

Efficient coding leads to novel features for speech recognition

Willie Smit, Etienne Barnard

Department of Electrical,
Electronic and Computer Engineering
University of Pretoria, 0002

ebarnard@up.ac.za

Abstract

The principle of efficient coding of stimuli can explain the receptive fields in the primary visual cortex and in the primary auditory cortex. When efficient coding is applied to a generative model, it forms biologically realistic basis functions and the code it produces has the spike-like property found in the cortex. We show that this representation can be used for isolated word speech recognition.

We have trained a temporal generative model on spoken single digits. The code from the model is spatiotemporal and spike-like, and we used a k-nearest neighbour classifier to classify this code. The network is able to classify 92% of test samples correctly.

1. Introduction

There is good evidence that properties of neurons in the primary visual cortex and in the primary auditory cortex can be understood in terms of efficient coding of natural stimuli [1, 2, 3, 4, 5]. Such an efficient code gives rise to natural features.

There are two approaches to find an efficient code: independent component analysis (ICA) and sparseness. Both approaches yield a similar code as both solve the same problem, but with slightly different assumptions [5].

ICA gives rise to basis functions resembling the receptive fields in the primary visual cortex [1, 2], but it does not produce a spike-like code as is seen in the cortex. Sparse coding on the other hand explains both the visual receptive fields [1, 5] and the spiking activity of neurons in the primary visual cortex [6]. *Sparse coding in time* [6] refers to a code that is sparse over both time and space; with such a code the activity of a channel over time is mostly close to zero, but occasionally it is very high, and the temporal activities of channels are statistically independent.

With a sparse code, a generative model finds the code that will reconstruct the input using a set of basis functions. When the input is temporal, the model needs to consider the temporal nature in order to find a code that is independent over time.

We developed a single word speech recognition model that is based on the efficient coding of stimuli and that

uses a temporal generative model to find the code. The code is spike-like and we interpret the activity as spikes. The spike trains are then classified with a k-nearest neighbour classifier.

2. Methods

2.1. Spectrotemporal processing

Sounds are spectrotemporally processed by the primary auditory cortex. We used a 16 channel spectrotemporal representation for the sounds. We further clipped values in the spectrogram that were below a certain threshold, as these values represent inaudible sounds.

2.2. Generative model

The cortex should implement a generative model in order to explain the sensory message. Most authors use a static generative model, where the current state of the model is independent of its previous states. A stimulus X of length N is encoded by M responses a_i , $i = 1, \dots, M$. The stimulus is a linear sum of M basis functions $\phi_i(t)$ of length N :

$$X(t) = \sum_{i=1}^M a_i \phi_i(t), \quad t = 1, \dots, N \quad (1)$$

A stimulus can be encoded efficiently with sparse coding. It requires that the responses should be independent. A set of natural stimuli is used to evaluate the independence of the responses; this way the probabilistic nature of the stimuli is also considered.

An over-complete presentation, where there are more basis functions than dimensions in the stimulus, has some advantages. It is more robust to noise and it can produce an even sparser code. When an over-complete representation is used, there are infinitely many solutions for the code. This means that other criteria can be used to find the “best” code. Sparseness is a good choice for the reasons mentioned earlier.

To encode a stimulus of length K with basis functions of length N , it is usually assumed that the stimulus is divided into $K - N + 1$ sections of length N , and each

section is encoded independently from the other sections. This leads to a redundant representation because the responses $a_i(t)$ at time t contains information about the stimulus in the window $[t - N, t - 1]$, it is not necessary for the responses at time $t + 1$ to duplicate the stimulus in that window. That is why Lewecky [4] and Klein et al. [3] found that the model in (1) forms similar basis functions at different delays. We use a temporal generative model [6] that eliminates this form of redundancy.

In this model, the past activity of channels also contribute to the current state of the model.

$$X(t) = \sum_i \sum_T a_i(T) \phi_i(T - t) \quad (2)$$

$$\phi_i(t) = \begin{cases} \Re & \text{if } 0 \leq t < \Delta t_\phi - 1 \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

For a multichannel input the basis functions have as many channels as the input. The generative model then becomes:

$$X_j(t) = \sum_i \sum_T a_i(T) \phi_{ij}(T - t) \quad (4)$$

with j the subscript to the input channel number.

