CHAPTER 8

Cortical Algorithms

If you just have a single problem to solve, then fine, go ahead and use a neural network. But if you want to do science and understand how to choose architectures, or how to go to a new problem, you have to understand what different architectures can and cannot do.

—Marvin Minsky[1]

Computational models inspired by the structural and functional properties of the human brain have seen impressive gains since the mid-1980s, owing to significant discoveries in neuroscience and advancements in computing technology. Among these models, *cortical algorithms* (CAs) have emerged as a biologically inspired approach, modeled after the human visual cortex, which stores sequences of patterns in an invariant form and recalls those patterns autoassociatively. This chapter details the structure and mathematical formulation of CA then presents a case study of CA generalization accuracy in identifying isolated Arabic speech using an entropy-based weight update.

Cortical Algorithm Primer

Initially developed by Edelman and Mountcastle (1978), and inspired by the visual human cortex, CAs are positioned to be superior to the early generations *of artificial neural networks* (ANNs), which do not use temporal and spatial relationships in data for building *machine learning* models.

The CA model consists of a multilayered network, with the cortical column as the basic structure. The network is trained in a two-stage manner: the first learning stage is unsupervised and trains the columns to identify independent features from the patterns occurring; the second stage relies on supervised feedback learning to create invariant representations.

Cortical Algorithm Structure

The human brain is a six-layered structure consisting of a very large number of neurons strongly connected via feedforward and feedback connections. An important property of the neocortex is its structural and functional uniformity: all units in the network seem similar, and they perform the same basic operation. Like this brain architecture, CA architecture has minicolumns of varying thickness (Edelman and Mountcastle 1978).
A *minicolumn* is a group of neurons that share the same receptive field: neurons belonging to a minicolumn are associated with the same sensory input region. The minicolumn is the basic structure in a cortical network, in contrast to neurons in a classical ANN. An association of minicolumns is called a *hypercolumn* or *layer*

[1]Marvin Minsky, "Scientist on the Set: An Interview with Marvin Minsky," in *HAL's Legacy: 2001's Computer as Dream and Reality*, by David G. Stork (Massachusetts Institute of Technology Press, 1998), p. 18.

(in what follows, the terms *column* and *minicolumn* are used interchangeably). Connections in a CA network occur in two directions: horizontally, between columns in the same layer, and vertically, between columns of consecutive layers. Although connections between nonconsecutive layers are present in the human cortex, these connections are omitted in CA, for the sake of simplicity.

Figure 8-1 displays a representation of a cortical network. The lateral inhibiting connections are not shown explicitly in the figure because their functionality is not physical; that is, these connections do not represent data propagated between neurons, but serve as a means of communication between the columns.

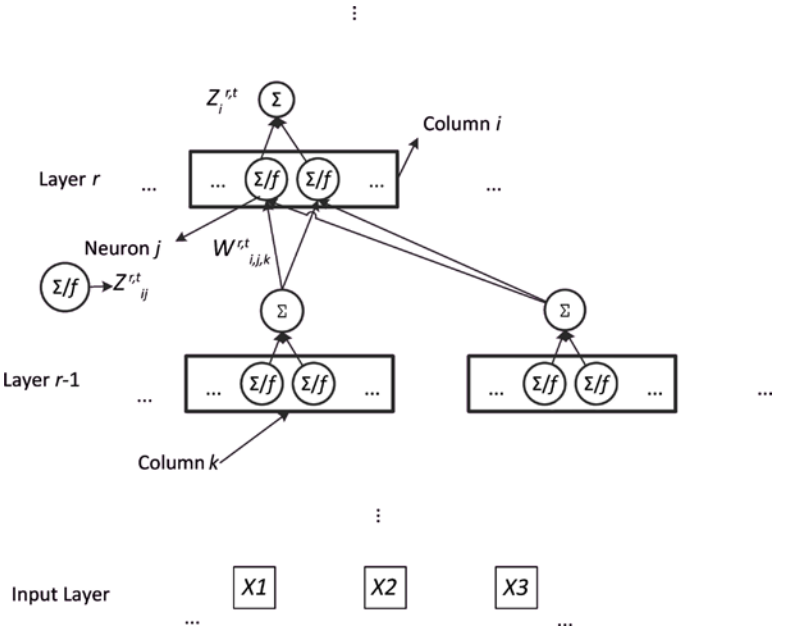


Figure 8-1. *Schematic of cortical network connectivity*

The notation adopted hereafter is given in Figure 8-2, where $W_{i,j,k}^{r,t}$ represents the weight of the connection between the *j*th neuron of the *i*th column of layer *r* and the *k*th column of the previous layer *(r-1)* during the training epoch *t*. **Bold variables** stand for vector entities, underlined variables represent matrices, and *italic variables* represent scalar entities.

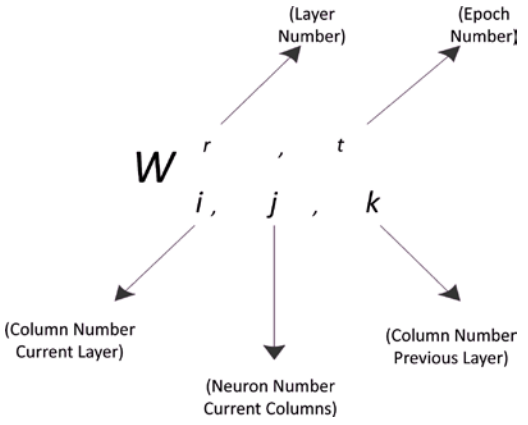


Figure 8-2. *Nomenclature conventions for the weight* $W_{i,j,k}^{r,t}$

During the learning process a connection is disabled by assigning to it a zero weight. If the network is fully connected, each neuron *j* in the column is connected to all the columns in the previous layer. All connections are *elastic*; that is, if a connection is disabled during the feedforward process, it can be restored during the feedback learning, and vice versa.

