

A Biologically Motivated Solution to the Cocktail Party Problem

Brian Sagi*

Department of Electrical and Computer Engineering, University of California, San Diego, La Jolla, CA 92093-0407, U.S.A.

Syrus C. Nemat-Nasser

Department of Physics, University of California, San Diego, La Jolla, CA 92093-0319, U.S.A.

Rex Kerr

Department of Biology, University of California, San Diego, La Jolla, CA 92093-0349, U.S.A.

Raja Hayek

Christopher Downing

Department of Electrical and Computer Engineering, University of California, San Diego, La Jolla, CA 92093-0407, U.S.A.

Robert Hecht-Nielsen

Department of Electrical and Computer Engineering, Program in Computational Neurobiology, Institute for Neural Computation, University of California, San Diego, La Jolla, CA 92093-0407, U.S.A., and HNC Software, San Diego, CA 92121, U.S.A.

We present a new approach to the cocktail party problem that uses a cortronic artificial neural network architecture (Hecht-Nielsen, 1998) as the front end of a speech processing system. Our approach is novel in three important respects. First, our method assumes and exploits detailed knowledge of the signals we wish to attend to in the cocktail party environment. Second, our goal is to provide preprocessing in advance of a pattern recognition system rather than to separate one or more of the mixed sources explicitly. Third, the neural network model we employ is more biologically feasible than are most other approaches to the cocktail party problem. Although the focus here is on the cocktail party problem, the method presented in this study can be applied to other areas of information processing.

* To whom all correspondence should be directed at bsagi@cerian.com.

1 Introduction

We consider the cocktail party problem as follows. A human listener is in a room in which many people are speaking. The listener's goal is to follow a single speaker (the *attended speaker*) and understand what that speaker is saying. To distinguish the above problem from other formulations, we refer to it as the *human cocktail party problem*.

It is important to note that our goal in the human cocktail party problem defined here is not to separate one or more of the sources from the mixture. We may think of our human cocktail party problem as a problem of attended source identification. This difference is significant because even though it is clear that people can perform the attending function quite well, technical (machine) solutions for the problem prove hard to find. Any biologically motivated solutions to the human cocktail party problem are of additional interest since they may provide insight into information processing that takes place when humans solve the human cocktail party problem. Finally, while we make the analogy to human speech recognition in a real cocktail party environment, the same technical approach could be applied to other important areas such as visual information processing, radar signal processing, and other types of sound processing.

We model our cocktail party problem as an aspect of the human speech recognition problem in a cocktail party environment. The model environment consists of a collection of speakers in a room, all talking simultaneously. A listener samples this auditory scene through a single microphone. This listener is familiar with the language and would like to focus on a particular speaker as if the other speakers did not exist. In this model, all speakers speak the same language and have the same voice qualities, such as pitch variation, vocal chord frequency, and volume. Such a set of identical speakers makes the problem more difficult in some respects, since the system cannot rely on voice quality variations to separate the sources. A dilemma we faced when designing the model problem was how to make the model problem as realistic as possible, without getting bogged down by problem-specific preprocessing and immense computational requirements. To solve this dilemma, we adopted a middle path using real-world speech data with a simple synthetic model of a language, allowing us to solve the model cocktail party problem with modest computational resources.

The cocktail party problem is traditionally treated as a blind source separation problem, with many techniques offered to handle the separation. Some of the most prominent solutions include the information maximization approach of Bell and Sejnowski (1995) and independent component analysis by the minimization of mutual information as examined by Comon (1994), Amari, Cichocki, and Yang (1996), and Oja and Karhunen (1995) among others. Lee, Girolami, Bell, and Sejnowski (2000) and Amari and Cichocki (1998) provide recent reviews of these techniques. Most approaches to blind source separation focus on reproducing all the original sources

as the measure of performance, and some require multiple microphones (n microphones to separate up to n sources).¹ These approaches to the cocktail party problem are blind in the sense that they ignore the properties of the source (other than assuming some general statistical characteristics of the source components). In addition, most methods used in blind source separation require a form of gradient-descent learning or the inversion of large matrices and are therefore computationally intensive and suffer from start-up delays.

Another area of research related to the human cocktail party problem is computational auditory scene analysis. These techniques take advantage of properties of real speech, but currently do not exploit language knowledge. For example, voice quality variations such as pitch and frequency characteristics may be used to separate different speakers. Although some research in this area is modeled after biological systems, the computational techniques employed include nonbiological methods of classification. Wang and Brown (1999) give an overview of computational auditory scene analysis and a specific approach to the separation of speakers with added noise. The literature on auditory scene analysis reports modest results at best for cocktail party environments consisting of more than a few speakers.

The distinction between the cocktail party problem and the single-speaker speech recognition problem is important. Modern speech recognition systems achieve a high level of accuracy in transcribing a single speaker's speech into text. Most of these systems are based on hidden Markov models (HMMs) and modeled after the SPHINX system (Lee, Hon, Hwang, & Huang, 1996). Nevertheless, these systems can operate only in environments exhibiting a high signal-to-noise ratio. In this work, we do not attempt to duplicate the capabilities of such systems.

The cocktail party problem is set in a low signal-to-noise ratio environment. A system solving the cocktail party problem is analogous to a front end to a speech recognition system that enables the system to operate in a high signal-to-noise ratio environment. For example, Yen and Zhao (1997) and Koutras, Dermatas, and Kokkinakis (1999) report independent efforts to use blind source separation techniques in front of an HMM automatic speech recognition system. Both efforts employ two microphones to separate two speakers in a room to produce a reasonable signal-to-noise ratio suitable for speech recognition.

Our approach to the human cocktail party problem uses an artificial neural network called a cortronic network. This network functions as part of the front end of a listener in the model cocktail party environment. The cortronic artificial neural network architecture was proposed by Hecht-Nielsen (1998) and is similar to a Steinbuch Learnmatrix (Steinbuch, 1963) or a Willshaw

¹ In fact, this requirement of n microphones has often been included in the definition of blind separation in the past. For example, see Bell and Sejnowski (1995).

associative network (Willshaw, Buneman, & Longuet-Higgins, 1969) with the addition of some new biologically motivated ideas. We use a hierarchical combination of cortronic regions to process temporal information in a biologically motivated way. We rely on training the network on the language spoken by the individual sources in the model cocktail party environment. Our model listener uses the structural knowledge of the language,² gained through training, together with the incoming speech to form expectations about the content of the attended speaker's speech. For example, familiarity with the language can allow the listener to complete partial phrases. This technical approach differs from other approaches to the cocktail party problem in several important respects. First, we do not depend on an inherent distinction between the sources other than the specific content of their speech. Second, we require only a single microphone regardless of the number of sources in the cocktail. Third, we use specific knowledge of the source. This part of our approach is contrary to the blind aspect of blind source separation but is consistent with the human cocktail party problem. For example, attempts by a human listener to focus on one source in a room of similarly sounding speakers, speaking an unfamiliar foreign language, are unlikely to succeed.

2 Technical Problem Definition

We define our cocktail party environment as follows: All possible sounds belong to a finite set of length d , $\{\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_d\}$, composed of *sound characters*. These sounds can be thought of as equal-length temporal snapshots of a speech stream, as with the vector quantization tokens used in commercial speech recognition systems. We define a *word* \mathbf{w}_i as a sequence of q sound characters $\mathbf{w}_i = (\mathbf{s}_{i(1)}, \mathbf{s}_{i(2)}, \dots, \mathbf{s}_{i(q)})$, $1 \leq i(j) \leq d$. The notation $\mathbf{s}_{i(j)}$ signifies the j th sound in word \mathbf{w}_i . For simplicity, we assume that all words consist of the same fixed number q of sound characters. We define a *language* as a sequence of l words $(\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_l)$. The language can be thought of as a very long string of sound characters, containing all valid verbal utterances. To simplify implementation, we require that all words be distinct; that is, no two words are made up of exactly the same set of q sounds in the same order and each word appears exactly once. A word in this model is not directly analogous to an English word but rather is analogous to an English word or phrase in a specific context. We believe that it will not prove difficult to remove the restriction that words must be distinct. For example, we already allow nearly arbitrary sequences of sounds due to the coordination provided by the words, so higher-order structures (phrases composed of words) would allow nearly arbitrary sequences of words.

² The listener is trained to be familiar with the structure of the language: how sounds are formed into words, words into sentences, and so forth.

