This repository | Search Pull requests Issues Gist

mlpack / **mlpack**

Watch 143 Star 1,476 Fork 557

Code Issues 62 Pull requests 29 Projects 0 Wiki Pulse Graphs

mlpack: a scalable C++ machine learning library -- http://www.mlpack.org/

machine-learning-library c-plus-plus deep-learning nearest-neighbor-search regression machine-learning

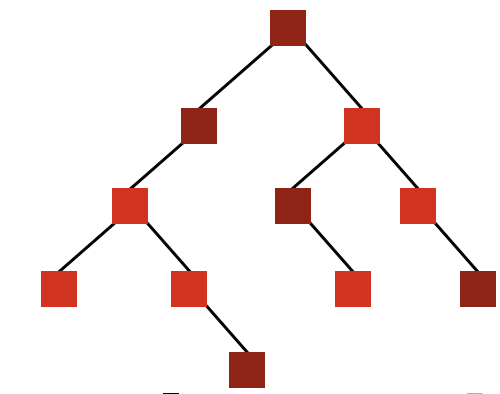
16,216 commits **6** branches **21** releases **84** contributors

Branch: master New pull request Create new file Upload files Find file Clone or download

zoq Initialize visitationOrder for shuffle true and false, to avoid locat… Latest commit `ab975f5` 2 days ago

.travis	travis: try ARMA_64BIT_WORD	2 years ago
CMake	Remove trailing spaces everywhere	a month ago
doc	Update version numbers.	14 days ago
src/mlpack	Initialize visitationOrder for shuffle true and false, to avoid locat…	2 days ago
.appveyor.yml	Update emails for AppVeyor and Travis.	5 months ago
.gitignore	- Mac gitignore changes.	7 months ago
.travis.yml	Decrease process number.	3 months ago
CMakeLists.txt	Update dependencies.	a month ago
COPYRIGHT.txt	Add first non-human contributor.	15 days ago
Doxyfile	Add nice Javascript search box (thanks Dirk for the tip).	2 months ago
HISTORY.md	Update history.	14 days ago
LICENSE.txt	Update license information to note other licenses.	10 months ago
README.md	Fix github listing (looks like github updaed the markdown preprocessor).	a month ago
UPDATING.txt	Outline the versioning policy. (This can change, of course, but I thi…	a year ago

README.md



a scalable C++ machine learning library

Download: current stable version (2.2.0)

mlpack is an intuitive, fast, scalable C++ machine learning library, meant to be a machine learning analog to LAPACK. It aims to implement a wide array of machine learning methods and functions as a "swiss army knife" for machine learning researchers.

0. Contents

1. Introduction

The mlpack website can be found at http://www.mlpack.org and contains numerous tutorials and extensive documentation. This README serves as a guide for what mlpack is, how to install it, how to run it, and where to find more documentation. The website should be consulted for further information:

- mlpack homepage
- Tutorials
- Development Site (Github)
- API documentation

2. Citation details

If you use mlpack in your research or software, please cite mlpack using the citation below (given in BiBTeX format):

```
@article{mlpack2013,
  title     = {{mlpack}: A Scalable {C++} Machine Learning Library},
  author    = {Curtin, Ryan R. and Cline, James R. and Slagle, Neil P. and
               March, William B. and Ram, P. and Mehta, Nishant A. and Gray,
               Alexander G.},
  journal   = {Journal of Machine Learning Research},
  volume    = {14},
  pages     = {801--805},
  year      = {2013}
}
```

Citations are beneficial for the growth and improvement of mlpack.

3. Dependencies

mlpack has the following dependencies:

```
  Armadillo     >= 4.200.0
  Boost (program_options, math_c99, unit_test_framework, serialization,
         spirit)
  CMake         >= 2.8.5
```

All of those should be available in your distribution's package manager. If not, you will have to compile each of them by hand. See the documentation for each of those packages for more information.

If you are compiling Armadillo by hand, ensure that LAPACK and BLAS are enabled.

4. Building mlpack from source

This section discusses how to build mlpack from source. However, mlpack is in the repositories of many Linux distributions and so it may be easier to use the package manager for your system. For example, on Ubuntu, you can install mlpack with the following command:

```
$ sudo apt-get install libmlpack-dev
```

There are some other useful pages to consult in addition to this section:

- Building mlpack From Source
- Building mlpack Under Windows

mlpack uses CMake as a build system and allows several flexible build configuration options. One can consult any of numerous CMake tutorials for further documentation, but this tutorial should be enough to get mlpack built and installed.

