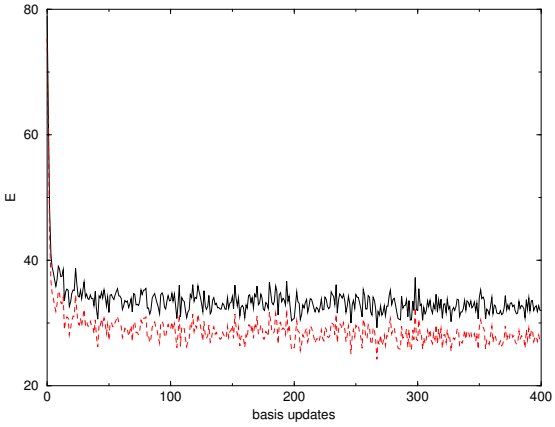

Figure 2: Evolution of the mean error $\langle E^\mu \rangle_\mu$ (averaged over 100 examples) in the course of training. Dashed curve: the proposed algorithm for the inner loop minimization. Solid curve: gradient descent minimization with the same amount of computational effort.

4 Results

Training was performed on 48 natural images (people, animals, landscapes etc.). Each unit had a receptive field of 13×13 pixels, the fields were fully overlapping. The learning rate η for the basis update was initially 0.2 and decreased exponentially to 0.1 within 300 iterations of the outer loop. The inner loop was repeated 100 times between successive basis updates. λ was set to 0.5.

The resulting receptive fields (fig. 1) look qualitatively similar to those obtained previously [6, 7, 8].

The proposed inner loop algorithm finds the minimum of E^μ with a computational effort that is equivalent to roughly 20 gradient descent steps once the basis vectors have converged (it is a bit slower initially, which is due to the fact that the randomly drawn basis vectors at the beginning of the training do not constitute a sparse code). This is not only beneficial to potential practical applications such as image compression, but it also substantially accelerates the learning process, as can be seen in fig. 2. The observed difference in E^μ is largely due to a difference in sparseness: the proposed algorithm produces a code with the same average reconstruction error as gradient descent. However,for a given patch about 88 % of the units are inactive, compared to the gradient descent algorithm where that is the case for only 6 % of the units. Moreover, fig. 2 indicates that to reach the average value of E^μ after 300 basis updates with the gradient descent minimiser takes approximately 20 iterations of the quadratic programming algorithm.

References

[1] Bishop, C. M. (1995): Neural Networks for pattern recognition, Oxford University Press

[2] Daugmann, J.G. (1988). Complete discrete 2-D Gabor transforms by neural networks for image analysis and compression. IEEE Transactions on Acoustics, Speech and Signal Processing 36(7), 1169-1179

[3] Fletcher,R. : Practial methods of optimization, Vol. 2, John Wiley & Sons (1980)

[4] Harpur, G. F. and R. W. Prager (1995). Techniques in low entropy coding with neural networks. Technical Report CUED/F-INFENG/TR 197, Cambridge University Engineering Department

[5] Harpur G. F. and R. W. Prager (1996). Development of low entropy coding in a recurrent network. Network: Computation in Neural Systems 7(2), 277-284.

[6] Harpur, G.F. and R. W. Prager (1999).Experiments with low entropy Neural Networks. in Information Theory and the Brain, eds. R Baddeley, P Hancock, P Földiák, Cambridge University Press, in press.

[7] Olshausen, B. A. and D. J. Field (1996a) Emergence of simple-cell receptive field properties by learning a sparse code for natural images. Nature 381(6583), 607-609

[8] Olshausen, B. A. and D. J. Field (1996b). Natural image statistics and efficient coding. Network: Computation in Neural Systems 7(2), 333-339.

[9] Pece, A. E. C. (1992). Redundancy reduction of a Gabor representation: A possible computational role for feedback from primary visual cortex to lateral geniculate nucleus. In I. Aleksander and J. Taylor (Eds.), Artificial Neural Networks 2, pp. 865-868. Amsterdam: Elsevier.