A set of basis functions is over-complete when there are more basis functions than channels, in which case we use sparse coding to find the ‘‘best’’ code. The sparseness of a temporal code can be quantified as:

$$\sum_T \sum_i S(a_i(T)) \quad (5)$$

$$S(a) = \ln(1 + (a/\sigma)^2) \quad (6)$$

where σ is a constant (we used $\sigma = 0.1$). The sparseness measure will be small when the code is sparse. Model 4 aims to find a code that will reconstruct the input by using a linear combination of the basis functions. The problem of finding such a code that is also sparse can be expressed as an optimization problem with the following cost function:

$$E = \lambda \sum_T \sum_j [X_j(T) - R_j(T)]^2 + \sum_T \sum_i S(a_i(T)) \quad (7)$$

where $R(T)$ is the reconstructed signal, λ is a constant that adjusts the balance between the reconstruction error and sparseness (we used $\lambda = 33.3$). The generative model allows for the code to have positive and negative values. This property does not agree with a spike-like code as spikes signal events [7]. We can force the code to only take on positive values by means of an inequality constraint $a_i(T) \geq 0$. We solved the optimization problem using the LFOPC optimization algorithm [8] as it is robust, it quickly moves close to a minimum and it does

not get stuck in small local minima. The derivative of the code with respect to the cost function is:

$$\frac{\partial E}{\partial a_i(t)} = -2\lambda \sum_T \sum_j [X_j(T) - R_j(T)] \phi_{ij}(t - T) + S'(a_i(t)) \quad (8)$$

There exists a set of basis functions that will minimize the cost function for a given set of stimuli. This set of basis functions can be found by making Φ part of the design variables of the optimization problem. Each iteration of this optimization problem is then solved in two steps; during the first step the code that minimizes the cost function given the set of basis functions is found. Then the basis functions are adapted to further minimize the cost function given the code. The second step was solved with the golden section method by searching along the direction of steepest descent. The derivative of the basis functions with respect to the cost functions is:

$$\frac{\partial E}{\partial \phi_{ij}(t)} = -2\lambda \sum_T [X_j(T) - R_j(T)] a_i(t + T) \quad (9)$$

The norm of the basis functions will grow without bounds, because the greater their norm, the smaller the values that the code require. The basis functions were rescaled after each iteration to have a norm of 0.1.

2.3. k-Nearest neighbour classifier

Once the code a has been found, it can be used as features for a classifier. We used the maximum value, A_i , of each channel as a feature, and also the time, τ_i , at which the maximum value occurs. The distance measure is the sum of two components, $d = d_A + d_\tau$, one for the maximum values and one for times at which these values occur. The components were calculated as follows:

$$d_A = \| A - A^{train} \| \quad (10)$$

and

$$d_T = \| \tau - \tau^{train} \| \quad (11)$$

The distance d_T should be independent of translations of either pattern. This can be accomplished by translating one pattern so that its temporal mean coincide with that of the other pattern:

$$\bar{D} = \tau - \tau^{train} \quad (12)$$

$$\bar{D}_{coincide} = \bar{D} - \text{mean}(\bar{D}) \quad (13)$$

Usually not all the channels are active for a sample; this means that for the inactive channels the times at which the maximum values occur are undetermined. The entries in $\bar{D}_{coincide}$ that correspond to channels that are inactive for both samples should be set to 0, and a penalty should be added for each such channel. The time component is now:

$$d_T = \| \bar{D}_{coincide} \| + \eta N_{XOR} \quad (14)$$

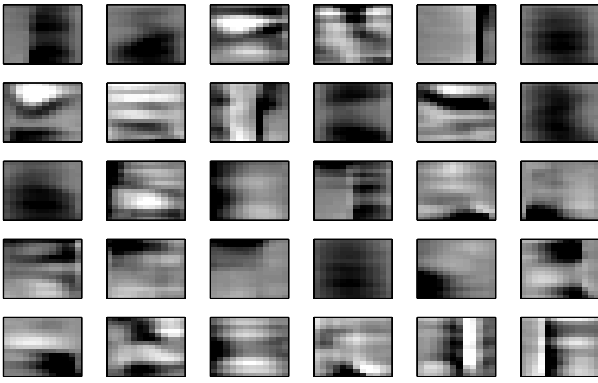


Figure 1: The basis functions after training. The y-axis gives the frequency. The x-axis gives the time. (Basis functions span 250ms).

where the penalty is η (we chose $\eta = 20$), and N_{XOR} is the number of channels that should be penalized.

Finally the distance should be normalized:

$$d = \alpha \frac{d_A}{\sum A + \sum A^{train}} + \frac{d_T}{N_{spikes} + N_{spikes}^{train}} \quad (15)$$

where N_{spikes} and N_{spikes}^{train} are the number of active channels for each pattern, α is a weight (we chose $\alpha = 100$) to set the balance between d_A and d_T .

3. Results

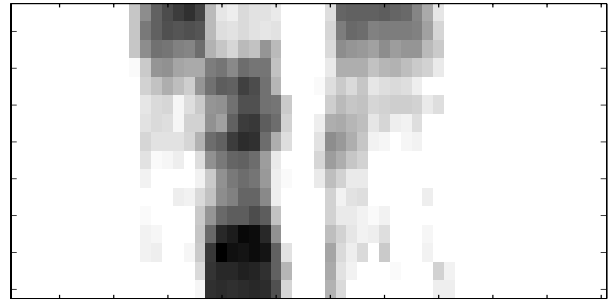
The recognition model needs to be trained in two steps. First the basis functions that will give a sparse code in time must be found and then the classification network has to be trained.