We can now regard the utterances of a *speaker* as a contiguous subsequence of the language. A speaker picks a particular position j in the language sequence and utters the words in a subsequence beginning with the chosen position: $(\mathbf{w}_j, \mathbf{w}_{j+1}, \dots)$. In our human cocktail party problem, the system is listening to a superposition of p speakers. All speakers are identical except for the specific content. In addition, we disallow multiple speakers from following the same series of words at the same time (an improbable event). We indicate by $\mathbf{s}_{i(j,k)}$ the k th sound uttered by the j th speaker, where the i function is responsible for mapping the k th sound uttered by the j th speaker to the appropriate sound index. In time slot k , the system receives the superposition $\sum_{j=1}^p \mathbf{s}_{i(j,k)}$ of sound characters (one from each speaker in the mixture). The task of the system is to focus on a particular speaker and isolate the word series that speaker is uttering. The system determines which stream to follow based on an expectation, provided externally or formed recursively (by another part of the system), for the content of the desired speech stream. With this general expectation, the system isolates that speaker's speech stream $(\mathbf{w}_{i(g,l)}), l = 1, 2, \dots$, where g is the attended speaker's index, and the i function is defined (in a similar way to the sound mapping function) to map the l th word speaker g utters to its appropriate word index. It is worthwhile to note that the l (i.e., word) timescale's rate is a factor of q (i.e., the number of sounds per word) slower than the k (i.e., sound) timescale's rate.

3 Model Architecture

Sparse binary associative memories form a central building block in our system. Their origin is in the early Learnmatrix work of Steinbuch in the 1950s (Steinbuch, 1963). In the late 1960s Willshaw and his colleagues (1969) advanced the study of such structures and presented a theory of their optimum storage capacity. Since the 1970s sparse binary associative memory architectures have received only modest attention, the work of Palm (1980) and Amari (1989) being of particular note.

Recently, Hecht-Nielsen (1998) proposed a new theory of the cerebral cortex in which sparse binary associative memories again play a central role. According to Hecht-Nielsen, a basic cerebral operation is association, which is performed in two stages (see Figure 1). A sparse set of neurons (a *token*) denoted by a vector \mathbf{u} is assumed active on one cortical region—region X . Through synaptic connections between region X and a second region—region Y —the token on X affects the activation level (weighted sum of inputs) of each of the neurons in Y . Region Y is then restarted; a competition takes place in which the m most active neurons in the region “win.” These m winners' outputs are set to 1, while all other neurons' outputs are set to 0, forming a sparse binary vector \mathbf{v} in region Y . This competitive process is termed a *restart competition* or *restart*. We can say that the network associates

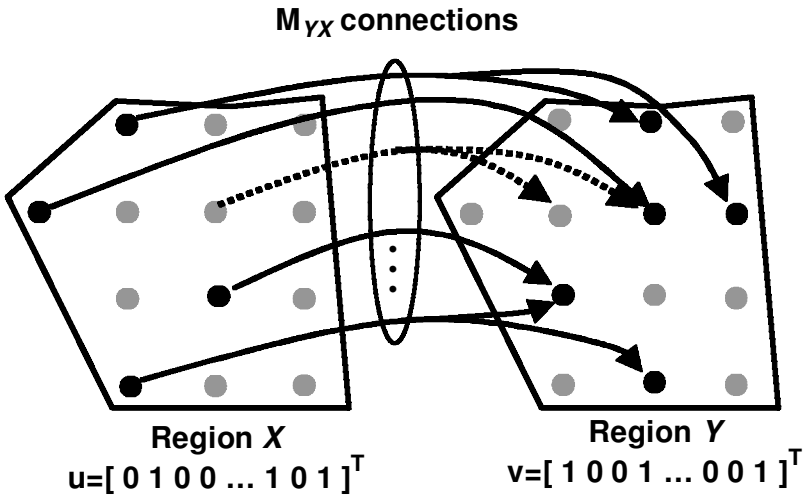


Figure 1: Cortronic regions. Each cortronic region contains neurons, which may become active in a sparse binary pattern. A matrix M_{YX} stores the connection weights between neurons in regions X and Y . At a given time, an active token in region X may broadcast its input to region Y through the weight matrix M_{YX} . If region Y subsequently undergoes a restart competition, a sparse pattern of neurons wins, forming a sparse binary token on region Y . This figure illustrates a token u on X associatively impinging on region Y , which then restarts, yielding a token v . In the general case, connections can go from any region to any other region, as needed, including from a region to itself.

token u in region X with token v in region Y . This recall of associations is dependent on a period of training in which pairs of tokens are introduced to the regions, and boolean Hebbian learning is used to form a binary connection matrix between the regions. For each token u on region X that is to be associated with a token v in region Y , the weight of the connections from the active units in X to the active units in Y is set to 1. Several variations of the training process have been considered in the literature (Palm, 1980)—for instance, binary versus nonbinary connection weights and fixed number of winners versus threshold—and many of these have similar performance characteristics. For simplicity, we use a fixed number of winners and binary connection weights. Relying on previous work in sparse binary associative memories and Hecht-Nielsen’s cortronic artificial neural network architecture, we designed a hierarchical construct of cortronic regions to address the cocktail party problem. During training, the cortronic network learns associations between sound and word tokens. As a consequence of its hierarchical structure, the network learns association in three levels, based on the principle of co-occurrence: that adjacent language constructs must be

related. First, the network learns the association between a particular sound and the sounds that follow it in the language. Next, the network learns the association between a particular sequence of sounds and the token representation of the word they convey. Finally, the network learns the association between a word and the word that follows it in the language. These three levels of association represent the network's knowledge of the language.

When employed in the model cocktail party problem, the system receives a superposition of sounds during each sound sampling interval. The system uses an expectation in the form of a word it anticipates hearing from the attended speaker. In the analogous human cocktail party problem, such an expectation cue may be formed by observing the specific speaker's lip movements, hearing a conversation fragment during a period of relative silence, or in an iterative trial-and-error process. In our model, at the beginning of operation, we assume that an initial expectation cue has been formed by some such mechanism and that it takes the form of a fragment of a word token—a sparse binary vector with very few bits set. This token fragment functions as a tentative initial hypothesis, and the system serves to test this hypothesis. If it is confirmed, then the expectation process continues and the token representations of the incoming sounds being attended to are separated, making them available for the higher cerebral mechanisms of comprehension.

Our cortronic neural network is composed of three different parts (see Figure 2): a sound input region I , q sound processing regions X_1, X_2, \dots, X_q , and a word region Y . Region I preprocesses the incoming sounds and maps them into token representations. It is well known that in the primary auditory cortex, local-in-time features employing multiple timescales, similar to Gabor Logons, are employed (Kohonen, 1996; Rauscheker, 1998).³

As an approximation, we use a set of wavelet-like transforms to convert the sound input into activation levels for the neurons in region I . A competition is then held in which the most active neurons win. This results in a distinct representation of each sound, forming a sparse binary code that can be used in the next stage of processing. The exact number of winners is modified in a way appropriate for the task at hand. During learning, a sparser set of winners is allowed to define an internal representation for a sound, while in the cocktail party environment, a larger number of winners is allowed to represent a superposition of the many possible sounds that could be attended to. The second portion of our architecture, the sound processing area, is composed of q (the number of sounds in each

³ The acoustical dynamics and complexity of the human ear introduce a number of significant effects. For the purposes of this study, however, we made a simplifying assumption that the auditory input system functions approximately as a microphone connected to a rich set of wavelet feature detectors. We expect the current experiments to form a basis on which others can build and include realistic modeling of the acoustic properties of biological ears.