The *weight matrix* representing the state of a column composed of *M* nodes during the training epoch *t* is defined by

$$\underline{W_i^{r,t}} = \left[\boldsymbol{W}_{i,1}^{r,t} \quad \boldsymbol{W}_{i,2}^{r,t} \quad \cdots \quad \boldsymbol{W}_{i,j}^{r,t} \quad \cdots \quad \boldsymbol{W}_{i,M}^{r,t} \right]. \tag{8-1}$$

The *weight vector* $\boldsymbol{W}_{i,j}^{r,t}$ of the connections entering neuron *j* of column *i* in layer *r*, composed of L_r columns, is given by

$$\boldsymbol{W}_{i,j}^{r,t} = \left[W_{i,j,1}^{r,t} \quad W_{i,j,2}^{r,t} \quad \cdots \quad W_{i,j,k}^{r,t} \quad \cdots \quad W_{i,j,L_{r-1}}^{r,t} \right]', \tag{8-2}$$

where L_{r-1} is the number of columns in the layer *(r-1)*, L_r represents the number of columns in the layer *r,* and the superscript ′ stands for the transpose operator.

Expanding $W_i^{r,t}$ yields

$$\underline{W_i^{r,t}} = \begin{bmatrix} W_{i,1,1}^{r,t} & \cdots & W_{i,j,1}^{r,t} & \cdots & W_{i,M,1}^{r,t} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ W_{i,1,k}^{r,t} & \cdots & W_{i,j,k}^{r,t} & \cdots & W_{i,M,k}^{r,t} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ W_{i,1,L_{r-1}}^{r,t} & \cdots & W_{i,j,L_{r-1}}^{r,t} & \cdots & W_{i,M,L_{r-1}}^{r,t} \end{bmatrix}. \tag{8-3}$$

The output vector $Z^{r,t}$ of layer *r* for epoch *t* is given by

$$\boldsymbol{Z}^{r,t} = \left[Z_1^{r,t}, Z_2^{r,t}, \ldots, Z_i^{r,t}, \ldots, Z_{L_r}^{r.t} \right]', \tag{8-4}$$

where $Z_i^{r,t}$ is the output of column *i* in the layer for the same training epoch.

Considering the output of a neuron to be the result of the nonlinear activation function *f(.)*, in response to the weighted sum of the connections entering the neuron, the output of the column is defined as the sum of the outputs of the neurons constituting the column.

$Z_{i,j}^{r,t}$ is the output of the *j*th neuron of the *i*th column in the *rth* layer at the training epoch *t*, given by

$$Z_i^{r,t} = \sum_{j=1}^{M} Z_{i,j}^{r,t} \; ; \; Z_{i,j}^{r,t} = f\left(\sum_{k=1}^{L_{r-1}} W_{i,j,k}^{r,t} Z_k^{r-1,t} \right). \tag{8-5}$$

$Z_{i,j}^{r}$ is the output of the *j*th neuron constituting the *i*th column of the *r*th layer, and $f\left(\sum_{k=1}^{L_{r-1}} W_{i,j,k}^{r,t} Z_k^{r-1,t} \right)$ is defined by

$$\begin{cases} f\left(\sum_{k=1}^{L_{r-1}} W_{i,j,k}^{r,t} Z_k^{r-1,t} \right) = \dfrac{1}{1+\exp\left\{ \sum_{k=1}^{L_{r-1}} W_{i,j,k}^{r,t} Z_k^{r-1,t} . \left(\varphi\left(\sum_{k=1}^{L_{r-1}} W_{i,j,k}^{r,t} Z_k^{r-1,t} \right) - T \right) \right\}} \\ \varphi\left(\sum_{k=1}^{L_{r-1}} W_{i,j,k}^{r,t} Z_k^{r-1,t} \right) = \begin{cases} -2 \; if \sum_{k=1}^{L_{r-1}} W_{i,j,k}^{r,t} Z_k^{r-1,t} = 1 \\ \sum_{k=1}^{L_{r-1}} W_{i,j,k}^{r,t} Z_k^{r-1,t} \; otherwise, \end{cases} \end{cases} \tag{8-6}$$

where *T* is a tolerance parameter empirically selected and constant for all epochs and columns. It is assumed that all weights are normalized and bounded between –1 and 1.

The nonlinear activation function is analogous to the propagation of the action potential through an axon in the neural system.

Training of Cortical Algorithms

Connectivity within the columns is modeled through the value of the synaptic weights. Initially, there is no specific connectivity between cortical columns. It is assumed that the network is fully connected before training. Also, all synaptic weights are initialized to random values that are very close to 0 to avoid preference to any particular pattern.

The training process, as introduced by Edelman and Mountcastle (1978) and developed further by Hashmi (2010), is described in the following sections, according to its main phases: unsupervised feedforward, supervised feedback, and weight update.

Unsupervised Feedforward

Feedforward learning trains columns to identify features via random firing and repeated exposure. When a pattern is presented, the input is propagated through the network. Each column has a small probability of firing, which means that most of the columns in a particular layer stay inactive. When the random firing of a particular column coincides with a particular input pattern, this activation is enforced. In other words, when activation is enforced, the column firing strengthens its weights, according to the strengthening weight update rule. At the same time, the column firing inhibits neighboring columns in the same layer from firing by weakening the weights, as presented in the inhibiting update rule.

The weight update rules are as follows:

- *Inhibiting*:

$$W_{i,j,k}^{r,t+1} = Z_k^{r-1,t} \cdot \left(W_{i,j,k}^{r,t} - \Omega\left(W_{i,j}^{r,t} \right) \right) \tag{8-7}$$

- *Strengthening*:

$$W_{i,j,k}^{r,t+1} = Z_k^{r-1,t} \cdot \left(W_{i,j,k}^{r,t} + C_{i,j,k}^{r,t} + \rho \cdot \frac{1}{1 + e^{\frac{\left(W_{i,j,k}^{r,t} - T \right)}{\Omega\left(W_i^{r,t} \right)}}} \right), \tag{8-8}$$

where $\Omega(W_{i,j}^{r,t})$ is given by

$$\Omega\left(W_i^{r,t} \right) = \sum_{j=1}^{M} \sum_{k=1}^{L_{r-1}} C_{i,j,k}^{r,t} W_{i,j,k}^{r,t}; C_{i,j,k}^{r,t} = \begin{cases} 1 \ \textit{if} \ W_{i,j,k}^{r,t} > \varepsilon \\ 0 \ \ \textit{otherwise} \end{cases} \tag{8-9}$$

and where ρ is a tuning parameter, and ε is the firing threshold chosen empirically to be constant for all epochs and columns.

