Journal of Machine Learning Research 12 (2011) 2383-2387 Submitted 6/10; Revised 3/11; Published 7/11

Waffles: A Machine Learning Toolkit

Mike Gashler MIKE@AXON.CS.BYU.EDU
Department of Computer Science
Brigham Young University
Provo, UT 84602, USA

Editor: Soeren Sonnenburg

Abstract

We present a breadth-oriented collection of cross-platform command-line tools for researchers in machine learning called *Waffles*. The *Waffles* tools are designed to offer a broad spectrum of functionality in a manner that is friendly for scripted automation. All functionality is also available in a C++ class library. *Waffles* is available under the GNU Lesser General Public License.

Keywords: machine learning, toolkits, data mining, C++, open source

1. Introduction

Although several open source machine learning toolkits already exist (Sonnenburg et al., 2007), many of them implicitly impose requirements regarding how they can be used. For example, some toolkits require a certain platform, language, or virtual machine. Others are designed such that tools can only be connected together with a specific plug-in, filter, or signal/slot architecture. Unfortunately, these interface differences create difficulty for those who have become familiar with a different methodology, and for those who seek to use tools from multiple tookits together. Toolkits that use a graphical interface may be convenient for performing common experiments, but become cumbersome when the user wishes to use a tool in a manner that was not foreseen by the interface designer, or to automate common and repetitive tasks.

Waffles is a collection of tools that seek to provide a wide diversity of useful operations in machine learning and related fields without imposing unnecessary process or interface restrictions on the user. This is done by providing simple command-line interface (CLI) tools that perform basic tasks. The CLI is ideal for this purpose because it is well-established, it is available on most common operating systems, and it is accessible through most common programming languages. Since these tools perform operations at a fairly granular level, they can be used in ways not foreseen by the interface designer.

As an example, consider an experiment involving the following seven steps:

1. Use cross-validation to evaluate the accuracy of a bagging ensemble of one-hundred decision trees for classifying the *lymph* data set (available at `http://MLData.org`).
2. Separate this data set into a matrix of input-features and a matrix of output-labels.
3. Convert input-features to real-valued vectors by representing each nominal attribute as a categorical distribution over possible values.
4. Use principal component analysis to reduce the dimensionality of the feature-vectors.

5. Use cross-validation to evaluate the accuracy of the same model on the data with reduced features.
6. Train the model using all of the reduced-dimensional data.
7. Visualize the model-space represented by the ensemble.

These seven operations can be performed with Waffles tools using the following CLI commands:

```
1. waffles_learn crossvalidate lymph.arff bag 100 decisiontree end
2. waffles_transform dropcolumns lymph.arff 18 > features.arff
   waffles_transform dropcolumns lymph.arff 0-17 > labels.arff
3. waffles_transform nominaltocat features.arff > f_real.arff
4. waffles_dimred pca f_real.arff 2 > f_reduced.arff
5. waffles_transform mergehoriz f_reduced.arff labels.arff > all.arff
   waffles_learn crossvalidate all.arff bag 100 decisiontree end
6. waffles_learn train all.arff bag 100 decisiontree end > ensemble.model
7. waffles_plot model ensemble.model all.arff 0 1
```

The cross-validation performed in step 1 returns a predictive accuracy score of 0.781. Step 5 returns a predictive accuracy score of 0.705. The plot generated by step 7 is shown in Figure 1.

It is certainly conceivable that a graphical interface could be developed that would make it easy to perform an experiment like this one. Such an interface might even provide some mechanism to automatically perform the same experiment over an array of data sets, and using an array of different models. If, however, the user needs to vary a parameter specific to the experiment, such as the number of principal components, or a model-specific parameter, such as the number of trees in the ensemble, the benefits of a graphical interface are quickly overcome by additional complexity. By contrast, a simple script that calls CLI commands to perform machine learning operations can be directly modified to vary any of the parameters. Additionally, the scripting method can incorporate tools from other toolkits, or even custom-developed tools. Because nearly all programming languages can target CLI applications, there are few barriers to adding custom operations. Graphical tools are unlikely to offer such flexibility.

