GALATEA: A C-LIBRARY FOR CONNECTIONIST APPLICATIONS

C. Mejia, L. Bottou, F. Fogelman Soulié

Laboratoire de Recherche en Informatique, bât. 490
Université de Paris Sud- 91 405 ORSAY Cedex - FRANCE

1 Introduction

Neural Networks are techniques which are increasingly used for real world applications. However, developing such applications does not simply mean applying academic ideas or algorithms: efficient programming tools and environments are required if one wants to efficiently design networks with optimal performances.
The ESPRIT-Pygmalion project has been set up to achieve a complete programming environment for connectionist applications. This environment will include both a Hich Level and Intermediate Level languages, a graphic monitor, and 2 Libraries: one in C and the other in the HL Language. Partners in the project are Thomson (HLL), UCL (ILL and graphic), IRIAC-LRI (Libraries) and INESC (C-Library). We present here the C-Library, Galatea, which has been designed so as to allow the partners in Pygmalion to easily develop and test their applications.

2 Galatea: the Algorithms C-Library in the Pygmalion Project

The C-algorithms library Galatea was intended to provide to partners in the ESPRIT-Pygmalion project a tool including efficient versions of the algorithms most commonly used for the applications developed within the project, i.e. image and speech processing. When developing the Library, we tried to enforce a unified description of the algorithms to ease the designing process of algorithms modules and further developments in the High Level Language of the Pygmalion environment.

2.1 User paradigm

A critical decision consists in defining the kind of objects the user will deal with. We might have defined a very general data structure, but it would have been too complex for most algorithms, and too restrictive for some others. Unless we use an ad-hoc compiler, such a choice would thus lead to inefficient programs for most algorithms and almost useless programs for other algorithms!
We describe a network as a black box, which has an input vector and an output vector. We also provide some ways to access internal variables as vectors. The user will first *define* a variable for its network, he will also ***initialize*** this network, with a specific network topology. He can then access the input and output vectors, the internal states, the weights; and set parameters. He can also *operate* the network: i.e. train the network, or retrieve results with it.

2.2 Vector-oriented data structures

We do not enforce a high level data structure, but a rather low level data structure: vectors of floating point numbers. The states of a single layer are grouped into a single vector, as well as the weights, and more generally all the internal data. This philosophy has a drawback: neural computations can poorly be deduced from the vector structure, except in some very simple cases (fully connected networks, for instance). However, in many applications, data naturally come as sets of vectors.

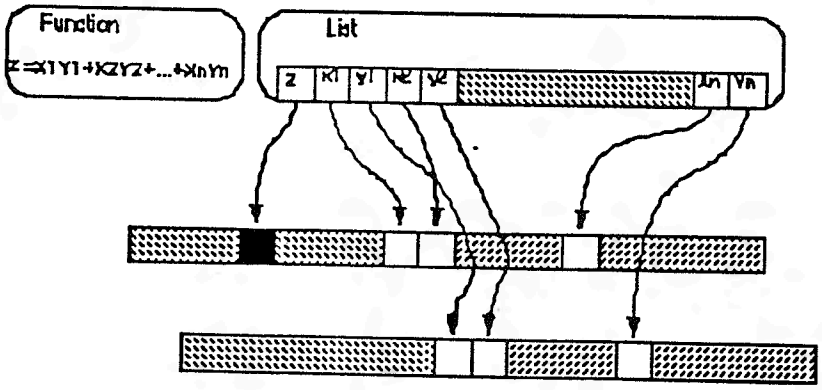

Figure 1: A recomputation rule example. The execution of this rule will compute the dot product x.y into z.

2.3 Computation rules

We also provide support routines for specifically describing neural computations. These routines are based on the idea of *recomputation rules*. A recomputation rule (fig.1) closely looks like a formula in a spreadsheet. They are basically composed of a function, and of a list of pointers to vectors elements. When a rule is executed, the list of pointers is passed to the function, which then performs a given computation.

Rule management functions include rules creation, extension, deletion, and execution. The rule system avoids the definition of complex data structures for designing the network architecture. They will simply describe the computations to be done rather than the network structure. These functions will be more precisely described below.

