5/7-61

N96- 12930

p. 8

CLIPS on the NeXT Computer

Elizabeth Charnock & Norman Eng
Pacific Microelectronics, Inc.
201 San Antonio Circle C250
Mountain View CA 94040

Abstract

This paper discusses the integration of CLIPS into a hybrid expert system-neural network AI tool for the NeXT computer. The main discussion is devoted to the joining of these two AI paradigms in a mutually beneficial relationship. We conclude that expert systems and neural networks should not be considered as competing AI implementation methods, but rather as *complimentary* components of a whole.

I. Introduction to NeuExpert

NeuExpert is the name of our system, which is the basis of this paper. NeuExpert was designed for the NeXT compu'er. NeXT was an ideal candidate for this type of development since it runs under Unix, and has an object-oriented programming environment as well as a nice large high-resolution monitor. The intent behind the design of NeuExpert was to make AI as accessible to the end user as possible, and, in particular, to remove some of the stigma associated with neural networks. The incorporation of neural networks was necessary since although neural networks are certainly not the answer to every problem, they *do* represent the resolution of most of the usual complaints against expert systems, namely a sometimes crippling lack of adaptability and flexibility.

The first natural combination of the two methodologies that comes to mind is a partitioning the knowledge space into areas owned by the expert system and areas owned by different neural networks, (neural *units* for short.) Since the granularity of the expert system is much less than that the neural network, this partitioning could be easily achieved by partitioning the knowledge space by *scale*. This can be conceptualized by considering the interval between 0 and 1, from which an arbitrarily large set of rational numbers can be extracted, nevertheless leaving behind an infinite amount of space occupied by the irrational numbers in that interval. Thus the first type of interaction that must occur between expert system and neural network is some arrangement regarding the ownership of different parts of the knowledge space.

II. Basic Integration Strategy

Starting from the previous observation, there are two possible ways to proceed: making the expert system and the neural networks compete for territory as two distinct species competing for turf, or creating an absolutely cooperative relationship in which both expert system and neural network would function as two organs in one body which perform different, but related functions.

We opted for the latter course because most actual knowledge in the brain seems to consist both of highly formalized components *and* completely unformalized hunches. Pursuing this line of thought seems to yield the following "common sense" model which classifies knowledge into three distinct categories:

1. knowledge which is of a *strictly procedural* nature, either because of a lack of real comprehension of what underlies the procedures, because of lack of any experience applying the knowledge, or because there *is* nothing at all underneath. For example, if I were to memorize bus schedules to various locations, the knowledge which I possess could be considered strictly procedural.
2. knowledge which can be considered largely *intuitive*; knowledge derived from extensive experience, or which only exists in the form of triggering associations between items in memory.
3. knowledge which lies in one of the first two categories, but is gravitating towards *joint membership*. This could occur because highly formalized knowledge from the procedural category has become endowed with the added knowledge acquired from experience, or because ideas that began as vague or indistinct associations have evolved into a more formalized representation.

This model clearly suggests a cooperative, rather than a competitive relationship between expert system and neural network. Based on this model, the following interactions between expert system and neural network were created:

a. neural network allocation for a specific rule node by the expert system based on rule usage. The neural network, or neural unit, can be considered "clamped" to that node which we will refer to hereafter as the "parent" node.
b. neural network "feeding," or "starvation," based upon rule usage in the expert system
c. neural network migrations to nodes having a very high positive or negative correlation with the parent node. Neural units which migrate in this way can be thought of as *associators*.
d. neural network migrations due to patterns of rule firings elsewhere in the system which are similar to patterns occurring in a group involving the parent node. Neural units performing this type of migration can be thought of as *concept generators*, since their task is to locate structural*[1] similarities in the information.
e. strong migrating units can actually cause the expert system to leapfrog from its appointed path, or "freely" associate, if permitted to do so by the user.

By *migration*, we mean that the attracting node becomes "close" enough to the parent node from the neural unit's perspective that the firing of the attracting node can cause activity in the neural unit. Conversely, the firing of the parent node can cause the attracting node to fire as a system "afterthought," which is displayed separately to the user.

These interactions allow for information in the third category to be appropriately hybridized. They create a true symbiotic relationship between expert system and neural network. However, the first category of knowledge clearly is a straight expert system application. Unfortunately at this time, the second category does require the training of neural networks. The third category, however, is the most important of the three since most applications that people wish to use expert systems for really fall into this category. The cold hard reality of category three is the reason for the inherent impossibility of "complete" knowledge acquisition.

III. How?

To accomplish this task, we require the basic inference engine machinery, a statistical "state keeper," a neural unit generator and supervisor, as well as an arbitrator to handle such conflicts as arise. The arbitrator keeps track of the current settings of the system parameters which affect all areas of interaction.

