Yawn Framework v.0.3

Project Documentation

GIAA, Universidad Carlos III de Madrid

09 May 2005

Table of Contents

1.1 Home

Welcome to Yawn

Yawn (*Yet Another {Weird | Wacky | Witty | Wonderful | W*} Network*) is a Java framework for training and testing neural networks independently from the neural model and the test environment being used.

The models implemented so far are AppART , GasART , fuzzy ARTMAP and Multi-Layer Perceptron (MLP). Note that the MLP implementation, although fully workable, is a reference implementation meant for educational use and might yield less-than-optimal results.

Yawn does not perform any kind of statistical pre- or post-processing neither any results analysis. These tasks are left over to the chosen test environment. Currently, Yawn is capable of interacting with the DELVE test environment and with plain text files. A Matlab and a xml-file test environment are currently under development.

Yawn is an open source software distributed under the GNU license . Yawn home on the web is yawn2.sourceforge.net (*). It is under constant development, therefore, if you are seeing an offline version of this document we recommend that you visit the site. The SourceForge project portal provides access to general project info, mailing lists, forums, bug submissions, feature requests, and a big etc..

If you use Yawn, please, let us know,

--
The Yawn Team

* Why yawn2? Because yawn.sf.net was already taken :).

1.2 Status

Current work

These are the main lines of development of Yawn.

- neural network serialization in xml; (done)
- configuration migration from Digester to Castor; (done)
- migration from ant to maven; (done)
- more JUnit test cases;
- completion of javadoc comments;
- improve error and exception management, and;
- general refactoring.

Long-term features

- complete documentation;
- lightweight networks suitable to be used on Lego Mindstorms (*) robots with the LeJOS virtual machine;
- implementation of PROBART, ellipsoid ARTMAP, etc. (tell us!), and;
- a graphical user interface (we are procrastinating about a posible integration with joone).

You can help us making a better Yawn. You can either checkout the current Yawn from the CVS and work on it, fill out our feature request form , or volunteer to work with us.

* **Note:** Yawn is not authorized, sponsored, or legally associated with the Lego Corporation in any way .

1.3 Institutions

Participating Institutions

The following institutions have taken part in the development of Yawn:

Laboratorio de Inteligencia Artificial , Facultad de Matemáticas y Computación , Universidad de La Habana , Cuba.

Dipartimento di Matematica e Informatica , Università degli Studi di Udine , Italy.

1.4 Download Page

http://sourceforge.net/project/showfiles.php?group_id=102668

1.5 SF.net Project Portal

http://www.sourceforge.net/projects/yawn2

2.1 Distributions

How to obtain Yawn

There are three distributions of Yawn:

- the binary distribution;
- the binary distribution with library dependencies, and;
- the source distribution.

Binary Distribution

The binary distribution consists of:

- `<yawn-dir>/yawn-<version>.jar` file;
- the maven generated documentation in `<yawn-dir>/docs`, and;
- a samples directory (`<yawn-dir>/samples`) with two problems:
  - A problem generated from DELVE's Demo data set (in `<yawn-dir>/samples/delve-demo-task`), and;
  - the circle-in-the-square problem in DELVE file format (in `<yawn-dir>/samples/circle-in-square`).

For each problem different sample experiment configs are provided. Note that this configurations are provided illustrative purposes and may not be optimal for the problem you are solving. Some elements of the configuration may be platform dependant (i.e. file system paths).

>

In order to make Yawn work you must also download and add to your CLASSPATH the project dependencies. See the dependencies page for information.

>

See the downloads page to obtain this distribution.

Binary plus dependencies

To make life easier for everyone we have released a binary distribution that contains all the project dependencies as .jar files in the `<yawn-dir>/lib` directory.

>

Go to the downloads page to obtain this distribution.

Source Distribution

The source distribution can be obtained either from the release site or directly from the cvs repository .

For details of about the source distribution, how to use it and build it see the developers guide .

2.2 Concepts

Fundamental Concepts

Yawn is built on top of three fundamental concepts: *neural networks*, *environments* and *experiments*.

A neural network class extends `yawn.nn.NeuralNetwork` implementing all the computational logic behind a given neural model. Note that although we have chosen the term `neural network' any other machine learning method can be added to Yawn.