The first step is unsupervised. We used all the single digits (“oh”, “zero”, “one”,... , “nine”) from the TIDIG-ITS training database for training [9]. The basis functions that were formed are given in figure 1.

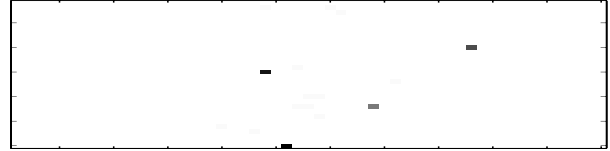
With a distance measure that does not include the d_T term, we found a correct classification rate of 92% of the test samples. With the d_T term included, the classification dropped to 91% of the test samples. The d_T term does not appear to improve performance of isolated word recognition, but it may help with continuous speech recognition, especially when the same word is spoken repeatedly.

4. Conclusion

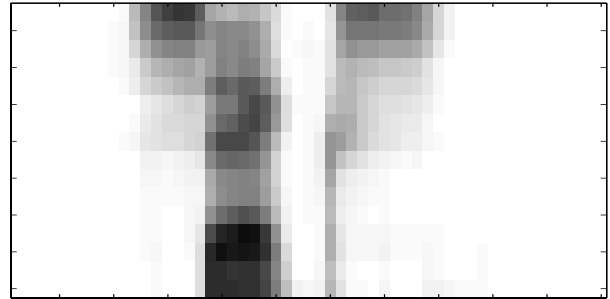
We have presented a simple speech recognition model based on recent research into the efficient coding of stimuli. From a pattern recognition perspective the model learns, without supervision which features to extract. The generative model then uses these features to reduce the dimensions of the input by transforming it into a sparse spatiotemporal code. The code gives the times at which a specific feature is present in the signal.



(a)



(b)



(c)

Figure 2: (a) A spectrogram of the word “six” ($X(t)$). (b) The sparse code ($a(t)$) for the word in figure 2(a) using the basis functions in figure 1. Notice that only four of the basis functions (1, 9, 16, 21) are used to reconstruct the spectrogram. (c) The spectrogram after it has been reconstructed ($R(t)$) from the sparse code.

We have showed that the temporal code can be used to do isolated word recognition. Since the code is temporal, it should be possible to use this code as the input to a continuous speech recognition system. The challenge is to develop a system that will be able to use the temporal code.

Sparse coding can also be applied to recognition problems in vision, or for that matter any other sensory domain. It is a promising approach to recognition problems, and the results researchers have obtained so far indicates that the cortex performs sparse coding.

5. Acknowledgements

The authors would like to thank the National Research Foundation for financial support (Grand number 2053242) and Intel for sponsoring the Ipercube.

6. References

- [1] A. J. Bell and T. J. Sejnowski, "The 'independent components' of natural scenes are edge filters," *Vision Research*, vol. 37, pp. 3327–3338, 1997.
- [2] J. H. V. Hateren and D. L. Ruderman, "Independent component analysis of natural images sequences yield spatiotemporal filters similar to simple cells in primary visual cortex," *Proceedings of the Royal Society of London*, vol. 265, pp. 2315–2320, 1998.
- [3] D. J. Klein, P. König, and K. P. Körding, "Sparse spectrotemporal coding of sounds," *EURASIP Journal on Applied Signal Processing*, vol. 7, pp. 659–667, 2003.
- [4] M. S. Lewicki, "Efficient coding of natural sounds," *Nature Neuroscience*, vol. 5, no. 4, pp. 356–363, 2002.
- [5] B. A. Olshausen and D. J. Field, "Sparse coding with an overcomplete basis set: A strategy employed by V1?" *Vision Research*, vol. 37, pp. 3311–3325, 1997.
- [6] B. A. Olshausen, *Probabilistic Models of the Brain: Perception and Neural Function*. MIT Press, 2002, ch. Sparse Codes and Spikes, pp. 257–272.
- [7] P. M. Hoyer, "Modeling receptive fields with non-negative sparse coding," *Neurocomputing*, vol. 52–54, pp. 547–552, 2003.
- [8] J. A. Snyman, "The LFOPC leap-frog algorithm for constrained optimization," *Computers and Mathematics with Applications*, vol. 40, pp. 1085–1096, 2000.
- [9] L. D. C. (LDC), "NIST CD-ROM version of the Texas Instruments-developed studio quality speaker-independent connected-digits corpus," IDC catalog no.: LDC93S10.