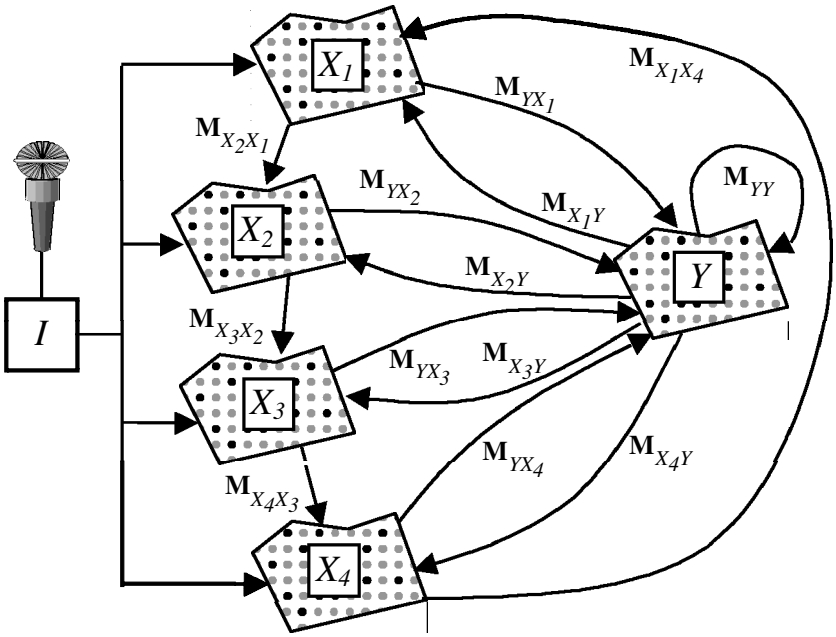


Figure 2: Cortronic network architecture for the model cocktail party problem. Region I represents the early processing apparatus of the raw sounds wherein each isolated pure sound is represented by a unique sparse binary pattern (as would emerge from a competition between feature detectors). Our focus is on information processing in the internal cortronic network within the sound processing regions X_1 , X_2 , X_3 , X_4 , and the word processing region Y .

word) cortronic regions that process consecutive sound tokens, extracting the sounds spoken by the attended speaker from the mixture present on I . Processing is cyclic: region X_1 processes the first sound token in a word, region X_2 processes the second sound token in a word, and so forth. In a biological system, this would correspond to feeding all the auditory data to all portions of the primary auditory cortex all the time, but restarting only the region that is now to respond (Rauscher, 1998). An issue of prime importance is the synchronization between incoming sounds and the appropriate region that processes the sound, as well as the detection of word boundaries. We assume that a cognitive process (i.e., a determinate sequence of cortical restarts) is used to synchronize the sounds and the sound processing regions. Such a process would be driven and sustained by successful recognition of word and sound boundaries. For simplification, we did not model those controlling cognitive processes. It is interesting to note that this need for synchronization between source and processing is evident in our own

experiences with the human cocktail party problem as well (Hecht-Nielsen, 1998). The last layer of our architecture is a single word cortronic region Y , which processes a higher-order representation of words in the language.

Knowledge of the language is conveyed in the connections between the different cortronic regions in our architecture, as follows. First, the sound input region I is connected to each of the q sound processing regions. Second, each sound processing region X_i is connected to the sound processing region X_{i+1} that follows it in the temporal processing sequence, with X_q connected back to X_1 . During the language training process, these connections learn the associations between consecutive sounds in the language. In addition, each sound processing region X_i , $i = 1, 2, \dots, q$ is connected to the word region Y , which in turn is connected back to the sound processing regions. These connections represent the associations between sounds and words in the language. Finally, the word region Y is self-connected. These connections represent the temporal links between sequential words in the language. The appendix describes this architecture and the experimental protocol in detail, using a notation we defined specifically for cortronic network architectures.

An important characteristic of cortronic regions is their *storage capacity*, defined as the number of vector associations the network can recall with acceptable accuracy. Referring back to the simple cortronic network in Figure 1 and using a derivation similar to Amari (1989) and Jagota, Narasimhan, and Regan (1998), assume regions X and Y contain n neurons each. Further, assume that the regions' forward connection matrix \mathbf{M}_{YX} contains elements trained using boolean Hebbian learning by presenting l pairs of uniformly random sparse binary vectors, where each n -bit vector contains m 1's and $n - m$ 0's. We would like to calculate how many pattern pairs l we can store in the connection matrix and still get acceptably accurate recall. In the recall process we activate a trained pattern on region X and restart region Y , retrieving the associated pattern on region Y . Since Hebbian learning connects all m active neurons in X to the m desired output neurons in Y , the recall process would be perfect unless all m active neurons in X are also connected to one or more extraneous neurons in Y . In this case, an erroneous bit may win the competition in region Y . We will proceed to calculate the probability of such an event's happening.

Assuming that the stored vectors are sparse ($m \ll n$), the probability that a particular neuron in X is connected to a particular neuron in Y is approximately $l\frac{m^2}{n^2}$ after Hebbian training of the network. Since each pattern has m active neurons, the probability that a particular vector on X is connected to a particular neuron in Y is $\left(l\frac{m^2}{n^2}\right)^m$. Thus, the probability p that there is no spurious neuron in Y with activation level m is

$$p = \left(1 - \left(\frac{lm^2}{n^2}\right)^m\right)^{n-m}. \quad (3.1)$$

One criterion we can use to optimize the storage capacity of our network is information maximization. The information I in a network with n neurons and represented by the l vector associations is given by the familiar information theory definition $I = l \log_2 \binom{n}{m}$. Applying Stirling’s formula for logarithms, solving equation 3.1 for l , and substituting gives

$$I(n, m) = \frac{n^2}{m^2} \frac{1}{\ln 2} e^{\frac{1}{m} \ln \left(1 - e^{-\frac{1}{n-m} \ln(p)} \right)} \cdot \ln \left(\frac{n^{n+\frac{1}{2}}}{m^{m+\frac{1}{2}} (2\pi)^{\frac{1}{2}} (n-m)^{n-m+\frac{1}{2}}} \right). \tag{3.2}$$

An approximate analytical solution for the optimal value of m and l given n is reached by dropping the lower-order terms above and differentiating, which yields $m \approx \log_2 \left(\frac{n}{4} \right)$ and $l \approx \frac{n^2}{m^2} \ln 2$. Note that the optimal value of m depends logarithmically on n : $m \sim \ln(n)$, justifying our assumption of vector sparseness. A second-order approximation can be derived by taking two terms in the Stirling approximation above, yielding

$$l \approx \frac{n^2}{m^2} e^{\frac{1}{m} \ln \left(1 - e^{-\frac{\ln(p)}{n-m}} \right)}. \tag{3.3}$$

The optimal number of active bits and resulting network capacity are important checks that help in the design of cortronic networks. On modern computers, regions containing on the order of 2048 neurons are computationally tractable; for a 1% probability of spurious activation ($p = 0.99$), approximately 11 bits should be active in each vector and up to 11,000 vector pairs can be stored. However, the above analysis, while instructive as a starting point, assumes a simple one-to-one connectivity between two regions and an absence of noise in the starting pattern. Our architecture contains many regions of one-to-one connectivity, but as shown in Figure 2, it also contains a convergence of the X_i regions onto Y , divergence from Y onto each X_i , use of partial patterns as expectation, noisy patterns, and input that is not completely random. While a complete accounting of these effects is difficult, each can be examined qualitatively. The case of q regions of size n converging to one is equivalent to the case of a single region of size qn providing input to a region of size n . Since m depends logarithmically on n , the optimal number of active bits will be raised only slightly from the original estimate. The active bits are now shared between q separate regions, so approximately m/q active bits should be used in each region. However, since we also rely on the divergent connection from Y back to each X_i , which is equivalent to the standard one given above, we cannot afford to use fewer active bits on the X_i regions. Note that there is some benefit to this arrangement also: the chance that there will be no spurious

activations on Y is now approximately

$$p = \left(1 - \left(\frac{lm^2}{n^2} \right)^{qm} \right)^{n-m} \tag{3.4}$$

Comparison of equations 3.1 and 3.4 shows that the chance of spurious activation is reduced exponentially with q . If the network is not over capacity, recovery of words on region Y will therefore be exponentially more accurate than recovery of individual sounds. Since the goal of our system is to recover speech at the word level, this is exactly the desired behavior.

