About Emergent

See the email Announcement for more specific contrasts with PDP++ [1], and Authors for who wrote it, and Grant Support that funded its development. The ChangeLog contains a list of changes for various releases.

Emergent (formerly PDP++ [1]) is a comprehensive simulation environment for creating complex, sophisticated models of the brain and cognitive processes using neural network models. These networks can also be used for all kinds of other more pragmatic tasks, like predicting the stock market or analyzing data. Emergent includes a full GUI environment for constructing networks and the input/output patterns for the networks to process, and many different analysis tools for understanding what the networks are doing. It has a new tabbed-browser style interface (in Qt 4), with full 3D graphics (via Open Inventor/Coin3D), and powerful new GUI programming tools and data processing and analysis capabilities. It supports the same algorithms as PDP++: Backpropagation (feedforward and recurrent), Self-Organizing (e.g., Hebbian, Kohonen, Competitive Learning), Constraint Satisfaction (e.g., Boltzmann, Hopfield), and the Leabra algorithm that integrates elements of all of the above in one coherent, biologically-plausible framework.

PDP++[1] was under constant development since 1993, and over the past several years has been completely overhauled and released with a new name: emergent (first released in August of 2007). The GUI was completely rewritten using Qt from Trolltech (the basis of the popular KDE environment) and Coin3d (an implementation of the Open Inventor framework for 3D scenegraph rendering). Everything has been boiled down to the most essential elements, which are now much more powerful and flexible. It is now much easier to understand and modify the flow of processing, and to create programs that automate various tasks, including the creation of complex training environments. Important new visualization tools have been added, and many optimizations have been implemented in terms of the memory footprint and script execution speed. There is also a convenient plugin architecture that makes extending the software much easier than before.

Reasons to Use Emergent and Motivations for its Development

1. Provides many powerful visualization and infrastructure tools tailored to the neural network domain that would otherwise take a lot of individual effort to write.
2. Provides a structured environment and GUI support for making it as easy as possible for others to use and modify models (e.g., an automatic GUI interface for operating the models, a GUI programming environment with high-level constructs that are easily understood and modified, builtin documentation system, etc).
3. Completely open source -- this makes it much easier to use emergent in educational or other cost-sensitive situations, in addition to all of the other major advantages of the open source development model.
4. Highly optimized runtime performance and extensive builtin object-oriented support for complex network structures -- supports multiple forms of parallel processing including distributed memory (clusters) and shared memory (threads).

Although tools like MATLAB satisfy point number 1 above, they do not satisfy the other important points. Put simply, if you are primarily developing new algorithms, use MATLAB. If you are primarily using existing algorithms and developing complex high-performance models, use emergent (if you're just using Backprop and relatively simple network architectures, LENS may be faster, but the GUI is not as powerful). If you are making detailed multi-compartment models of individual neurons, use NEURON or GENESIS.
General Characteristics of Emergent

- Written in C++
- Gui-based, cross platform (Windows, MacX, Linux/Unix)
- Uses the Trolltech Qt graphical and foundation class library see: http://www.trolltech.com
- Use the Coin "Inventor" library for Open-GL-based 3D visualizations see: http://www.coin3d.org
- Supports the multiple different neural network learning and processing algorithms, from backpropagation to more biologically-based algorithms including Leabra.
- Provides for dynamic "plugins" which can extend the main program capabilities in many ways (new network algorithms, new visualizers, new data sources, network interfaces, etc.)
- Provides low-level multi-dimensional matrices, and higher level "tables" (similar to an in-memory database) for data logging and manipulation
- Provides visual graphing and logging
- Networks can be entirely internal (data input/output driven inside the program) or can be hooked to external input/output (ex. a simulation environment, ACT-R, etc.)
- Provides a data analysis framework, including multi-dimensional matrices, statistical processing, data summarization, etc.
- Linked against the Gnu Scientific Library -- our matrix class is compatible with the matrix classes defined in that library see http://www.gnu.org/software/gsl/
- Can be run on clusters under MPI (Message Passing Interface)
- Supports multi-CPU/core computers
- Includes TA and CSS (described below)

Libraries and Components of Emergent

The Emergent Toolkit

This is the foundation on which emergent is built (see TEMT wiki [2] for more details). It includes TA and CSS capabilities described below, basic application framework classes (String, Variant, lists, arrays, matrices, tables), a general-purpose application browser, a tabbed property editor (extensible to provide custom visualization and editing views for various types of application objects), a full 3D visualization/rendering library, low-level text-based CSS scripting and higher-level gui-based Programs (which render CSS scripts to execute.)

TA (TypeAccess)

TypeAccess (TA) is a sophisticated meta-type information system for C++ programs. The header files of the application are scanned by a program called maketa which builds a comprehensive description of all the classes (TypeDef) and their methods (MethodDef), members (MemberDef), etc. This information is then available at runtime, which enables things like:

- Programmatic access to members and methods
- Automatic streaming of complex object hierarchies
- Automated building of property edit dialogs
- Dynamic creation of object instances based on dynamic runtime typing
- Dynamic generic runtime manipulation of objects, such as querying its members, methods, class type, whether it inherits from a particular class, etc.
- Full API docs avail in Help Browser and on this wiki.

TA basically provides the same kinds of services as "reflection [3]" in systems like Java and C#. A set of directives can be applied to comments in the header files to control runtime behavior, such as visibility of member variables, etc.
CSS (C-Super Script)

CSS is a C++-like scripting language built into emergent. It is used as the basis of network simulation control, and can also be used to customize data input/output to networks, and to facilitate interactions with external components.

Third-Party Software Emergent Interfaces with

Qt

Qt is a comprehensive application framework and gui library. It is C++-based, cross-platform, and runs on virtually every modern system. It also has Python and (new) Java bindings. It has basic classes such as String, Variant, etc., container classes (lists, etc.), a complete graphical framework (windows, widgets, etc.), and libraries for network/socket access, and XML. Qt includes an elegant but sophisticated mechanism "signals and slots" for dynamically connecting objects together at runtime. It also includes a meta-typing system, which provides somewhat of a subset of the TA system.

Coin / Inventor

"Inventor" is a 3D API for describing collections of 3D objects for visualization. It renders these objects using the OpenGL framework. "Coin" is an open-source implementation of the Inventor API.

References

Article Sources and Contributors

About Emergent: Source: https://grey.colorado.edu/emergent/index.php?oldid=8673  Contributors: Administrator, Brian.mingus, Dpfurlani, Oreilly

Image Sources, Licenses and Contributors