With repeated exposure the network learns to extract certain features of the input data, and the columns learn to fire for specific patterns. Layers in the network extract aspects of the input in increasing complexity. Thus, lower layers detect simple features, whereas higher stages learn concepts and more complex abstractions of the data.

Supervised Feedback

Feedforward learning trains columns to identify features of the data, such that the hierarchical network starts to recognize patterns. When the network is exposed to a variation of a pattern that is quite different from the previous one, the top layer of columns that are supposed to fire for that pattern do not, and only some of the columns in the hierarchy may fire, which leads to a misclassification. Through the CA feedback mechanism, the error occurring at the top layer generates a feedback signal that forces the column firing for the original pattern to fire, while inhibiting the column that is firing for the variation. Over multiple exposures the top layer should reach the desired firing scheme (also called *stable activation*). More specifically, designated columns in the top layer learn to fire for a particular pattern. Once the columns start to give a stable activation for pattern variations, the feedback signal is propagated back to the previous layers. Each layer is then trained until a convergence criterion, expressed as an error term in function of the actual output, and a desired output (firing scheme) are reached. The feedback signal is sent to the preceding layers only once the error in the layer concerned converges to a value below a certain, predefined tolerance threshold. The excitatory and inhibiting signals follow the same update rules as for the feedforward learning.

When used for the feedback learning of the network, CA can be summarized by the following steps:

1. Following the feedforward unsupervised batch learning (i.e., after the training data are entirely propagated through the network), a desired output scheme per layer is formed by averaging the column outputs. If $Z_{i_k}^r$ is the output of the *i*th column in the *r*th layer of the network for a certain training instance denoted by *k* and given *N* instances in total; the desired output for this particular column $Z_{i_d}^r$ is given by:

$$Z_{id}^r = avg\left(Z_{i_k}^r \right) = \frac{1}{N} \sum_{k=1}^{N} Z_{i_k}^r . \tag{8-10}$$

2. Starting with the last layer, compare the measured output of each column as a response to each instance *k*, $Z_i^{r,t}$ with the desired value of $Z_{i_d}^r$. If the desired output of a column is a firing state, whereas the actual is different, the column is strengthened (see Equation 8-8; the column is inhibited (see Equation 8-7) if the opposite occurs (i.e., if the actual output is firing, whereas a nonfiring state is desired).
3. Repeat step 2 until the error threshold is met.
4. Follow the same procedure for the previous layers, one layer at a time.

Weight Update

In CA, good accuracy is taxed with computationally expensive and lengthy training. This cost is mainly due to the computation of the exponential function invoked during the weight update process for each neuron while the weights of the network are learned.

For a particular node $W_{i,j,k}^{r,t}$, Equation 8-8 may be written as:

$$\begin{cases} W_{i,j,k}^{r,t+1} = \alpha W_{i,j,k}^{r,t} + \theta\left(W_{i,j,k}^{r,t}\right) \\ \theta\left(W_{i,j,k}^{r,t}\right) = \beta . \dfrac{1}{1+\exp\left[\dfrac{W_{i,j,k}^{r,t} - T}{\Omega}\right]} + \delta . \\ \alpha = Z_k^{r-1,t}, \beta = Z_k^{r-1,t}\rho \\ \delta = Z_k^{r-1,t} . C_{i,j,k}^{r,t} \end{cases} \tag{8-11}$$

Here, α, β, and δ are variables that depend on the training epoch as well as the column considered; therefore, a suitable nomenclature would be in the form $\chi_k^{r-1,t}$. For the sake of simplicity, one can omit the subscripts and superscripts for these variables, referring to $\Omega(W_i^{r,t})$ as Ω.

As demonstrated in Equation 8-11, the parameters of the exponential weight update rule—α, β, δ, Ω, and T—depend on the state of the column considered. Therefore, it can be inferred that the strengthening rule is a family of exponential functions with varying parameters for each column. The update of a column requires the computation of the exponential function for each of the nodes—hence, the lengthy training.

Figure 8-3 shows a plot of θ, with respect to the value of the neuron weight for a random node.

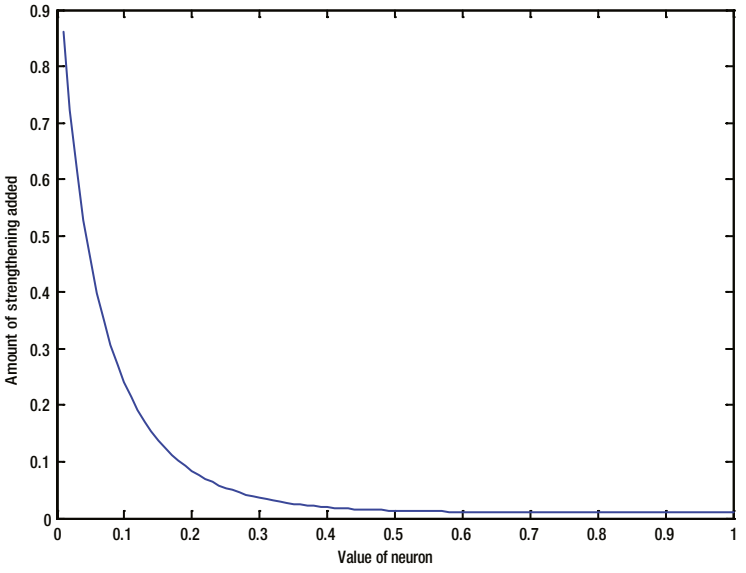


Figure 8-3. *Plot of* $\theta\left(W_{i,j,k}^{r,t}\right)$ *versus* $W_{i,j,k}^{r,t}$

The computational cost involved in the strengthening rule also comes from the calculation of the exponential function. For example, MATLAB software uses the binomial theorem (see Equation 8-12) to compute the approximate value of an exponential, and this approximation is computed up to orders ranging from 5 to 10 (Mohler 2011):

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots + \frac{x^n}{n!} + \cdots = \sum_{i=0}^{\infty} \frac{x^i}{i!}. \tag{8-12}$$

The number of operations required to compute the exponential function is summarized in Table 8-1.