In a noisy environment, we may lose a fraction of our active bits to noise. The neurons in the correct pattern will then only be guaranteed an activation level of mz , where z is the fraction of bits unaffected by noise. To compensate, we may scale the number of active bits by a factor of r . Assuming that noise scales linearly with the number of active bits, our approximate formula for the probability of correct recall becomes

$$p = \left(1 - \left(\frac{mr}{mzr} \right) \left(\frac{lm^2r^2}{n^2} \right)^{mzr} \right)^{n-mr} \tag{3.5}$$

If we assume $n \gg mr$ and apply Stirling's formula, we find that

$$p \approx \left(1 - \frac{1}{\sqrt{2\pi m z(1-z)r}} \left(\frac{lm^2r^2}{z(1-z)^{(1-z)/zn^2}} \right)^{mzr} \right)^n \tag{3.6}$$

$$\frac{dp}{dr} \approx p^{\frac{n-1}{n}} n \frac{(Ar)^{2mzr}}{\sqrt{Br}} \left(\frac{1}{2r} + 2mz \left(\ln \left(\frac{1}{Ar} \right) - 1 \right) \right) \tag{3.7}$$

where

$$A^2 = \frac{lm^2}{z(1-z)^{(1-z)/zn^2}}, B = 2\pi m z(1-z). \tag{3.8}$$

Note that the derivative is positive when $Ar < e^{-1}$. From the definition of A , we can see that this requirement is fulfilled when the system is running sufficiently below capacity, that is, with l sufficiently small. Even without knowing exact values for noise, this suggests a strategy for noisy conditions: run the system below capacity and increase the number of active bits in the sparse representation. If the level of noise is known, equation 3.7 can be used to maximize p as a function of r ; if not, experimentation can yield an optimal value for mr , the number of active bits.

Our system not only contains noise of various types but also uses vectors, generated from real-world data as described below, that are not completely uncorrelated. So we made qualitatively informed adjustments to the optimal parameters derived for the simple case of two connected regions. The specific parameters used are described below.

sparse binary pattern representing the sound input.⁴ We control the number of winners in the competition differently depending on whether the network is learning (with a single speaker) or performing (in the cocktail party environment).

Before tackling the cocktail party problem, the network must first be trained on the language. The connections between the regions in the cortic network, described in the previous section, are trained by boolean Hebbian learning in response to sequential input of language constructs under noise-free conditions. In the biologically analogous situation, the connections would be made gradually over time as a human learns a language by being exposed to it. Specifically, we train each of the connection matrices as follows. The connections between the sound input region and each of the sound processing regions X_1, X_2, \dots, X_q , that is, $\mathbf{M}_{X_1 I}, \dots, \mathbf{M}_{X_q I}$, are set as random row permutations of the identity matrix, to emulate a lookup table. Exactly m winners on I are allowed during training. This converts each real sound \mathbf{s}^R into its sparse binary representation \mathbf{s}^B and maps that representation to a different random token \mathbf{x}_i on each sound region X_i , $i = 1, 2, \dots, q$. For each word in the language, we impose the sounds in the word through the input region and onto the appropriate sound processing region X_i . In other words, the sparse binary token that corresponds to the first sound in the word, as it passes through the input region, is imposed on region X_1 , the sparse binary token that corresponds to the second sound in the word, as it passes through the input region, is imposed on region X_2 , and so on. The architecture learns the connections between each sound processing region X_i and its (temporally) consecutive sound processing region X_{i+1} (as well as, cyclically, between X_q and X_1), as captured in the connection matrices $\mathbf{M}_{X_2 X_1}, \mathbf{M}_{X_3 X_2}, \dots, \mathbf{M}_{X_q X_{q-1}}, \mathbf{M}_{X_1 X_q}$. In addition, a randomly picked token corresponding to the word represented by the above q sound tokens is imposed on region Y , and the connections $\mathbf{M}_{Y X_1}, \dots, \mathbf{M}_{Y X_q}$, as well as the reciprocal connections $\mathbf{M}_{X_1 Y}, \dots, \mathbf{M}_{X_q Y}$, between each of the X regions and region Y are trained. Finally, as the input sound streams proceed from one word to the next, we train the Y to Y connections $\mathbf{M}_{Y Y}$.

To simulate the cocktail party environment, we pick p speakers and feed the network an additive superposition of their speech streams. In this superposition, we take the amplitude of all speech characters in the speech streams to be approximately equal. Next, we follow a series of restarts of the different regions. Our architecture uses an expectation to process the incoming mixture. Initially, we set this expectation \mathbf{e} to be a fragment of the first word token we expect to hear from the attended speaker. At $t = 1$ we feed the arithmetic superposition of preprocessed sounds from all speakers through the input region. For small numbers of speakers p , we allow

⁴ Other preprocessing approaches may provide equal or superior performance. The main goal is for the preprocessing to provide a robust yet sufficiently segmented representation of the input.

m winners on the input region; in this case, the complete token for every sound could be present, depending on the nature of the input sounds. For larger numbers of speakers, however, this would lead to sound tokens apparently being present when they were not, due to overlapping patterns. We therefore restrict the number of winners such that accidental patterns are unlikely. This restriction is performed empirically by requiring a “hallucination” of less than 6%, as described below. Although we perform this operation manually in our system, a test for excessive accidental patterns could be included as part of the processing operation. Even a fixed maximum number of winners, $5m$ for instance, provides similar performance for fewer than 10 speakers.

Once winners—usually many more than m —have been selected on I , they are mapped onto region X_1 . Added to this input signal on X_1 is the expectation \mathbf{e} from the word region Y . This token fragment \mathbf{e} projects onto X_1 through the binary weight matrix $\mathbf{M}_{X_1 Y}$ at an intensity of α (a predetermined weighting parameter). We now conduct a restart competition on region X_1 . The result of this restart is a sparse vector with only m active bits. This vector is equivalent to a hypothesis—it may or may not contain active units that correctly match the sound associated with the expected word. We will call this active token $\mathbf{x}'_1(1)$, as it approximates the sound token $\mathbf{x}_1(1)$ that corresponds to the sound that the attended speaker uttered at $t = 1$. At $t = 2$ we feed the arithmetic superposition of sounds from all speakers through I and onto region X_2 . Note that we have moved one time step forward along each of the superposed speech streams. In addition, we add the expectation from the word region as well as signal from the active token $\mathbf{x}'_1(1)$ on X_1 . That is, $\mathbf{x}'_1(1)$ is projected associatively onto X_2 through the binary weight matrix $\mathbf{M}_{X_2 X_1}$ at an intensity of β (another predetermined weighting parameter). As before, we hold a restart competition in X_2 resulting in $\mathbf{x}'_2(2)$, an approximation for the second sound the attended speaker uttered. In the following time step we repeat the above process on regions X_3, \dots, X_q to form all q sounds that compose the first word $\mathbf{x}'_1(1), \mathbf{x}'_2(2), \dots, \mathbf{x}'_q(q)$.

Now we would like to recover the first word the attended speaker uttered. We project the recovered sound tokens $\mathbf{x}'_1(1)$ through $\mathbf{x}'_q(q)$ on regions X_1 through X_q , respectively, to the word region Y . We hold a restart competition on the word region to form $\mathbf{y}'(1)$, an approximation to the token for the first word uttered by the attended speaker.⁵ This vector can be com-

⁵ The goal in the human cocktail party problem is to attend to a single speaker in a mix rather than to separate the sources themselves. In this problem, the actual sound is never reconstructed from the token. The wavelet-based speech preprocessor we employed does not preserve phase and amplitude relationships in the original audio signal. In the context of this model problem, it is unnecessary (and, in fact, is impossible for our system) to reconstitute the original signal from this representation. Our experiment demonstrates expectation-assisted source attendance at a higher level of abstraction than source separation.

pared to $\mathbf{y}(1)$, the token representation of the attended speaker's first word, to obtain a performance measure. To form an expectation that will assist in recovering the second word's sounds, we self-feed the word region through the \mathbf{M}_{YY} associative connection matrix. Restarting the Y region generates the expectation token for the next word's sounds. We will denote this automatically generated expectation token as $\mathbf{y}^*(2)$. It represents the next stage of the process of synchronization with the desired sound stream.⁶ Note again that \mathbf{e} —a single fragment of the starting word, representing a vague expectation—initiated the entire sequence of processing. From then on, processing is self-sustaining.

For the next q time steps that compose the second word's sounds, we proceed as with the first word's sounds with the following changes. The full token \mathbf{y}^* is used, whereas before we used a fragment \mathbf{e} of the word token. This implies a greater strength of expectation. The restart of region X_1 that generates the first sound in the second word benefits from a cyclical input from region X_q of the last sound in the previous word. This procedure generates $\mathbf{x}'(q+1)$ through $\mathbf{x}'(2q)$. As before, we associatively feed the recovered sounds $\mathbf{x}'(q+1)$ through $\mathbf{x}'(2q)$ to the word region Y . Restarting the word region produces the second recovered word. Again, we can derive a performance measure by comparing the recovered word $\mathbf{y}'(2)$ with the token that represents the second original word $\mathbf{y}(2)$ that our attended speaker uttered.