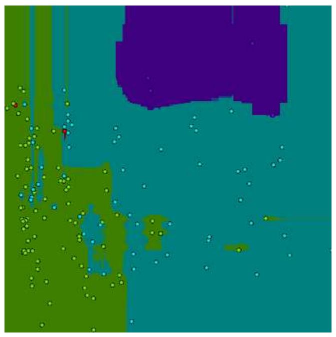
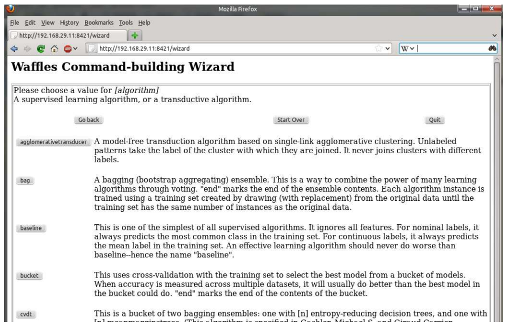


Figure 1: The model-space visualization generated by the command in step 7.


Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://192.168.29.11:8421/wizard
http://192.168.29.11:8421/wizard

Waffles Command-building Wizard

Please choose a value for *[algorithm]*
A supervised learning algorithm, or a transductive algorithm.

agglomerativetransducer	A model-free transduction algorithm based on single-link agglomerative clustering. Unlabeled patterns take the label of the cluster with which they are joined. It never joins clusters with different labels.
bag	A bagging (bootstrap aggregating) ensemble. This is a way to combine the power of many learning algorithms through voting. "end" marks the end of the ensemble contents. Each algorithm instance is trained using a training set created by drawing (with replacement) from the original data until the training set has the same number of instances as the original data.
baseline	This is one of the simplest of all supervised algorithms. It ignores all features. For nominal labels, it always predicts the most common class in the training set. For continuous labels, it always predicts the mean label in the training set. An effective learning algorithm should never do worse than baseline--hence the name "baseline".
bucket	This uses cross-validation with the training set to select the best model from a bucket of models. When accuracy is measured across multiple datasets, it will usually do better than the best model in the bucket could do. "end" marks the end of the contents of the bucket.
cvdt	This is a bucket of two bagging ensembles: one with [n] entropy-reducing decision trees, and one with

Figure 2: A partial screen shot of the Waffles Wizard tool displayed in a web browser.

2. Wizard

One significant reason many people prefer to use tools unified within a graphical interface over scriptable CLI tools is that it can be cumbersome to remember which options are available with CLI tools, and to remember how to construct a syntactically-correct command. We solve this problem by providing a "Wizard" tool that guides the user through a series of forms to construct a command that will perform the desired task. A screen shot of this tool (displayed in a web browser) is shown in Figure 2.

Rather than execute the selected operation directly, as most GUI tools do, the Waffles Wizard tool merely displays the CLI command that will perform the operation. The user may paste it directly into a command shell to perform the operation immediately, or the user may choose to incorporate it into a script. This gives the user the benefits of a GUI, without the undesirable tendency to lock the user into an interface that is inflexible for scripted automation.

3. Capabilities

In order to hilight the capabilities of Waffles, we compare its functionality with that found in Weka (Hall et al., 2009), which at the time of this writing is the most popular machine learning toolkit by a significant margin. Our intent is not to persuade the reader to choose Waffles instead of Weka, but rather to show that many useful capabilities can be gained by using Waffles in conjunction with Weka, and other toolkits that offer a CLI.

One notable strength of Waffles is in unsupervised algorithms, particularly dimensionality reduction techniques. Waffles tools implement principal component analysis (PCA), isomap (Tenenbaum et al., 2000), locally linear embedding (Roweis and Saul, 2000), manifold sculpting (Gashler et al., 2011a), breadth-first unfolding, neuro-PCA, cycle-cut (Gashler et al., 2011b), unsupervised backpropagation and temporal nonlinear dimensionality reduction (Gashler et al., 2011c). Of these, only PCA is found in Weka. Waffles contains clustering techniques including *k*-means, *k*-medoids, agglomerative clustering, and related transduction algorithms including agglomerative transduction, and max-flow/min-cut transduction (Blum and Chawla, 2001).