We alluded in the last section to statistical correlations. These correlations, along with an overall summary of rule usage, represent the backbone of our extended CLIPS structure. Although they are significant baggage to carry around, they serve three very valuable functions:
1) They provide a basis for optimization and learning *solely* on the part of the expert system
2) They provide the migration paths for the neural units
3) They are used to make the user aware of unusually strong correlations which can represent a bug in the knowledgebase, or a serious gap in the knowledge acquisition. Better still, they could actually be used to point out "new" knowledge in the form of genuine relationships between events which had not been previously noted.

The expert system learning is accomplished through the use of "dynamic" salience values - CLIPS salience values for rules which are updated based on rule usage, starting from the initial salience values (if any) declared by the user. This same mechanism allows the user to define different "experts" having different "experiences," by loading different salience values into CLIPS. In addition, this means that an expert system would learn to behave differently if it were placed in different environments.

To complete the expert system interface to the neural network, we endow the CLIPS structure with three additional properties which are generally associated with neural networks: firing thresholds, back-propagation, and rule learning methods which we will call filter functions.

The firing threshold construct is made possible by our single addition to the CLIPS fact: certainty factors. Rules can be thresholded to different values based on the summed certainty of the information which the rule is acting upon. Thus certainty affects the execution of rules since a rule will not fire if the overall certainty of information does not reach the necessary threshold for that rule.

A loose form of back-propagation has been implemented in the form of a "Reality Inspector," which in addition to providing an explanation facility, allows the user to replace an inappropriate answer with a "better" answer, and have the system readjust salience values appropriately. Since this is a potentially dangerous operation, large changes of this nature are discouraged. The filter functions determine the amount of activity which must occur for a given rule to have its salience value adjusted. We call them filter functions, because they filter out what the user defines as an "irrelevant" amount of stimulation, or lack of it.

To accommodate all of this, we have created a system in which information is cyclically evaluated first by the CLIPS inference engine, okayed or altered by the arbitrator which then checks for the existence of any powerful*[2] neural units which could force a different path from that which was agreed upon by the CLIPS engine and the arbitrator. After this last step is performed the information is passed to the object that handles the graphic display. All updating of system information is performed *after* the session (unless otherwise requested) in order to minimize the amount of time that the user must wait while the system updates.

IV. Computation

Unfortunately this task requires a large amount of computation, and eats a nice chunk of memory for storage. Using the NeXT somewhat minimizes the latter problem, since the optical disks utilized by NeXT are intended for storage of large amounts of information. The computation is a weightier problem. However, there too the use of the NeXT affords an advantage due to the presence of the DSP (digital processing chip) which allows for rapid array processing. Since space is limited, we will concentrate on the computation and maintenance of the correlation data.

The DSP requires all inputs to lie in the interval [-1, 1]. As we will see, this is sufficient for our purposes. The first step of encoding the data consists of constructing a rule network from the rules declared in CLIPS. Beginning from the first "layer" having more then one rule in the network, we determine the boundary of the knowledge space by arbitrarily assigning one of these rules a tag of -1, and another rule a tag of +1. The tag represents ownership of an interval around the tag. Additional rules are assigned tags which are equidistant from one another. For example, if there were only one additional rule it would be assigned a tag of 0. Proceeding to the next layer down, we repeat the process. Only this time, rules descending from a parent must share the portion of the interval which was allotted to the parent. This process continues until the all of the expert system rules have been similarly assigned a tag.

The mathematical set formed from this process is the interval [-1, 1] - {the set of boundary points between adjacent owned territory.} Physically, the set can be thought of as a dotted line with the number of gaps in any part of the line being proportional to the number of rules occupying that part of the interval. Each time the system is run to completion, this skeleton set is "filled in" to show which rules fired during that run.

The path taken by the system is like a mathematical footprint which describes the order of rule execution. All of the necessary information is derivable from this set. A new image of the system state is created for each complete system run. These images can be stacked on top of one another to create a pictorial as well as mathematical three dimensional system history.

Since one of the system's tasks is to alert the user to unusually high correlations between rule firings*3, the system must be continually aware of the occurrence of these events. This is accomplished through inspecting different similarly sized peaks to see if the contexts of the rules firing matches up with the absolute number of firings. Clearly a peak located in a subpartition of another peak's interval is not of interest, since the first rule would be a direct ancestor of the second rule. If an unusual correlation is discovered it is reported to the user, who can then decide if some modification is merited.

V. The Neural Units

There is not sufficient space for an extended discussion of the neural units, however a few words on the subject are merited. Clamped neural units with a small number of hidden layers can appear either through system allocation, or by user request. The system will notify the user each time it adds a new neural unit. The user will then be requested to specify a set of inputs and outputs, and may then begin training. If the user chooses not to train the neural unit, the system will *not* remove the unit unless explicitly instructed to do so by the user, but instead will train it randomly. This is because the allocation of a neural unit is a *considered* action on the part of the system which is intended to inform the user that some system attention should be devoted to the area of the knowledge space where the neural unit was placed. The training of the neural unit is quite similar to the previously described "Reality Inspector," in order to minimize the difference between the two components.