We can continue by applying equivalent steps to recover additional sounds and words. It is easy to detect if the initial sound fragment hypothesis was incorrect, and the system can be restarted when a new hypothesis is supplied. The appendix presents in a succinct code the operations that the cortronic network performs throughout its operation.

5 Results

Most of our experiments were made with the parameters presented in Table 1. We constructed the speakers' language subsequences by picking the beginning of each subsequence uniquely at random from the language string. A new set of speakers was picked for each of the 1000 trials used to generate each data point. As a performance metric, we used the fraction of correctly retrieved bits in the sparse binary vectors.

The information capacity of the network at the given size, as derived in equation 3.6, is $l = 7600$ patterns for a mean recall error of 1%. Our network is scaled so the capacity exceeds the number of words, in order to compensate for correlated input vectors and to provide improved performance under the noisy conditions of multiple interfering speakers. Note also that

⁶ Processing of incoming sensory data is a highly active operation. The vector $\mathbf{y}^*(2)$ represents the feedback expectation that will be used in the processing of the next word. This is one of the ways in which knowledge of the language is explicitly used.

Table 1: Typical Experimental Parameters.

Description	Symbol	Value
Number of sounds per word	q	4
Number of unique sounds	d	254
Number of words in the language	l	1024
Number of neurons in each cortronic region	n	2048
Number of active units in a sparse binary vector	m	16
Expectation weighting	α	0.25
Weighting of previous sound	β	0.05

the number of active units has been scaled upward from the theoretical optimum of 11 to 16, again to compensate for noise. This number was chosen empirically to give reasonable performance.

Figure 4 shows the performance of our network in retrieving the sounds and words that the attended speaker uttered. Word recovery is superior to sound recovery. This is analogous to the human cocktail party experience in that a listener may isolate the content of a speaker from the cocktail but often cannot identically reproduce the speaker's sound stream. In addition, both sound recovery and word recovery improve over time. This improvement is most noticeable between word 1 and word 2, while improvement after word 3 is minute. Put differently, it is more difficult to follow a particular speaker's conversation at its beginning, when the listener's expectations are still vague. As the conversation progresses, it is easier for the listener to isolate and follow the attended speaker's speech.

To test the system more thoroughly, we conducted a simple experiment in which the attended speaker's sounds were deliberately omitted from the input so that the word expectations mismatched the sound input. As Figure 5 depicts, except for the trivial case of no input signal, the network did not get confused; it did not erroneously generate the expected speech stream, which was absent from the auditory environment. If expectation is weighted too heavily, the "recovery" of a nonexistent speech stream can occur regardless of the input, a process we term *hallucination*. We kept hallucination to under one bit (6%) by restricting the number of winners on I .

Finally, we tested the scalability of our solution by measuring word recovery performance with various network and language sizes (see Figure 6). As the size of the vocabulary grows beyond the region's capacity, performance begins to degrade gradually and later drops more precipitously. As expected, using larger region sizes allows the system to handle larger vocabularies.

6 Discussion

Previous approaches to the cocktail party problem have used a blind approach. For instance, methods based on independent component analysis

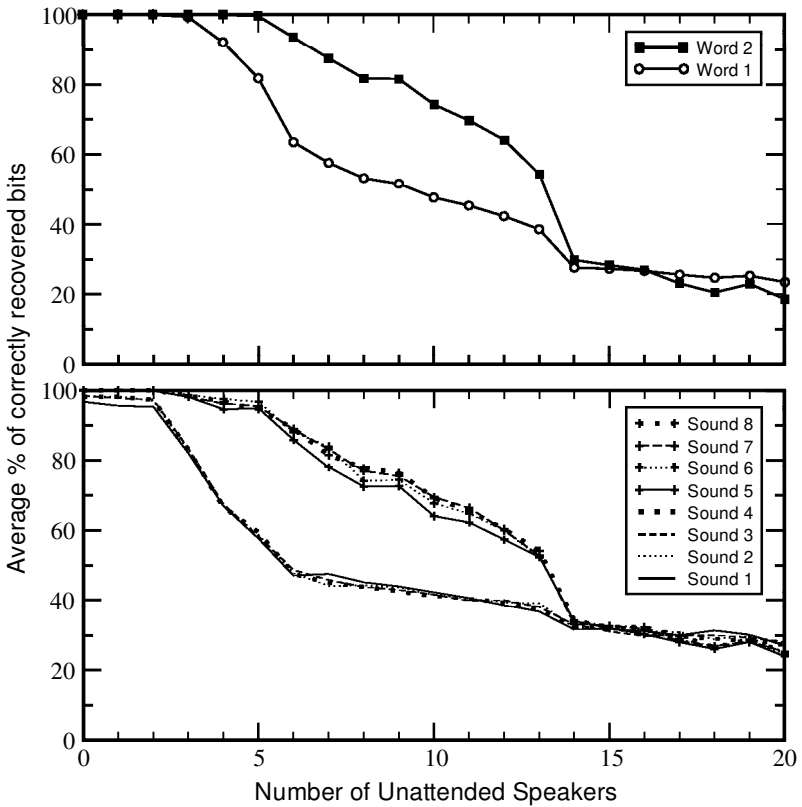


Figure 4: Word and sound recovery as functions of the number of simultaneous speakers including the attended speaker. The cortronic network successfully extracts words from the mix. Both sound recovery and word recovery improve temporally. Word recovery performance is superior to sound recovery. This is analogous to the human cocktail party: a listener may isolate the content of a speaker from the cocktail but cannot identically reproduce the speaker's sound stream. The following parameters apply to these data: number of words in language $l = 1024$; number of unique sounds $d = 254$; number of sounds per word $q = 4$; number of neurons per region $n = 2048$; number of active neurons in a sparse binary vector $m = 16$; number of expectation bits = 4 for first word; $\alpha = 0.25$, $\beta = 0.05$; 1000 trials per data point.

can be highly effective, reconstructing 20 or more sources with virtually no error (Lee, Girolami, Bell, & Sejnowski, 1999), given at least as many microphones as sources. However, methods that allow fewer microphones report results that are more modest; systems attempting to separate three sources with two microphones generate signal-to-noise ratios (SNRs) of a

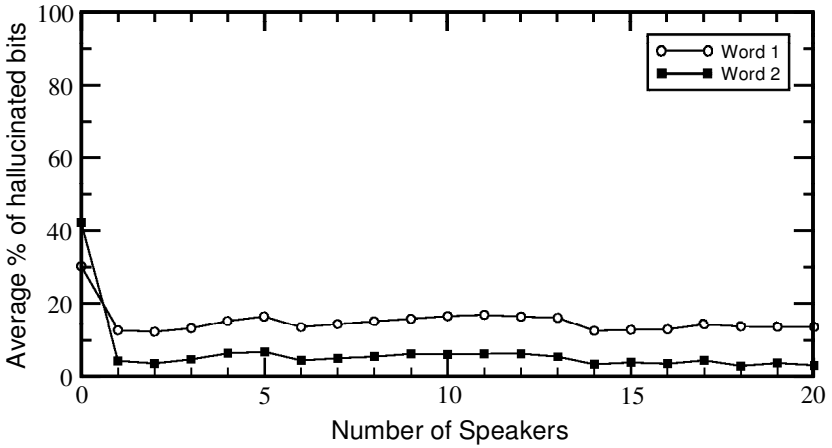


Figure 5: Word hallucination. Hallucination refers to the performance of the system when the attended speaker signal is missing from the cocktail. The expectation is strong enough to complete the desired words when there is no sound input to the cortronic network. However, with any number of real input sources present, the network does not reproduce the desired words. Thus, the network does not suffer from word hallucination and will not erroneously pick out a source that is not present in the input stream. The parameters used in this experiment are identical to those used in the word and sound recovery experiment of Figure 4.

few decibels at most (Van Hulle, Yu-Hen, Larsen, Wilson, & Douglas, 1999). This is not surprising given the difficulty of reconstructing all sources, of arbitrary type, from a small number of mixtures; the problem can be mathematically underdetermined. In contrast, relying on intimate knowledge of the source allows our system to identify the desired content for large numbers of superposed sources using a single microphone. In our model problem, we were able to isolate the content of an attended speaker from up to approximately 10 others with minimal noise. Indeed, our system may seem to provide speaker separation performance that surpasses human capability, probably because of the relatively small vocabulary used in our model. However, the scalability of our model indicates the possibility of application to problems with larger vocabularies. In addition, traditional blind source separation could be used to complement our approach. Multiple inputs such as two ears could be used to generate partially separated output prior to an expectation-based system.