Table 8-1. *Required Operations for Exponential Function*

Expression	Operations	Total Number of Operations
$i!$	$2*3*\ldots*i$	i-two multiplications
x^i	$x*x*x*\ldots$	i multiplications
$\frac{x^i}{i!}$	$\frac{x*x*x\ldots}{2*3*\ldots*i}$	$2i$-one operation
e^x	$\sum_{i=0}^{n} \frac{x^i}{i!}$	$\sum_{i=0}^{n}(2i-1)+n = n^2+n$

The workflow for CA training is displayed in Figure 8-4.

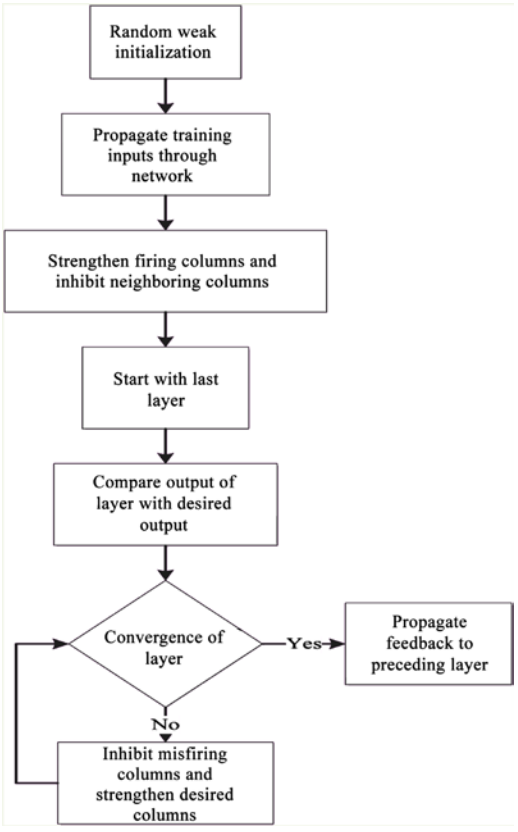


Figure 8-4. *Training of a cortical network*

Experimental Results

Experimental results for various pattern recognition databases obtained from the University of California, Irvine, Machine Learning Repository (Bache and Lichman 2013) show CA superior performance, as detailed for the following datasets:

- *Letter Recognition dataset*: This dataset consists of a collection of 20,000 black-and-white images to be classified as one of the 26 capital letters of the English alphabet (Slate 1991). Each instance is represented by a set of 16 features of integer type, normalized into a range of 0-15 representing aspects of the image, such as horizontal and vertical position, and width and length. The best accuracy reported for this dataset is 97.58 percent (Bagirov and Ugon 2011).

- *Image Segmentation dataset*: This dataset is collection of images sized 3 × 3 each, represented by 19 attributes describing features of the image, such as average intensity, saturation, and hue (Vision Group 1990). The dataset is divided into a training set consisting of 210 instances and a testing set of 2,100 instances; each image belongs to one of 7 classes. Dash et al. (2003) achieved an accuracy of 98.6 percent.
- *ISOLET (Isolated Letter Speech Recognition) dataset*: The task in this experiment is to classify a collection of isolated spoken English letters as one of 26 classes (A-Z). The dataset is composed of 2,800 instances uttered by 150 speakers, each instance represented by a set of 617 features, including spectral coefficients, contour features, sonorant features, presonorant features, and postsonorant features (Cole and Fanty 1994). The reported accuracy of this database is 96.73 percent (Dietterich 1994).
- *PENDIGITS (Pen-Based Recognition of Handwritten Digits) dataset*: This experiment consists of pen-based recognition of handwritten digits. The database collects 10,992 samples from 44 writers, each sample being a sequence of (x, y) coordinates representing the trajectory of the pen during the writing process. The sequences have been resampled to obtain a fixed-length attribute vector equal to 16 (eight pairs of (x, y) coordinates) and normalized to eliminate the effect of artifacts resulting from different handwritings. The 10,992 samples are divided into a training set of 7,494 instances and 3,498 instances for testing. The accuracy of this dataset reached 98.6 percent (Alpaydin and Alimoglu 1998).
- *Multiple Features dataset*: This dataset consists of 649 features, for a total of 2,000 patterns of handwritten numerals (`0'--`9') extracted from a collection of Dutch utility maps (Duin 2013). These digits are represented in terms of six feature sets: 76 Fourier coefficients of the character shapes; 216 profile correlations; 64 Karhunen-Loève coefficients; 240 pixel averages, in 2 × 3 windows; 47 Zernike moments; and six morphological features. The best accuracy achieved is 98 percent (Perkins and Theiler 2003).
- *Abalone dataset*: The task for this dataset is to classify the age of a collection of 4,177 abalones from a total of eight physical measurements, such as height, weight, diameter, and length. This dataset is characterized by a highly unbalanced class distribution and has achieved an accuracy of 79.0 percent (Tan and Dowe 2003).

Table 8-2 compiles the recognition rate, training time, and total number of required iterations for convergence, based on a fourfold cross-validation, using the mean squared error (MSE) and the well-formed *cross-entropy* (CE) cost functions at the output layer.