A test environment is any program or system capable of supplying train and testing datasets to Yawn and receives predictions in order to process and asses them. For example, the PlainTextEnvironment (`yawn.envs.plaintext.PlainTextEnvironment`) read data from a text file and produces a text file with the predictions obtained by Yawn. All environment classes extends `yawn.envs.Environment`.

Finally, an experiment is the combination of a particular setup of a neural network and a test environment.

Networks Committees

A network committee (`yawn.nn.committe.NetworkCommittee`) is an extension to `yawn.nn.NeuralNetwork` that does not represents a neural network. Instead, it represents a *committee* or *set* of neural networks that are trained with the same dataset (this is not completely true since each network is trained with a different ordering of the dataset). This set can be later used concurrently to generate predictions more statistically valid.

One of the benefits of the NetworkCommittee class is that it can contain different implementations of neural models and/or models with a different set of initial parameters. The networks that are members of a committee are trained in a multi-threaded fashion. Keep this in mind when implementing your own models.

2.3 Tools

Working with Yawn

Although Yawn was primarly developed to be used as a software library we have developed two helper applications for interacting with it.

RunExperiment

Predict

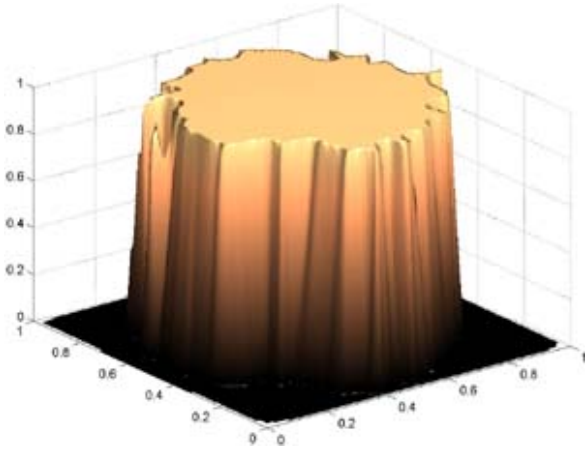
2.4 A Sample Problem

A Test Case: The Circle in the Square

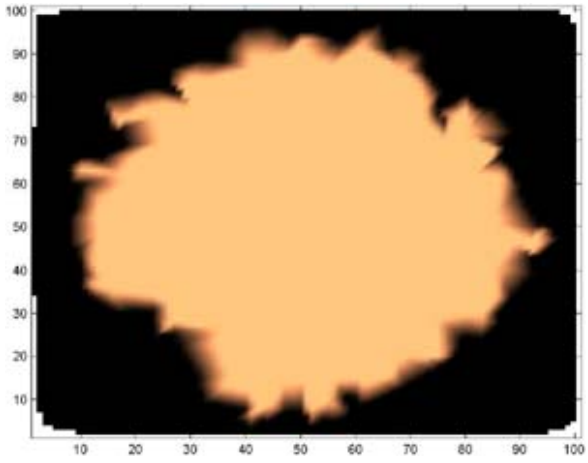
In the circle in the square problem the network must learn to classify points that are inside a circle and those that are not. In the sample files included with yawn the network must predict a "1" if the point is inside de circle and a "0" if not.

The figures bellow show the original test set and the predictions made by fuzzy ARTMAP and AppART. These figures were created with `plot_cs.m`, a matlab .m file included in the `<yawn-dir>/data/circle-in-square` directory. These results, in particular those of AppART are not optimal. For example, by increasing the `<trainMatchVigilance>` parameter a more precise but less generalizing approximation is obtained.