The performance of our system is insensitive to moderate parameter changes; there is a broad range of parameter space in which it performs adequately. Such parameter space robustness is what we would expect from

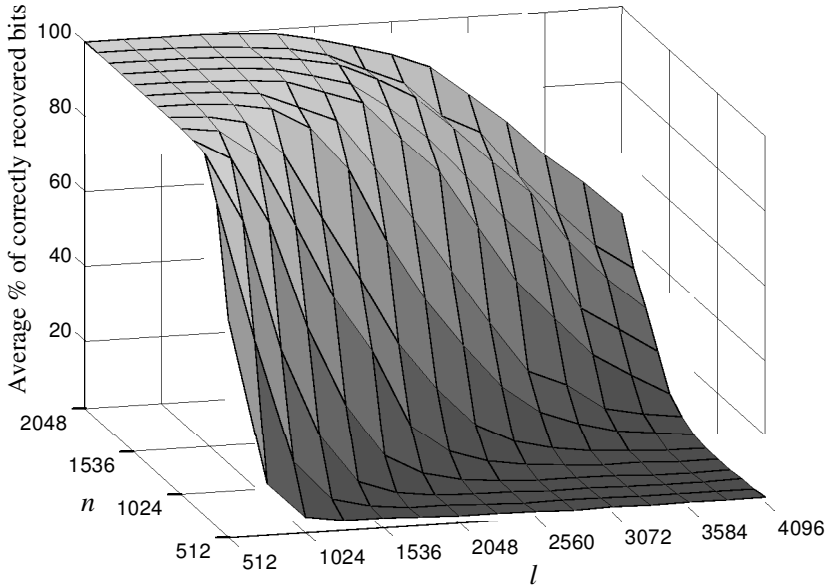


Figure 6: Scalability of the cocktail party cortronic network as a function of number of neurons in each region (n) and number of words in the language (l). As the vocabulary size increases, performance degrades gradually at first, before falling off rapidly. As expected, using a larger region size allows the system to handle larger vocabularies. The following parameters apply to these data: number of unique sounds $d = 254$; number of sounds per word $q = 4$; number of active neurons in a sparse binary vector $m = 16$; number of expectation bits = 4 for first word; $\alpha = 0.25$, $\beta = 0.05$; number of speakers $p = 4$; 1000 trials per data point.

a real biological system. Better results would be possible through the use of token completion, a feature of the associative memory model employed in the cortronic network (see Hecht-Nielsen, 1998, and Schwenker, Sommer, & Palm, 1996, for a description of this mechanism). We believe that the restriction we placed by setting a fixed number of sounds per word and relying on an external mechanism for word and sound synchronization can be overcome, as cortronic networks are suited for internally controlling their operation. For example, internal management of expectation weighting and number of winners could be implemented with a “sanity check” mechanism that would test for the apparent presence of inputs that were known to be highly improbable.

A key advantage of our approach is that while we require knowledge of the sounds we wish to attend to, we require no knowledge of background

sounds such as static, traffic, and music. This is because of the mapping of raw sounds onto the largely random tokens at the front end of the system. The probability of the background sounds' interfering significantly with our known sounds is negligibly small. During training, the network also requires no knowledge of the specific superposition of sources. The network, trained on an isolated source, does not need retraining to isolate that source from a variety of different mixtures, avoiding the need for additional, time-consuming computation when addressing novel environments.

Additionally, the cortronic architecture is based on a simplification of observed biological structures and capabilities. We use a simple form of Hebbian learning; real neurons are capable of more complex interactions. The restart competition provides a simplified way to implement sparse coding of data, with each neuron being either on or off. Biological systems that use sparse coding, for instance, hippocampal place fields (Wilson & McNaughton, 1993), typically have graded activity for each neuron. Although the exact operation of our cortronic network is unlikely to be duplicated in a biological system, the simplified network suggests an approach to solving the cocktail party problem that may be analogous to strategies used by the human brain. On a final note, the processing of information in the cortronic network is massively parallel, as is the biologically analogous human cerebral cortex. The use of dedicated hardware would enable cortronic networks of very large capacity with reasonable processing speed and no start-up delay.

Appendix

This appendix presents a concise definition of the architecture of the cortronic network in our simulation and the processing operations that it uses. For an overview of the basic concepts behind cortronic network operation, see the main text.

We use a specialized code, defined below, to describe the architecture and processing functions of our cortronic network. Use of this code allows the cortronic network to be described with a series of succinct operations that mimic the actual processing operations necessary to implement such a network on a computer.

We use the following terms throughout the appendix:

- A *sparse binary vector* is an n -ary vector with binary elements such that the vector contains many more 0's than 1's.
- A *cortronic region* (or simply a region) consists of n elements (neurons), each of which can be either active or inactive. The *pattern of activity* on a region is the binary representation of those elements that are active (1 = active, 0 = inactive). Typically this pattern is a sparse binary vector.

- An *associative connection* between two regions consists of a binary $n \times n$ matrix that specifies the elements in the second region that receive input from any particular element in the first region.
- A *restart competition* on a region modifies its pattern of activity based on which elements are receiving the greatest input from other active elements. The pattern of activity does not change based on input without a restart competition.
- A *cortronic network* consists of a number of cortronic regions that are associatively connected. This network represents objects as sparse binary codes, learns associations between objects by training the associative connection matrices, and then uses these trained associations to process information using restart competitions.

We use the following notation for components of a cortronic network:

X, Y, Z, \dots Uppercase italic letters denote regions. When used in a mathematical expression, they denote the column vector representing the pattern of activity of the region (typically an n -ary sparse binary vector with m bits active).

$\mathbf{x}, \mathbf{y}, \mathbf{z}, \dots$ Lowercase bold letters denote sparse binary vectors. Unless stated otherwise, all vectors are assumed to be of length n and have exactly m bits active.

\mathbf{M}_{YX} Uppercase bold letters denote $n \times n$ binary weight matrices, in this example corresponding to connections from region X to region Y . Note that \mathbf{M}_{XX} would be a matrix that connects region X to itself.

To define the associative connections between regions, we use the following notation:

$X \rightarrow^\alpha Y$ X connects to Y with strength α via a matrix \mathbf{M}_{YX} . When such a connection is present, X sends as input to Y the vector $\alpha_{YX} \mathbf{M}_{YX} X$, where X is viewed as a column vector. (Note that $Y \rightarrow X$ is distinct from $X \rightarrow Y$.)

$X \rightarrow Y$ Shorthand for $X \rightarrow^1 Y$, that is, $\alpha = 1$.

$\alpha_{YX} = \alpha'$ Change the strength of the connection to Y from X from α_{YX} to α' . One biologically analogous situation occurs where intermediate neurons that propagate a signal from X to Y are activated or inhibited, thus modulating the strength of the signal.

Operations used primarily during training are given below:

- $G(\mathbf{x})$ Generate a sparse binary vector \mathbf{x} uniformly at random.
- $G(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_k)$ Generate k distinct random sparse binary vectors $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_k$. (Distinct means $i \neq j \Rightarrow \mathbf{x}_i \neq \mathbf{x}_j$.)
- $\mathbf{x} \downarrow X$ Activate \mathbf{x} on X : the pattern of activity on region X becomes the sparse binary vector \mathbf{x} .
- $T(Y, X)$ Associate the pattern on X to the pattern on Y : boolean Hebbian learning is used to add weights to the connection matrix \mathbf{M}_{YX} corresponding to the pattern of activity on X and Y . If X and Y are viewed as column vectors, this is equivalent to $\mathbf{M}_{YX, (new)} = \mathbf{M}_{YX, (old)} \vee YX^T$ where \vee denotes the boolean OR operation. In essence, this connects every currently active element in X to every currently active element in Y .
- $T(Y)$ Associate the current pattern of activity on Y with the pattern of activity on Y that existed before the last association or restart.