Waffles provides some of the most-common supervised learning techniques, such as decision trees, multi-layer neural networks, *k*-nearest neighbor, naive Bayes, and some less-common algorithms, such as Mean-margin trees (Gashler et al., 2008). Waffles' collection of supervised algorithms is much smaller than that of Weka, which implements more than 50 classification algorithms. Waffles, however, provides an interface that offers several advantages in many situations. For example, Weka requires the user to set up filters that convert data to types that each algorithm can handle. Waffles automatically handles type conversion when an algorithm receives a type that it is not implicitly designed to handle, while still permitting advanced users to specify custom filters. The Waffles algorithms also implicitly handle multi-dimensional labels. As some algorithm-specific examples, the Waffles implementation of multi-layer perceptron provides the ability to use a diversity of activation functions, and also supplies methods for training recurrent networks. The *k*-nearest neighbor algorithm automatically supports acceleration structures and sparse training data, so it is suitable for use with problems that require high scalability, such as document classification.

As was demonstrated in the first example in this paper, Waffles features a particularly convenient mechanism for creating bagging ensembles. It also provides a diversity of collaborative filtering algorithms and optimization techniques that are not found in Weka. Waffles also provides tools to perform linear-algebraic operations, and various data-mining tools, including attribute selection and several methods for visualization.

4. Architecture

The Waffles tools are organized into several executable applications. These include:

1. **waffles_wizard**, a graphical command-building assistant,
2. **waffles_learn**, a collection of supervised learning techniques and algorithms,
3. **waffles_transform**, a collection of unsupervised data transformations,
4. **waffles_plot**, tools related to visualization,
5. **waffles_dimred**, tools for dimensionality reduction and attribute selection,
6. **waffles_cluster**, tools for clustering data,
7. **waffles_generate**, tools for sampling distributions, manifolds, etc.,
8. **waffles_recommend**, tools related to collaborative filtering, and
9. **waffles_sparse**, tools for learning with sparse matrices.

Each tool contained in each of these applications is implemented as a thin wrapper around functionality in a C++ class library, called *GClasses*. This library is included with Waffles so that any of the functionality available in the Waffles CLI tools can also be linked into C++ applications, or into applications developed in other languages that are capable of linking with C++ libraries. The entire Waffles project is licensed under the GNU Lesser General Public License (LGPL) version 2.1, and also later versions of the LGPL (`http://www.gnu.org/licenses/lgpl.html`). Also, some components are additionally granted more permissive licenses. Waffles uses a minimal set of dependency libraries, and is carefully designed to support cross-platform compatibility. It builds on Linux (with g++), Windows (with Visual C++ Express Edition), OSX (with g++), and most other common platforms. A new version of Waffles has been released approximately every six months since it was first released to the public in 2005. The latest version can be downloaded from `http://waffles.sourceforge.net`. Full documentation for the CLI tools, including many examples, and also documentation for developers seeking to link with the GClasses library can also be found at that site. In order to augment the developer documentation, several demo applications are also included with Waffles, showing how to build machine learning tools that link with functionality in the GClasses library.

References

A. Blum and S. Chawla. Learning from labeled and unlabeled data using graph mincuts. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 19–26. Morgan Kaufmann Publishers Inc., 2001. ISBN 1558607781.

M. Gashler, C. Giraud-Carrier, and T. Martinez. Decision tree ensemble: Small heterogeneous is better than large homogeneous. In *Seventh International Conference on Machine Learning and Applications, 2008. ICMLA '08.*, pages 900–905. Dec. 2008. doi: 10.1109/ICMLA.2008.154.

M. Gashler, D. Ventura, and T. Martinez. Manifold learning by graduated optimization. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, PP(99):1 –13, 2011a. ISSN 1083-4419. doi: 10.1109/TSMCB.2011.2151187.

M. Gashler, D. Ventura, and T. Martinez. Tangent space guided intelligent neighbor finding. In *The 2011 International Joint Conference on Neural Networks (IJCNN)*, July 2011b.

M. Gashler, D. Ventura, and T. Martinez. Temporal nonlinear dimensionality reduction. In *The 2011 International Joint Conference on Neural Networks (IJCNN)*, July 2011c.

M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. Witten. The WEKA data mining software: an update. *SIGKDD Explor. Newsl.*, 11(1):10–18, 2009. ISSN 1931-0145. doi: 10.1145/1656274.1656278.

S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, 2000.

S. Sonnenburg, M.L. Braun, C.S. Ong, S. Bengio, L. Bottou, G. Holmes, Y. LeCun, K.R. Müller, F. Pereira, C.E. Rasmussen, et al. The need for open source software in machine learning. *Journal of Machine Learning Research*, 8:2443–2466, 2007.

J. Tenenbaum, V. de Silva, and J. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000.