3 Galatea organization

3.1 Components

We describe now more precisely the Galatea components. Galatea has five main parts (fig.2):

- The *Algorithm Independent Part* (AIP) contains the computation support routines. It includes functions performing on vectors, functions for dealing with recomputation rules, and some basic routines for memory and error management.
- The*Tools Library* contains routines for managing standardized data files, network architecture files, and functions for implementing the most common classification criteria.
- The *Algorithm Modules*, contain the programs for the connectionist algorithms implemented.
- *Algorithm Evaluation Programs* are text oriented front-ends for programmer-designed modules. These front-ends allow the users to rapidly test various algorithms for their applications.
- These programs will be written using a standard *Environment Library* that implements most of the text oriented front-ends. Writing an evaluation program merely amounts in defining the command names, the associated C functions, and calling a MainLoop function. The Environment Library provides functionalities for managing help files, command files, and calling the system functions.

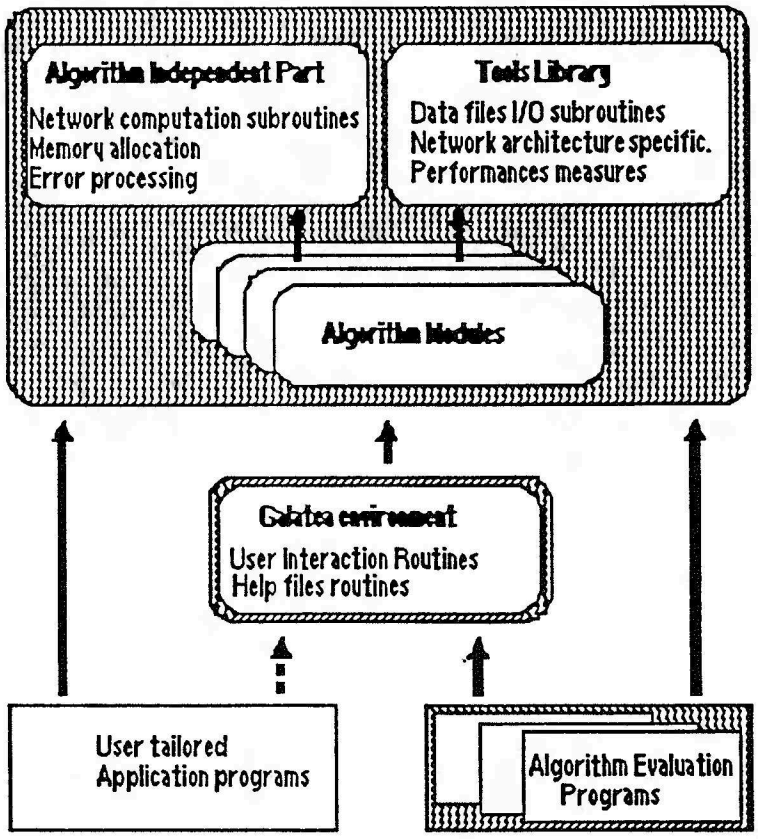

Figure 2: Organization of Galatea

3.2 Algorithms modules in Galatea

Galatea includes at present a large range of algorithms (fig.3). More will be added in the near future.

LAM	Linear Associative memories (Pseudo-Inverse Methods)
Hop	Hopfield nets
Kan	Kanerva associative memory
BAM	Bi-Directional Associative Memory
GBP	Gradient Back Propagation algorithm
GBPF	Gradient Back Propagation with Feed-back
TMap	Kohonen Topological maps
SimAnn	Simulated Annealing
BM	Boltzmann Machines
ART1	Adaptive Resonance Theory
LVQ	Learning Vector Quantization

Figure 3: Modules implemented in Galatea

4 How to use Galatea?

4.1 Using the C library.

The easiest way to use Galatea is through the Algorithm Evaluation Programs. These small programs are text-oriented interfaces to the Galatea functions. They are able to work with one algorithm and one network only. They share a common environment, common file formats, and similar commands.
Many users, however, need more powerful or more specific network programming. They should then write a C language program, that directly calls the Galatea routines. So, they gain the ability to deal with many networks, using different algorithms. They also can compile and link a complete C program.