We use the information processing operations defined below:

- $R(X)$ Restart X : Perform a restart competition on X . The total input to X is given by $\sum_Z \alpha_{XZ} \mathbf{M}_{XZ} Z$, where Z ranges over all regions in the network that are connected to X . After the restart, the pattern of activity is changed to a sparse binary vector with m bits active by picking the active elements that had maximal total input before the restart. In the case where $k > m$ elements could become active (because of tied input scores), elements that are tied for least input among the k are discarded uniformly at random until exactly m elements remain active. Note that this newly active pattern now provides input to all regions that X connects to.
- $R_k(X)$ Restart X and allow k winners.
- $R(X) \rightarrow \mathbf{x}$ Restart X and use \mathbf{x} to denote the resulting pattern of activity.

We introduce the following notation to help remind readers of the implicit broadcast of active patterns that takes place in a cortronic network. The notation does not affect the actual processing operations.

- $\mathbf{x}: X$ \mathbf{x} , the pattern of activity on X . This is used to remind the reader explicitly which pattern is active on X .
- $X \mapsto^\alpha Y$ X provides input to Y of strength α . This reminds readers that when X is active, it sends input to Y through \mathbf{M}_{YX} , and that

this input to Y is relevant for the current processing step. (We abbreviate $X \mapsto^1 Y$ as $X \mapsto Y$.)

$x: X \mapsto^\alpha Y$ Shorthand for $x: X, X \mapsto^\alpha Y$ (and similarly for $x: X \mapsto Y$).

+

Used to indicate that multiple inputs provide the total input to one region. For instance, we would write $X \mapsto Z + Y \mapsto Z$.

To illustrate the usage of the above notation, we reproduce the architecture and simple associative recall shown in Figure 1:

- $X \rightarrow Y$ Define the architecture. Region X is associatively connected to region Y .
- $G(\mathbf{u}), G(\mathbf{v})$ Generate the sparse binary vectors \mathbf{u} and \mathbf{v} .
- $\mathbf{u} \downarrow X, \mathbf{v} \downarrow Y, T(Y, X)$ Activate \mathbf{u} on X and \mathbf{v} on Y and train an association between them.
- $\mathbf{u} \downarrow X, R(Y) \rightarrow \mathbf{v}'$ Perform associative recall by activating \mathbf{u} on X and restarting Y to generate \mathbf{v}' , the recalled pattern corresponding to \mathbf{v} . (If recall is perfect, $\mathbf{v} = \mathbf{v}'$.)

In order to implement the cocktail party problem cortronic network of Figure 2, recall that we start with I , the sound input region, $X_j, j = 1..q$, the sound processing regions, and Y , the word processing region.

- $I \rightarrow X_1, I \rightarrow X_2, \dots, I \rightarrow X_q$ The sound input region connects to all sound processing regions.
- $X_1 \rightarrow^\beta X_2, \dots, X_{q-1} \rightarrow^\beta X_q, X_q \rightarrow^\beta X_1$ The sound processing regions each connect to the following region (cyclically) with strength β .
- $X_1 \rightarrow Y, X_2 \rightarrow Y, \dots, X_q \rightarrow Y$ All sound processing regions connect to the word processing region.
- $Y \rightarrow^\alpha X_1, \dots, Y \rightarrow^\alpha X_q$ The word processing region connects to each sound processing region with weight α .
- $Y \rightarrow Y$ The word processing region connects forward in time to itself.

Recall that the complete synthetic language of our speakers consists of a series of words $(\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_l)$, each of which consists of a sequence of sounds from the set of sounds $\{\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_d\}$. We use sound segments taken

from human speech $\{\mathbf{s}_1^R, \mathbf{s}_2^R, \dots, \mathbf{s}_d^R\}$ to generate internal binary representations $\{\mathbf{s}_1^B, \mathbf{s}_2^B, \dots, \mathbf{s}_d^B\}$, but we need tokens to represent each word:

$G(\mathbf{w}_2, \mathbf{w}_2, \dots, \mathbf{w}_l)$ Generate a unique sparse binary representation for each word. In addition, we pick a unique sequence $(\mathbf{s}_{i(j,1)}, \mathbf{s}_{i(j,2)}, \dots, \mathbf{s}_{i(j,q)})$ of q sounds corresponding to each word \mathbf{w}_j . Here we define $i(j, k)$ to be the index of the k th sound of word \mathbf{w}_j .

Before training, we first pick weight matrices connecting I to each of X_1, X_2, \dots, X_q , which are row permutations of the identity matrix. This simulates a random learned connection between the representation of a sound on I and the corresponding one on each X_i without the processing overhead of actually training such connections. In an actual implementation, such permutations of identity can be simulated by a lookup table. Training then proceeds as follows, where we use $\mathbf{x}_{i(j,k)}$ to refer to the representation of sound $\mathbf{s}_{i(j,k)}$ on the sound processing region X_k . (Note that in normal operation, X_k always represents the k th sound of each word, so this is a sufficiently general notation.)

$\alpha_{X_2 X_1} = 0, \dots, \alpha_{X_q X_{q-1}} = 0, \alpha_{X_1 X_q} = 0,$ Decrease the strength of connections to avoid interference with training.
 $\alpha_{X_2 Y} = 0, \dots, \alpha_{X_q Y} = 0$

$\mathbf{s}_{i(1,1)}^R \downarrow I, R(I) \rightarrow \mathbf{s}_{i(1,1)}^B, R(X_1) \rightarrow \mathbf{x}_{i(1,1)}$ Note that $I \mapsto X_1$. Listen to the first sound of the first word, find its binary representation on I , and translate it to X_1 . This also defines the representation $\mathbf{x}_i(1, 1)$ on X_1 of the first sound of the first word.

$\mathbf{s}_{i(1,2)} \downarrow I, R(I), R(X_2) \rightarrow \mathbf{x}_{i(1,2)},$ Listen to the second sound, activate it on X_2 , and associate the representation of the second sound with that of the first. (Note that
 $T(X_2, X_1)$ $I \mapsto X_2$.)

... Continue for all the sounds of the first word.

$\mathbf{s}_{i(1,q)} \downarrow I, R(I), R(X_q) \rightarrow \mathbf{x}_{i(1,q)},$ Associate the last two sound representations of the first word.
 $T(X_q, X_{q-1})$

$\mathbf{w}_1 \downarrow Y, T(X_1, Y), T(Y, X_1),$ Associate all the sound representations with the word and vice versa.
 $\dots, T(X_q, Y), T(Y, X_q)$

$\mathbf{s}_{i(2,1)} \downarrow I, R(I), R(X_1) \rightarrow \mathbf{x}_{i(2,1)}, T(X_1, X_q)$	Associate the last sound representation of the first word with the first of the second word.
...	Continue for the second word, and associate all the sound representations with that word.
$T(Y)$	Associate the current word (the second) with the previous (the first).
...	Continue associating sound representations with following ones, words with following words, and sound representations with their word and vice versa.
$\mathbf{w}_1 \downarrow Y, T(X_1, Y), T(Y, X_1), \dots, T(X_q, Y), T(Y, X_q)$	Associate the last sequence of sound representations with the last word.
$T(Y)$	Associate the last two words.
$\mathbf{s}_{i(1,1)} \downarrow I, R(I), R(X_1), T(X_1, X_q), \mathbf{w}_1 \downarrow Y, T(Y)$	Wrap around by associating the last word with the first word, and the last sound representation of the last word with the first sound representation of the first word.
$\alpha_{X_2 X_1} = \beta, \dots, \alpha_{X_q X_{q-1}} = \beta, \alpha_{X_1 X_q} = \beta, \alpha_{X_1 Y} = \alpha, \dots, \alpha_{X_q Y} = \alpha$	Set connections to the appropriate strength for processing.

Now we apply the network to the cocktail party problem. Let there be p speakers, each uttering words from a distinct point in the language. Let $i(j, k)$ be the index of the k th sound uttered by the j th speaker where i is an index to a sound vector or the representation of a sound vector on a sound processing region, and let it be the index of the k th word uttered by the j th speaker when applied as an index to a word vector. Let $g \in \{1, 2, \dots, p\}$ be the index of the speaker to whom we are attending. Let a pattern \mathbf{e} be active on Y that contains a few bits of the word we are listening for (expectation). Let h be the number of bits we allow active on I to serve as a binary representation of the possible sounds present. Empirically we found that with our parameters, for $p < 5$, $h = mp$ is appropriate (to approximate the superposition of the binary representations of the sounds). For $p > 5$, $h = 5m$ provides more discriminating performance; we picked h as large as

possible such that hallucination remains under 6% on the second word for a given number of speakers. (Note that h is a function of p , but need only be calculated once for a given number of speakers, not every time the system processes input.)