Two experiments were performed:

- *Experiment 1*: CA with the exponential weight update rule and MSE as a cost function
- *Experiment 2*: CA with the exponential weight rule and CE as a cost function

Table 8-2. *Experimental Results*

Dataset	Measure	Experiment 1	Experiment 2
Letter Recognition	% Accuracy	98.3	98.8
	Training time (min)	223	235
	Number of epochs	237	225
	Number of operations	$8.9 * 10^{12}$	$1.1 * 10^{13}$
Image Segmentation	% Accuracy	99.3	99.7
	Training time (min)	45	52
	Number of epochs	77	69
	Number of operations	$22 * 10^{12}$	$2.5 * 10^{12}$
ISOLET	% Accuracy	98.1	98.7
	Training time (min)	54	67
	Number of epochs	147	131
	Number of operations	$2.6 * 10^{12}$	$3.2 * 10^{12}$
PENDIGITS	% Accuracy	99.8	100
	Training time (min)	135	154
	Number of epochs	94	82
	Number of operations	$6.5 * 10^{12}$	$7.4 * 10^{12}$
Multiple Features	% Accuracy	98.7	99.1
	Training time (min)	35	42
	Number of epochs	66	53
	Number of operations	$1.4 * 10^{12}$	$2.0 * 10^{12}$
Abalone	% Accuracy	91.8	92.2
	Training time (min)	56	68
	Number of epochs	70	62

On average the CE cost function results in better classification accuracy. However, this is achieved at the expense of an increase in computational complexity and training time.

■ **Note** Despite their superior hypothetical performance, CAs remain less widely used than ANNs, owing to their longer and more expensive training and computational requirements. These make them unattractive for online learning, energy-aware computing nodes, and large datasets with stringent restrictions on the training duration.

Modified Cortical Algorithms Applied to Arabic Spoken Digits: Case Study

Because CAs have not been extensively implemented for *automatic speech recognition* (in particular for the Arabic language), the following sections show how CA strengthening and inhibiting rules originally employed during feedback were modified with weighted entropy concepts that were added to the CA cost function and the weight update rule.

Entropy-Based Weight Update Rule

During the feedback learning stage of a CA, the output of each layer is compared with a desired state of firing, and the weights are updated until an error term is reduced to a minimum threshold value. Using the least squares criterion, large error values influence the learning process much more than smaller ones. For a class of problems, the gradient descent algorithm, with the MSE as a criterion for weight updates, can be trapped in a local minimum and so it will fail to find the optimal solution. In contrast, the well-formed CE criterion, employing a gradient descent algorithm, guarantees convergence to the optimal solution during learning (Wittner and Denker 1988).

The three properties of a well-formed error function of the form $J\left(W_{i,j,k}^{r,t}\right)=\sum h(W_{i,j,k}^{r,t})$ are as follows:

- For all $W_{i,j,k}^{r,t}$ the derivative of $h(W_{i,j,k}^{r,t})$, defined as $h^{'}\left(W_{i,j,k}^{r,t}\right)$, must be negative.
- There must exist an $\epsilon > 0$, such that $-h^{'}\left(W_{i,j,k}^{r,t}\right)\geq\epsilon$ for all $W_{i,j,k}^{r,t}\leq 0$.
- The function *h* must be differentiable and bounded.

CE as a cost function criterion can be written as

$$J^{r,t}=\sum_{i=1}^{L_r} Z_{di}^{\ r}\ln\frac{Z_i^{r,t}}{Z_{di}^{\ r}},$$

where $Z_{di}^{\ r}$ is the desired output of the *i*th column of layer *r* at epoch *t*.

If you adopt the same procedure for the feedback learning, and assume that training convergence of each layer happens when the entropy measure falls below a predetermined threshold value, the weight update rule becomes:

- *Inhibiting*:

$$W_{i,j,k}^{r,t+1} = \frac{\Delta J^{r,t}}{\Delta Z_i^{r,t}} . Z_k^{r-1,t} \left(W_{i,j,k}^{r,t} - \Omega\left(W_i^{r,t} \right) \right) \tag{8-13}$$

- *Strengthening*:

$$W_{i,j,k}^{r,t+1} = \frac{\Delta J^{r,t}}{\Delta Z_i^{r,t}} . Z_k^{r-1,t} . \left(W_{i,j,k}^{r,t} + C_{i,j,k}^{r,t} + \rho . \frac{1}{1 + \exp\left[\frac{W_{i,j,k}^{r,t} - T}{\Omega\left(W_i^{r,t} \right)} \right]} \right) \tag{8-14}$$

One advantage of using the proposed gradient descent weighted rules is that the CE cost function diverges if one of the outputs converges to the wrong extreme; hence, the gradient descent reacts quickly. In contrast, the MSE cost function approaches a constant, and the gradient descent on the least square will wander on a plateau, even though the error may not be small.

Experimental Validation

Using a CA of six hidden layers, starting with 2,000 columns of 20 nodes for the first hidden layer and decreasing the number of columns by half between consecutive layers, four experiments, employing the weighted entropy weight update rule, were performed based on a fivefold cross-validation:

- *Experiment 1*: CA trained using the MSE cost function and the original weight update rule
- *Experiment 2*: CA trained using the MSE cost function and the proposed weight update rule
- *Experiment 3*: CA trained using the CE cost function and the original weight update rule
- *Experiment 4*: CA trained using the CE function and the proposed weight update rule