Just as in the expert system case, very frequently used neural unit output nodes will spawn addition neural units following the conventions described above. The system will have a small selection of accepted learning methods from which the user may choose. The user does not directly control the migration of the neural units, although he/she can adjust some system parameters which will affect the neural units' definition of sufficient proximity for migration. Determination of "proximity" in the case of the so-called concept generators is a "hard-wired" behavior of neural units which are entirely concealed from the user.

The neural units described in this paper are clearly differentiated in purpose, as well as "physical" appearance: for example, neural units which have migrated to other nodes maintain an umbilical cord from the parent node. This differentiation results from a combination of the experiences which the expert system as a whole happens to have, as well as the placement of the individual unit. On a conceptual level this is quite similar to contemporary neurobiological models of neuronal differentiation in the brain.

In summary, the neural units augment the capabilities of the expert system by providing recognition of detail and imperfect instances, gap-filling in the knowledge acquisition, and the important power of association between vaguely similar items. As with human beings, there can be no guarantee that every association is a valuable, or relevant one. Yet it is inarguable that much of human memory and reasoning ability stems from the capability to recognize unformalized similarities between otherwise unrelated pieces of information.

VI. Other Supporting Features

In addition to a standard windowed environment, our system supports several user interface oriented features worthy of discussion since they augment the value of the system capabilities which were discussed in the previous sections.

We have added the construct of rule "groups." This is a construct which is completely external to CLIPS, but which is useful to represent system progress to the user. Our system allows the user to specify execution paths with the mouse that temporarily override all other existing salience values. The group construct is also used for many of the graphics representations discussed below.

Rules can also be defined as objects in a limited sort of way. Since our rules have eight attributes in all (name, definition, category, salience, threshold, object type, filter function and group,) it is inconvenient for the user to have to replicate this information for every rule in a large set of rules which the user wishes to have several identical attributes. Thus an object type is determined by a user defined ID name, the number of traits the user wishes the rules to have*[4] and any default settings, such as a threshold value. In addition, each rule group must have a filter function associated with it that all of its rules possess. Object type is also completely external to the CLIPS engine.

Although both rules and facts have optional "category" slots for reference purposes, our system offers additional aid for knowledge extraction in the form of a "librarian" which maintains a record of the *context**[5] of rule usage. This corrects for any errors or omissions made categorizing the information initially.

Another important system feature is the presence of the five different graphic visualization methods which are part of the knowledge debugging environment. These representations encourage the user to view data in different ways which accent different traits of the data. However, the most important function of these different modes is making the system as transparent as possible to the user. Doing so makes the task of using the system more interesting because it involves genuine comprehension of the underlying knowledge, as well as some degree of demystification about the inner workings of the system. In addition to switching between graphic modes the user may take "snapshots," from different perspectives which yield different simultaneous views. The user may collect these snapshots in a "photo album."

The different graphic visualizations are as follows:

1. a rule network, in which each rule is represented as a node connected to other nodes which can result from its firing. Under this mode, the user may view the correlations between different nodes which are illustrated as connecting lines whose width varies according to the degree of the correlation, and whose coloration varies according to whether the correlation is positive or negative. The user may also graphically view the different rule thresholds as well as the certainty with which each rule fired following the end of a completed run. Neural units are visible as entities, but without any detail.
2. a condensed overview of the entire system which, in a biological analogy, depicts the regions of greatest neural unit activity as having "denser tissue." Individual regions can be more closely examined with a "microscope"
3. a "fuzzy" view showing a partition of the knowledge space which accents the uncertainty of information using a fuzzy set representation.
4. a moving drivers' seat view graphically depicted by the entirely on the graphics screen filling with a graphic object representing the rule group currently being examined.
5. a text-based explanation mode

This diversity of viewpoint is necessary to ensure that the user is able to comprehend the information which the system is acting upon. We believe that one of the principal functions of an expert system should be helping its user to better understand the expert system's *knowledge*, not just the expert system. Thus, an expert system can be thought of chiefly as fulfilling a knowledge distribution function, while increasing its knowledge store both through its own experience and through modification by very "knowledgeable" users.

VII. Conclusion & Summary

We have discussed the integration of a CLIPS-based expert system and neural networks in a unified, cooperative system. Our conclusion is that these two AI methodologies should not be viewed as antithetical to one another, but rather as naturally symbiotic partners operating in complimentary portions of the knowledge domain.

Footnotes:

*1 structural in terms of data involving patterns of occurrence.
*2 A powerful neural unit is one that has been very well "fed" by the expert system.
*3 that is, non-trivial rule firings. The relationships between the firing of rules directly descended from one another is not of interest in terms of providing data for associations.
*4 For each new rule, only name and definition *must* be entered. Category and group are optional: if a rule is not assigned one, it does not receive any default value. If no threshold is defined, the rule is presumed to have none. If no salience is defined, an average salience value is assigned by the system. If no object type is defined, the rule is presumed to be of the generic type.
*5 By *context* we mean the knowledge category context. The system keeps track of what each rule was used for, based upon the category of the end result. This ensures that all of the system knowledge necessary to operate in some subdomain of the system is extractable.