$\left(\sum_{j=1}^p \mathbf{s}_{i(j,1)}^R\right) \downarrow I, R_h(I), R(X_1) \rightarrow \mathbf{x}'_{i(g,1)}$ Note that $I \mapsto X_1 + \mathbf{e}$: $Y \mapsto^\alpha X_1$. Process the first sound, generating first a rich internal representation of the input mixture on I , and then generating the recovered sound representation $\mathbf{x}'_{i(g,1)}$ from that mixture plus expectation. We can score the accuracy of this recovery by counting the percentage of correctly placed active bits in $\mathbf{x}'_{i(g,1)}$ relative to the representation $\mathbf{x}_{i(g,1)}$ generated during noise-free training. (This percentage is equivalent to the fraction $\mathbf{x}_{i(g,1)} \cdot \mathbf{x}'_{i(g,1)} / m$.)

$\left(\sum_{j=1}^p \mathbf{s}_{i(j,2)}^R\right) \downarrow I, R_h(I), R(X_2) \rightarrow \mathbf{x}'_{i(g,2)}$ Process the second sound now using X_1 feedforward from X_1 . (We now have $I \mapsto X_2 + \mathbf{e}$: $Y \mapsto^\alpha X_2 + \mathbf{x}'_{i(g,1)}$: $X_1 \mapsto^\beta X_2$.)

...

Repeat for all q sounds.

$\alpha_\gamma = 0$

We have to confirm our expectation before we can generate the next word, so turn off the feedforward connection.

$R(Y) \rightarrow \mathbf{w}'_{i(g,1)}$

(Note that $\mathbf{x}'_{i(g,1)}$: $X_1 \mapsto Y, \dots$, $\mathbf{x}'_{i(g,q)}$: $X_q \mapsto Y$.) Restart Y generating $\mathbf{w}'_{i(g,1)}$, an approximation of $\mathbf{w}_{i(g,1)}$, the first word spoken by the attended speaker. Accuracy can be determined as with sounds.

$\alpha_{YX_1} = 0, \dots, \alpha_{YX_q} = 0$

Ignore current sound input to word region. Turn on feedforward connections in the word region.

$\alpha_{Y\gamma} = 1$

$R(Y) \rightarrow \mathbf{w}^*_{i(g,2)}$

Restart Y to compute the expectation for the next word to be spoken. (Input is from the previous word, as $\mathbf{w}'_{i(g,1)}$: $Y \mapsto Y$.)

$$\alpha_{YX_1}, \dots, \alpha_{YX_q} = 1$$

Effect of sound input resumes.

$$\left(\sum_{j=1}^p \mathbf{s}_{i(j,q+1)}^R \right) \downarrow I, R_h(I)$$

$$R(X_1) \rightarrow \mathbf{x}'_{i(g,q+1)}$$

Repeat the above process again, now using $\mathbf{w}_{i(g,2)}^*$ as the self-generated expectation, which replaces \mathbf{e} . (Now $I \mapsto X_1 + \mathbf{w}_{i(g,2)}^*; Y \mapsto^\alpha X_1 + \mathbf{x}'_{i(g,q)}$; $X_q \mapsto^\beta X_1$.)

...

Continue word recovery for as many iterations as desired.

To test whether the network will hallucinate a speech stream based on expectation, when the actual corresponding sound input is not available, we generate a $p + 1$ st imaginary speaker who does not actually contribute to the superposition of sounds. We then set $g = p + 1$ and run the cortronic network as above. If a trial run were to generate significant hallucination, an appropriate correction would be to lower the value of h used for that p , or to decrease the strengths of expectation, α and β , as appropriate.

Acknowledgments

R. H-N. gratefully acknowledges support of his research by DARPA (N00014-98-C-0356) and ONR (N0014-96-C-0109).

References

- Amari, S. (1989). Characteristics of sparsely encoded associative memory. *Neural Networks, 2*, 451–457.
- Amari, S., & Cichocki, A. (1998). Adaptive blind signal processing—neural network approaches. *Proceedings of the IEEE, 86(10)*, 2026–2048.
- Amari, S., Cichocki, A., & Yang, H. H. (1996). A new learning algorithm for blind signal separation. In D. Touretzky, M. Mozer, & M. Hasselmo (Eds.), *Advances in neural information processing systems, 8* (pp. 757–763). Cambridge, MA: MIT Press.
- Bell, A. J., & Sejnowski, T. J. (1995). An information-maximization approach to blind separation and blind deconvolution. *Neural Computation, 7(6)*, 1129–1159.
- Comon, P. (1994). Independent component analysis, a new concept? *Signal Processing, 36*, 287–314.
- Hecht-Nielsen, R. (1998). A theory of the cerebral cortex. In *Proceedings of the 1998 International Conference on Neural Information Processing (ICONIP '98), Japanese Neural Network Society, Kitakyushu, Japan* (pp. 1459–1464). Burke, VA: IOS Press.
- Jagota, A., Narasimhan, G., & Regan, K.W. (1998). Information capacity of binary weights associative memories. *Neurocomputing, 19* (1–3), 35–58.

- Kohonen, T. (1996). *Self-organizing maps* (2nd ed.), New York: Springer-Verlag.
- Koutras, A., Dermatas, E., & Kokkinakis, G. (1999). Blind signal separation and speech recognition in the frequency domain. In *Proceedings of ICECS '99. 6th IEEE International Conference on Electronics, Circuits and Systems, Pafos, Cyprus* (pp. 427–430). Piscataway, NJ: IEEE.
- Lee, T. W., Girolami, M., Bell, A. J., & Sejnowski, T. J. (1999). Independent component analysis using an extended infomax algorithm for mixed subgaussian and supergaussian sources. *Neural Computation*, *11*(2), 417–441.
- Lee, T. W., Girolami, M., Bell, A. J., & Sejnowski, T. J. (2000). A unifying information-theoretic framework for independent component analysis. *Computers & Mathematics with Applications*, *39*(11), 1–21.
- Lee, K., Hon, H., Hwang, M., & Huang, X. (1996). Speech recognition using hidden Markov models: A CMU perspective. In H. Fujisaki (Ed.), *Recent research towards advanced man-machine interface through spoken language* (pp. 249–266). Amsterdam: Elsevier.
- Oja, E., & Karhunen, J. (1995). Signal separation by nonlinear Hebbian learning. In *Computational intelligence—a dynamic system perspective* (pp. 83–97). M. Palaniswami, Y. Attikiouzel, R. Marks, II, D. Fogel, and T. Fukada (Eds.), New York: IEEE Press.
- Palm, G. (1980). On associative memory. *Biological Cybernetics*, *36*(1), 19–31.
- Rauschecker, J. P. (1998). Cortical processing of complex sounds. *Current Opinions in Neurobiology*, *8*(4), 516–521.
- Schwenker, F., Sommer, F. T., & Palm, G. (1996). Iterative retrieval of sparsely coded associative memory patterns. *Neural Networks*, *9*(3), 445–455.
- Steinbuch, K. (1963). *Automat und Mensch* (2nd ed.), New York: Springer-Verlag.
- Van Hulle, M., Yu-Hen, H., Larsen, J., Wilson, E., & Douglas, S. (1999). Clustering approach to square and non-square blind source separation. In *Proceedings of the 1999 Signal Processing Society Workshop: Neural Networks for Signal Processing IX, Madison, Wisconsin* (pp. 315–323). Piscataway, NJ: IEEE Press.
- Wang D. L., & Brown, G. J. (1999). Separation of speech from interfering sounds based on oscillatory correlation. *IEEE Transactions on Neural Networks*, *10*(3), 684–697.
- Willshaw, D., Buneman, O., & Longuet-Higgins, H. (1969). Non-holographic associative memory. *Nature*, *222*, 960–962.
- Wilson, M. A., & McNaughton, B. L. (1993). Dynamics of the hippocampal ensemble code for space. *Science*, *261*, 1055–1058.
- Yen, K. & Zhao, Y. (1997). Co-channel speech separation for robust automatic speech recognition: stability and efficiency. In *IEEE International Conference on Acoustics, Speech, and Signal Processing, Munich, Germany* (Vol. 2, pp. 859–862). Los Alamitos, CA: IEEE Computer Society Press.