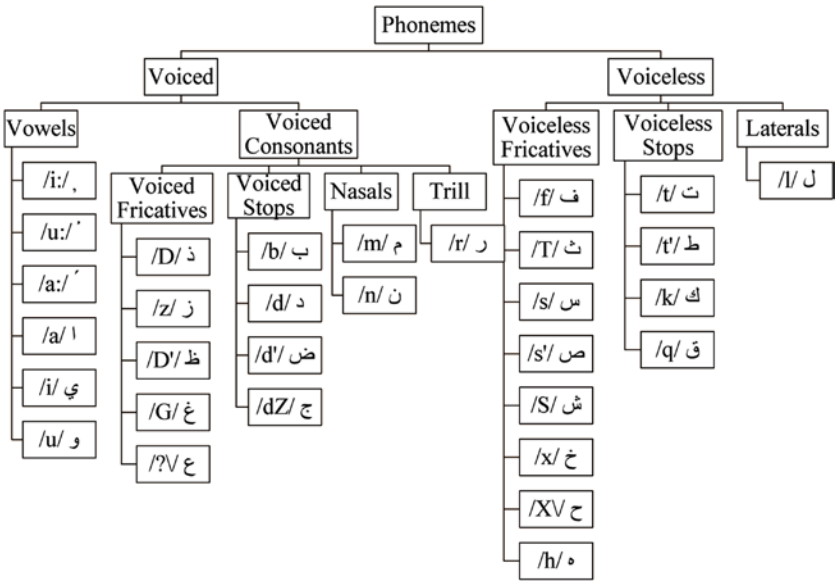
Simulations were executed, using MATLAB R2011a software on an Intel i7 at 2GHz and 6GB RAM on a Windows 7 Home Premium operating system, using a modified central nervous system (CNS) library. Developed at the Massachusetts Institute of Technology, by Mutch, Knoblich, and Poggio (2010), the CNS library is a framework for simulating cortically organized networks.

The database was obtained from the UCI Machine Learning Repository and consists of a collection of 13 Mel frequency cepstral coefficient (MFCC) frames representing 8,800 spoken Arabic digits—one of ten classes (0-9), uttered by 88 different speakers, obtained after filtering the spoken digits, using a moving Hamming window. Several techniques were validated on this database; the best achieved result shows a 97.03 percent recognition rate, based on a threefold cross-validation, using a multiclass SVM classifier (Ji and Sun 2011).

TREE REPRESENTATION FOR ARABIC PHONEMES

As the first language in 22 countries, Arabic ranks fifth among the most spoken languages in the world (Mosa and Ali 2009). Although applications treating speech recognition have increased significantly (e.g., iPhone 4S Siri interface), implementation for the Arabic language is limited, mainly because of its morphological complexity. For Arabic automatic speech recognition, the recognition of phonemes constitutes an important step in continuous speech analysis. Most research proceeds by extracting isolated phonemes or small phonetic segments (El-Obaid, Al-Nassiri, and Maaly 2006; Awais 2003; Gevaert, Tsenov, and Mladenov 2010; Al-Manie, Alkanhal, and Al-Ghamdi 2009) for analysis of longer speech signals (Abushariah et al. 2010) and broadcast news (Al-Manie, Alkanhal, and Al-Ghamdi 2009), using several techniques, such as ANN (Essa, Tolba, and Elmougy 2008), fuzzy HMM (Shenouda, Zaki, and Goneid 2006), fuzzy logic, concurrent self-organizing maps (Sehgal, Gondal, and Dooley 2004), and HMM (Satori, Harti, and Chenfour 2007; Bourouba et al. 2010; Biadsy, Moreno, and Jansche 2012).

Spoken in the Middle East and North Africa, Arabic has different dialects. However, Literary Arabic (also called Modern Standard Arabic) is the official form used in documents and for formal speaking in all Arabic-speaking countries. One of the differences between spoken and written Arabic is the presence in the latter of diacritics (marks used to indicate how a letter should be pronounced). The complexity of Arabic is the result of its unusual morphology: words are formed using a root-and-pattern scheme, in which the root is composed of 3 consonants, leading to several possibilities from one root. Phonetically, Arabic has 28 consonant segments and 6 vowels (Newman 1984). Phonemes can be grouped according to the articulation of the lips and tongue during speech, as shown in the classification of Arabic phonemes.



■ **Note** MFCCs can model the acoustic content of speech independently of the source (speaker). MFCCs are calculated by mapping the logarithm of the spectrum into the Mel scale and converting the obtained signal back to the time domain, using discrete cosine transform (Klatau 2005).

For consistency, the first experiment with the data used the 13 MFCCs provided and then added the first and second derivatives of the MFCCs, that is, coefficients with a feature vector of size 39. Tables 8-3 and 8-4 show a comparison of the results obtained for all experiments with the average recognition rate obtained, training time, and number of epochs required for convergence.

Table 8-3. *Results for the Spoken Arabic Digit Dataset, Using 13 MFCCs*

	Experiment 1	Experiment 2	Experiment 3	Experiment 4
Recognition rate (%)	97.4	98.4	97.9	99.0
Training time (min)	90	110	100	115
Number of epochs until convergence	240	232	235	220

Table 8-4. *Results for the Arabic Spoken Digit Dataset, Using 39 MFCCs*

	Experiment 1	Experiment 2	Experiment 3	Experiment 4
Recognition rate (%)	98.2	99.1	98.6	99.7
Training time (min)	125	142	137	156
Number of epochs until convergence	335	298	322	280

Tables 8-3 and 8-4 demonstrate that training of the cortical network, using the entropy cost function and the proposed weight update rule, performed better than the original training parameters. This improvement is achieved at the expense of a small worsening of the required training time. Despite the lengthy training time, however, the proposed weight update rule requires fewer training epochs to converge, compared with the original weight update rule. This is because the amount of strengthening added using the proposed rule is proportional to the gradient of the cost function, meaning that fewer training epochs are necessary to reach convergence. The proposed weight update rule involves computing the entropy gradient, which is computationally more expensive, compared with the original weight update rule.

The confusion matrices in Figure 8-5, obtained for the image segmentation dataset using both cost functions, demonstrate that although a significant trend is observed in the confusion between classes 1, 7, and 8 with the classical distance measure, the proposed entropy-based update rule was able to correct this trend partially.