4.2 An example

We show in figure 4 a program written using the GBP evaluation program to perform a task of hand-written digit recognition [Bottou 89].

```
#
# a hand written numbers classifier
#
echo "--loading the data ..."
data load numin.dat numout.dat                          { load the data files}
echo "--defining training set"
data training 0 319                        {use 320 first patterns for training}
echo "--defining test set"
data test 320 479                          {use 160 following patterns for test}
echo "--loading the network ..."
network load handnum.net                          {load network topology file}
echo "--setting parameters"                 {this is a MLP with shared weights}
forget inverse 2.4                                {set weights initial values}
epsilon sqrt 0.1                                        {set iteration step}
echo "--set classify criteria by maximal range cell"
classify max
echo "--status display"
show width 80
status                                  {display the status of network and data}
echo "--training performance (before learning)"
perf range 0 9             {compute and display performances for 10 first digits}
echo "--loading weights ..."
load-weights handnum.wei             {load weights from another learning session}
echo "--training performance (after learning)"
perf range 0 9             {compute and display performances for 10 first digits}
```

Figure 4: Command file in the GBP evaluation environment

As can be seen from this command file, programming a multi-layer network, even with a complicated architecture (here with shared weights [Lang, 88, Bottou 90]), is very easy because of the use of the predefined functions provided by Galatea in the Tools Library.
The previous command file produces the following script file (fig. 5):

```
Pygmalion C library - GBP evaluation program
(C) IRIAC 1989
--loading the data ...
--defining training set
--defining test set
--loading the network ...
--setting parameters
--set classify criteria by maximal range cell
--status display
Network      :  loaded from <handnum.net>
                aged <0> sweeps, epsilon sqrt <0.1000>
                momentum not allowed, decay not allowed
Data         :  <480> patterns (input+output) loaded
Patterns     :  a <480 x 256> matrix
Desired      :  a <480 x 10> matrix
Training set :  from <0> to <319>
Test set     :  from <320> to <479>
Classify     :  <max>
Function     :  <standard sigmoid (-1.7 to 1.7)>
--training performance (before learning)
Set {0,9}: sweep 0     , Error= 0.29643, Performance= 20.00
--loading weights ...
--training performance (after learning)
Set {0,9}: sweep 0     , Error= 0.06378, Performance=100.00
gbp> forget sqrt 1.5                  {here Galatea lets you type-in new commands}
gbp> epsilon sqrt 0.1
gbp> run 1 4
---------------------                                          {performances on}
Set {0,319}: sweep 1     , Error= 0.16296, Performance= 55.00      {learning set}
Set {320,479}: sweep 1     , Error= 0.17202, Performance= 38.75         {test set}
---------------------
Set {0,319}: sweep 2     , Error= 0.13862, Performance= 77.81
Set {320,479}: sweep 2     , Error= 0.15280, Performance= 67.50
---------------------
Set {0,319}: sweep 3     , Error= 0.12355, Performance= 89.38
Set {320,479}: sweep 3     , Error= 0.13975, Performance= 79.38
---------------------
Set {0,319}: sweep 4     , Error= 0.11456, Performance= 93.75
Set {320,479}: sweep 4     , Error= 0.13253, Performance= 85.63
```

Figure 5: script file in Galatea

5 Conclusion

We have presented the Galatea Library which has been developed for the ESPRIT-Pygmalion project. Galatea intends to offer a fast access to the most commonly used algorithms at the present time. In Galatea, the user can easily incorporate the library functions into his own program implementation for his specific application. In addition to the inherent ability to switch from one algorithm to another, when testing an application, through its friendly text-environment, Galatea invites us to standardize the programming style. Galatea has been widely distributed both within and out of the Pygmalion consortium. Comments about its use for developing real size applications are expected from the users.

Galatea has been developed by the IRIAC team, at LRI (Université de Paris-Sud). INESC (Portugal) contributed some of the modules. The Library is available upon request from the authors.

References

L. Bottou, F. Fogelman Soulié, P. Blanchet, J.S. Blanchard: Speaker independent isolated digit recognition: multi layer perceptrons vs dynamic time warping. Neural Networks. To appear.

K. Lang, G.E. Hinton: The development of TDNN architectures for speech recognition. Tech. Report CMU-CS-88-152, 1988.

C. Mejia, L. Bottou: C Library preliminary specifications. Document 1, ESPRIT Pygmalion project n°2059, 1989.

L. Bottou, X. Driancourt, C. Mejia, E. Viennet: Evaluation of the C-Library. R140-3 Report M12. ESPRIT Pygmalion project n°2059, 1989.