C-in-S test set

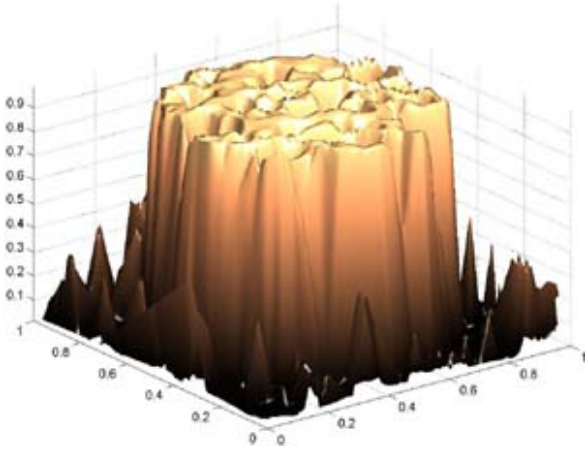


C-in-S test set contour

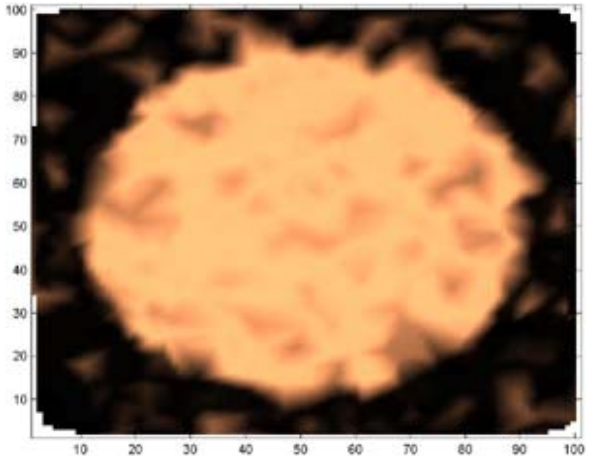


Fuzzy ARTMAP prediction of the test set

Contour of the fuzzy ARTMAP prediction of the test set



AppART prediction of the test set



Contour of the AppART prediction of the test set

3.1 Building Yawn

Building Yawn

Yawn is now fully mavenized , so building it from source is a piece por cake. This also means that you need to download and install Maven if you want to build Yawn.

Maven downloads all neccessary dependencies for you - so you don't need to worry about that any more!

Maven uses a set of standard build commands. For example, if you run maven at the base dir of Yawn:

```
$ maven
```

this executes `'java:jar'` which compiles the source, runs the unit tests and creates a jar.

Maven reads four files for building a project:

- the `project.xml` file, that stores the project description, dependencies, etc.;
- the `maven.xml` file, that customizes the maven goals to meet the project's requierements;
- the `project.properties` file, that defines project wide properties, and;
- the `build.properties` file that contain local (per user) project properties (e.g. proxy configuration, repository passwords, etc.).

Be aware that you might need to edit your build.properties file in order to build Yawn. In particular, if you are in the Universidad de La Habana campus you need to set your http proxy settings.

See the maven documentation for more details and information about the other standard build commands.

We have succesfully used the Meivenide plug-in for Eclipse to manage all Maven related stuff.

3.2 Extending Yawn

Extending Yawn

Implementing Neural Networks

A Note on Persistance

3.3 JUnit Testing

JUnit Test Cases

Test cases for neural networks

We provide a set of classes meant to used with JUnit in order to provide a common testing framework for neural networks implementations.

The `yawn.nn.AbstractNeuralNetworkTest` class contains all the functionality common to all tests and defines abstract methods that are used by descendant classes provide custom environment and network setups and train methods.

For example, the

These test cases are supported by the synthetic environments (see the `yawn.envs.synthetic` package).

When implementing a neural network you should create a

Test cases for configuration classes.

3.4 Old Yawn (0.1) Instructions

Old Yawn distribution (0.1)

Yawn v0.1 is now *depricated*.

System Requirements

The apache ant build tool is the preferred tool for compilation. The build script provided allows the compilation of the package, the generation of a distributable jar-file and the generation of the package's api documentation (javadoc). You can also do it by hand or with the IDE of your preference.

You also need the following libraries in your classpath (in the `<yawn-dir>/lib` directory, if you are using the ant build script as-is):

- the Jakarta Commons Digester component , which at its time requires:
- the Jakarta Commons BeanUtils , Collections and Logging components, and;
- an XML parser conforming to SAX (Simple API for XML) 2.0 or JAXP (Java API for XML Parsing) 1.1. Two open source XML parsers suitable for use with Digester are:
  - Xerces
  - Crimson

If you are in the Univ. de La Habana campus you can find out all these libs in the Java software repository

Using Yawn

A helper application is provided to allow the use of the yawn library of classes. This application is invoked from the command line as:

```
java yawn.core.Main <your-config-file.xml>
```

having yawn (the compiled .class files or the .jar file) in your classpath.

There is a build target called '`run`' in the ant build script provided that does the above described task.

The configuration file for `yawn.core.Main` contains the description of the parameters for each network and test environments. As mentioned earlier, a more or less self explanatory configuration file for the circle-in-the-square problem could be found in `<yawn-dir>config/config.xml`.

4.1 SourceForge.net

http://sourceforge.net