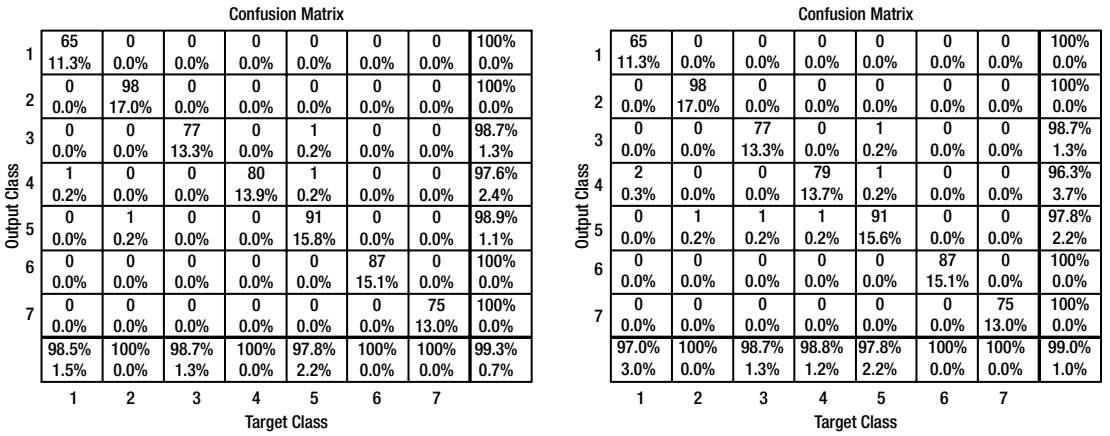


Figure 8-5. *Confusion matrices for the Image Segmentation dataset: left, exponential rule; right, linear rule*

Figure 8-6 compares the CE cost function with the training epochs obtained while training the cortical network using the entropy cost function for the proposed and the regular weight update rules. Note that the proposed weight update converges to a smaller MSE value, compared with the regular update, which is consistent with the recognition rates obtained earlier.

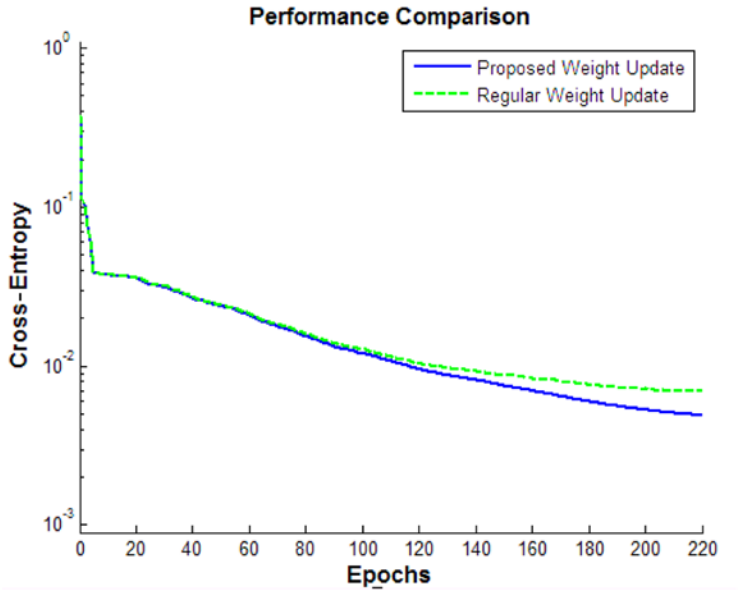


Figure 8-6. *Entropy cost function comparison for regular and proposed weight update rules*

References

Abushariah, Mohammad A. M., Raja N. Ainon, Roziati Zainuddin, Moustafa Elshafei, and Othman O. Khalifa. "Natural Speaker-Independent Arabic Speech Recognition System Based on Hidden Markov Models Using Sphinx Tools." In *Proceedings of the 2010 International Conference on Computer and Communication Engineering*, Kuala Lumpur, Malaysia, May 11–12, 2010, 1–6. Piscataway, NJ: Institute of Electrical and Electronic Engineers, 2010.

Awais, M. M. "Recognition of Arabic Phonemes Using Fuzzy Rule Base System." In *Proceedings of the 7th International Multitopic Conference*, Islamabad, Pakistan, December 8–9, 2003, 367–370. Piscataway, NJ: Institute of Electrical and Electronic Engineers, 2003.

Bache, K., and M. Lichman. "University of California, Irvine, Machine Learning Repository." Irvine: University of California, 2013. http://archive.ics.uci.edu/ml/index.html.

Bagirov, A. M., J. Ugon, and D. Webb. "An Efficient Algorithm for the Incremental Construction of a Piecewise Linear Classifier." *Journal of Information Systems* 36, no. 4 (2011): 782–790.

Biadsy, Fadi, Pedro J. Moreno, and Martin Jansche. "Google's Cross-Dialect Arabic Voice Search." In *Proceedings of the 2012 IEEE International Conference on Acoustics, Speech, and Signal Processing, Kyoto, Japan, March 25–30, 2012,* 4441–4444. Piscataway, NJ: Institute of Electrical and Electronic Engineering, 2012.

Bourouba, H., R. Djemili, M. Bedda, and C. Snani. "New Hybrid System (Supervised Classifier/HMM) for Isolated Arabic Speech Recognition." In *Proceedings of the 2nd Conference on Information and Communication Technologies,* Damascus, Syria, April 24–28, 2006, 1264–1269. Piscataway, NJ: Institute of Electrical and Electronic Engineering, 2006.

Cole, Ron, and Mark Fanty. "ISOLET Data Set." University of California, Irvine, Machine Learning Repository. Irvine: University of California, 1994. https://archive.ics.uci.edu/ml/datasets/ISOLET.

Dash, Manoranjan, Huan Liu, Peter Scheuermann, and Kian Lee Tan. "Fast Hierarchical Clustering and Its Validation." *Data and Knowledge Engineering* 44, no. 1 (2003): 109–138.

Dietterich, Thomas G., and Ghulum Bakiri. "Solving Multiclass Learning Problems via Error-Correcting Output Codes." *Journal of Artificial Intelligence Research* 2, no. 1 (1995): 263–286.

Duin, Robert P. W. "Multiple Features Data Set." University of California, Irvine, Machine Learning Repository. Irvine: University of California, 2013. http://archive.ics.uci.edu/ml/datasets/Multiple+Features.

