COGNIMEM™
Technologies, Inc.

CogniMem Teaching Basics
TECHNICAL BRIEF

Teaching the neurons of the CogniMem chip is very simple thanks to their built-in model generator. The operation consists of broadcasting to the neurons a list a reference vectors along with their associated category and let the neurons decide autonomously which of the vectors to keep in memory because they represent novelty.

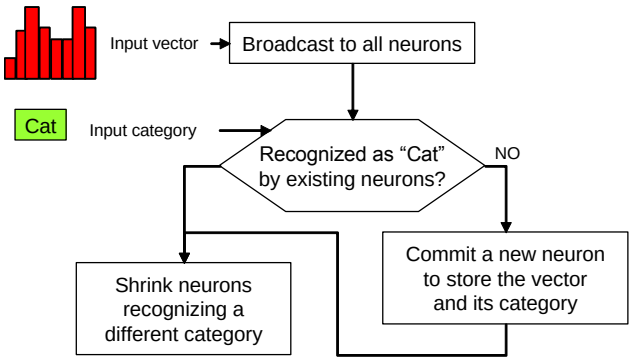
The purpose of this Technical Brief is to demonstrate how easy it is to teach the neurons from a developer's standpoint, but you are welcome to refer to the CogniMem Decision Space Mapping Manual and CogniMem Reference Guide (see Reference Tab on Cognimem's website) for more information about the knowledge building concept and methodology, the selection and generation of the reference vectors, the usage of multiple neural networks to address applications with high cost of a mistake, and more.

A SIMPLE LEARNING MECHANISM

Let's take the example of Vector #1 composed of parameters profiling a *Good Payer* and Vector #2 composed of the same parameters but profiling a *Bad Payer*. Broadcasting Vector#1 to the neurons with a category Good Payer will commit the first neuron to store Vector#1 in its memory and Good Payer in its category register. The same operation with Vector#2 and the category Bad Payer will commit a second neuron to store Vector#2 and the category Bad Payer. This is the first foundation of a knowledge intended to discriminate good and bad payers. Obviously, additional vectors must be broadcasted to enrich and tune this knowledge, but in any case it is the neurons which will decide which vectors to store in their memory as significant prototypes and which ones to discard because they do not represent new information.

When a new example of category A is presented for learning, the neural network first attempts to recognize it.

- If the example is not recognized by any existing neurons, a new neuron is automatically added to the network in order to store the new example and its category value A.
- If the example is recognized by one or more neurons and they all agree that it matches category A, then the new example is discarded since it does not add any new information to the existing knowledge base.
- If the example is recognized by several neurons where one or more identify it with a category other than A, these neurons which are in disagreement with the category to learn automatically reduce their Influence field to exclude the new example. This corrective action changes the knowledge base by making certain neurons more conservative in their classification process.



As a result, a learning operation can have the following impact on a knowledge base:

- Add a new neuron
- Reduce the Influence field of existing neurons
- Reduce the Influence field of existing neurons and add a new neuron
- Do nothing

It is important to realize that when the influence fields of neurons are reduced, it might very well happen that an example which was recognized with the correct category at an earlier time is no longer recognized as such because the neuron which originally recognized the said example now excludes it. Therefore repeating the learning of all examples until the number of neurons reaches a constant is a good method to build a robust knowledge.

The learning curve describing the rate at which neurons are committed when examples are taught should be asymptotic. At the beginning, you may get one neuron committed per example, but this should slow down rapidly and reach an asymptote if the vectors are representative and discriminating features of a population. Indeed we want the neurons to be able to generalize which means recognizing vectors never seen before but similar to learned examples.

For a given set of examples, you can generate multiple knowledge bases using different region size. A knowledge with fewer (and bigger) neurons has better generalization capabilities, but can generate classification with uncertainties if not mistakes. A knowledge with more neurons (and smaller) has a more conservative behavior, will make less mistakes, but will classify new situations as unknown rather than associating them to known examples.

METHODOLOGY TO VALIDATE A KNOWLEDGE

1) RECOGNIZE WHAT YOU HAVE TAUGHT

A first validation of a knowledge consists of recognizing the taught examples. A majority of them should be positively identified with the correct category and a minority should be classified with an uncertainty between the correct category and other possible categories. The percentage of uncertain classifications can be an indication that the selected vectors or the feature they represent are not necessarily the best.

2) RECOGNIZE NEW EXAMPLES

The true accuracy of a knowledge is rated by recognizing examples never seen before. It is realistic to expect false classifications at first, but CogniMem can easily correct these inaccuracies by learning new examples. Multiple iterations between learning and validation might be necessary to obtain a robust knowledge. An incorrect classification is caused by one of the following: an insufficient training, an erroneous or misleading training, use of an improper or irrelevant feature vector.

DO I NEED ONE NEURON PER EXAMPLE I TEACH?

No. A new neuron is committed only if the neural network decides that the example adds information to the existing knowledge. This is determined by verifying that the new example is not already recognized and classified with the correct category by the existing committed neurons.

A good learning curve should be exponential. At the beginning of a teaching session, neurons are committed frequently. After a while their number should become steady meaning that the knowledge is robust and recognizes the majority of the examples.

CM1K SPECIFICS

Teaching a vector of N components if made with N+1 Write commands is described in the following table:

Description	RTL commands	Timings
Broadcast the N-1 components	For (i = 0; i<N-1; i++) Write CM_COMP, Vector(i);	N-1 clock cycles
Broadcast the last component (which updates the ID, UNC flags)	Write CM_LCOMP, Vector(N)	3 clock cycles
Broadcast the category of the vector	Write CM_CAT, value	1 clock cycles if ID 19 otherwise