Edelman, Gerald M., and Vernon B. Mountcastle. *The Mindful Brain: Cortical Organization and the Group-Selective Theory of Higher Brain Function*. Cambridge, MA: Massachusetts Institute of Technology Press, 1978.

Essa, E. M., A. S. Tolba, and S. Elmougy. "A Comparison of Combined Classifier Architectures for Arabic Speech Recognition." In *Proceedings of the 2008 International Conference on Computer Engineering and Systems*, Cairo, Egypt, November 25–27, 2008, 149–153. Piscataway, NJ: Institute of Electrical and Electronic Engineering, 2008.

Gevaert, Wouter, Georgi Tsenov, and Valeri Mladenov. "Neural Networks Used for Speech Recognition." *Journal of Automatic Control* 20, no. 1 (2010): 1–7.

Ji, You, and Shiliang Sun. "Multitask Multiclass Support Vector Machines." In *Proceedings of the 11th International Conference on Data Mining Workshops*), Vancouver, BC, December 11, 2011, 512–518. Piscataway, NJ: Institute of Electrical and Electronic Engineering, 2011.

Klautau, Aldebaro. "The MFCC," 2012. www.cic.unb.br/~lamar/te073/Aulas/mfcc.pdf.

Hashmi, Artif G., and Mikko. H. Lipasti. "Discovering Cortical Algorithms". In *Proceedings of the International Conference on Fuzzy Computation and International Conference on Neural Computation*, Valencia, Spain, October 24–26, 2010, 196–204.

Manie, Mohammed A. Al-, Mohammed I. Alkanhal, and Mansour M. Al-Ghamdi. "Automatic Speech Segmentation Using the Arabic Phonetic Database." In *Proceedings of the 10th WSEAS International Conference on Automation and Information*, Prague, Czech Republic, March 23–25, 76–79. Stevens Point, Wisconsin: World Scientific and Engineering Academy and Society, 2009.

Mohler, Cleve. "Exponential Function." Chap. 8 in *Experiments with MATLAB*. MathWorks, 2011. www.mathworks.com/moler/exm/chapters/exponential.pdf.

Mosa, Ghassaq S., and Abduladhem Abdulkareem Ali. "Arabic Phoneme Recognition Using Hierarchical Neural Fuzzy Petri Net and LPC Feature Extraction." *Signal Processing: An International Journal* 3, no. 5 (2009): 161–171.

Mutch, Jim, Ulf Knoblich, and Tomaso Poggio. "CNS: A GPU-Based Framework for Simulating Cortically-Organized Networks." Technical Report, Massachusetts Institute of Technology, 2010.

Newman, Daniel. "The Phonetics of Arabic." *Journal of the American Oriental Society* 46 (1984): 1–6.

Obaid, Manal El-, Amer Al-Nassiri, and Iman Abuel Maaly. "Arabic Phoneme Recognition Using Neural Networks." In *Proceedings of the 5th WSEAS International Conference on Signal Processing*, Istanbul, Turkey, May 27–29, 2006, 99–104. Stevens Point, Wisconsin: World Scientific and Engineering Academy and Society, 2006.

Perkins, Simon, and James Theiler. "Online Feature Selection Using Grafting." In *Proceedings of the Twentieth International Conference on Machine Learning, Washington, DC, August 21–24, 2003,* 592–599. Menlo Park, CA: Association for the Advancement of Artificial Intelligence, 2003.

Satori, Hassan, Mostafa Harti, and Nouredine Chenfour. "Introduction to Arabic Speech Recognition Using CMU Sphinx System." In *Proceedings of the Information and Communication Technologies International Symposium*, Fez, Morocco, April 3–5, 2007, edited by Mohammad Essaaidi, Mohammed El Mohajir, Badreddine El Mohajir, and Paolo Rosso, 139–142. Piscataway, NJ: Institute of Electrical and Electronic Engineers, 2007.

Sehgal, M. S. B., Iqbal Gondal, and Laurence Dooley. "A Hybrid Neural Network Based Speech Recognition System for Pervasive Environments." In *Proceedings of the 8th International Multitopic Conference,* Lahore, Pakistan, December 24–26, 2004, 309–314. Piscataway, NJ: Institute of Electrical and Electronic Engineers, 2004.

Shenouda, Sinout D., Fayez W. Zaki, and A. M. R. Goneid. "Hybrid Fuzzy HMM System for Arabic Connectionist Speech Recognition." In *Proceedings of the Twenty-Third National Radio Science Conference, Monufia, Egypt, March 14–16,* 1–8. Piscataway, NJ: Institute of Electrical and Electronic Engineers, 2006.

Slate, David J. "Letter Recognition Data Set." University of California, Irvine, Machine Learning Repository. Irvine: University of California, 1991. http://archive.ics.uci.edu/ml/datasets/Letter+Recognition.

Tan, Peter J., and David L. Dowe. "MML Inference of Decision Graphs with Multi-Way Joins and Dynamic Attributes." In *AI 2003: Advances in Artificial Intelligence; Proceedings of the 16th Australian Conference on AI, Perth, Australia, December 3–5, 2003,* edited by Tamás Domonkos Gedeon and Lance Chun Che Fung, 269–281. Berlin: Springer, 2003.

Alpaydin, E., and Fevzi Alimoglu. "Pen-Based Recognition of Handwritten Digits Data Set." University of California, Irvine, Machine Learning Repository. Irvine: University of California, 1998. https://archive.ics.uci.edu/ml/datasets/Pen-Based+Recognition+of+Handwritten+Digits.

Vision Group. "Image Segmentation Data Set." University of California, Irvine, Machine Learning Repository. Irvine: University of California, 1990. https://archive.ics.uci.edu/ml/datasets/Image+Segmentation.

Wittner, Ben S., and John S. Denker. "Strategies for Teaching Layered Networks Classification Tasks." In *Neural Information Processing Systems: Denver, CO, 1987*, edited by Dana Z. Anderson, 850–859. Berlin: Springer, 1988.