First, unpack the mlpack source and change into the unpacked directory. Here we use mlpack-x.y.z where x.y.z is the version.

```
$ tar -xzf mlpack-x.y.z.tar.gz
$ cd mlpack-x.y.z
```

Then, make a build directory. The directory can have any name, not just 'build', but 'build' is sufficient.

```
$ mkdir build
$ cd build
```

The next step is to run CMake to configure the project. Running CMake is the equivalent to running `./configure` with autotools. If you run CMake with no options, it will configure the project to build with no debugging symbols and no profiling information:

```
$ cmake ../
```

You can specify options to compile with debugging information and profiling information:

```
$ cmake -D DEBUG=ON -D PROFILE=ON ../
```

Options are specified with the -D flag. A list of options allowed:

```
DEBUG=(ON/OFF): compile with debugging symbols
PROFILE=(ON/OFF): compile with profiling symbols
ARMA_EXTRA_DEBUG=(ON/OFF): compile with extra Armadillo debugging symbols
BOOST_ROOT=(/path/to/boost/): path to root of boost installation
ARMADILLO_INCLUDE_DIR=(/path/to/armadillo/include/): path to Armadillo headers
ARMADILLO_LIBRARY=(/path/to/armadillo/libarmadillo.so): Armadillo library
```

Other tools can also be used to configure CMake, but those are not documented here.

Once CMake is configured, building the library is as simple as typing 'make'. This will build all library components as well as 'mlpack_test'.

```
$ make
```

You can specify individual components which you want to build, if you do not want to build everything in the library:

```
$ make mlpack_pca mlpack_knn mlpack_kfn
```

If the build fails and you cannot figure out why, register an account on Github and submit an issue; the mlpack developers will quickly help you figure it out:

mlpack on Github

Alternately, mlpack help can be found in IRC at `#mlpack` on irc.freenode.net.

If you wish to install mlpack to `/usr/local/include/mlpack/` and `/usr/local/lib/` and `/usr/local/bin/` , once it has built, make sure you have root privileges (or write permissions to those three directories), and simply type

```
$ make install
```

You can now run the executables by name; you can link against mlpack with `-lmlpack` and the mlpack headers are found in `/usr/local/include/mlpack/` .

If running the programs (i.e. `$ mlpack_knn -h`) gives an error of the form

```
error while loading shared libraries: libmlpack.so.2: cannot open shared object file: No such file or direc
```

then be sure that the runtime linker is searching the directory where `libmlpack.so` was installed (probably `/usr/local/lib/` unless you set it manually). One way to do this, on Linux, is to ensure that the `LD_LIBRARY_PATH` environment variable has the directory that contains `libmlpack.so`. Using bash, this can be set easily:

```
export LD_LIBRARY_PATH="/usr/local/lib/:$LD_LIBRARY_PATH"
```

(or whatever directory `libmlpack.so` is installed in.)

5. Running mlpack programs

After building mlpack, the executables will reside in `build/bin/`. You can call them from there, or you can install the library and (depending on system settings) they should be added to your PATH and you can call them directly. The documentation below assumes the executables are in your PATH.

Consider the 'mlpack_knn' program, which finds the k nearest neighbors in a reference dataset of all the points in a query set. That is, we have a query and a reference dataset. For each point in the query dataset, we wish to know the k points in the reference dataset which are closest to the given query point.

Alternately, if the query and reference datasets are the same, the problem can be stated more simply: for each point in the dataset, we wish to know the k nearest points to that point.

Each mlpack program has extensive help documentation which details what the method does, what each of the parameters are, and how to use them:

```
$ mlpack_knn --help
```

Running `mlpack_knn` on one dataset (that is, the query and reference datasets are the same) and finding the 5 nearest neighbors is very simple:

```
$ mlpack_knn -r dataset.csv -n neighbors_out.csv -d distances_out.csv -k 5 -v
```

The `-v (--verbose)` flag is optional; it gives informational output. It is not unique to `mlpack_knn` but is available in all mlpack programs. Verbose output also gives timing output at the end of the program, which can be very useful.

6. Further documentation

The documentation given here is only a fraction of the available documentation for mlpack. If doxygen is installed, you can type `make doc` to build the documentation locally. Alternately, up-to-date documentation is available for older versions of mlpack:

- mlpack homepage
- Tutorials
- Development Site (Github)
- API documentation

7. Bug reporting

(see also mlpack help)

If you find a bug in mlpack or have any problems, numerous routes are available for help.

Github is used for bug tracking, and can be found at https://github.com/mlpack/mlpack/. It is easy to register an account and file a bug there, and the mlpack development team will try to quickly resolve your issue.

In addition, mailing lists are available. The mlpack discussion list is available at

mlpack discussion list

and the git commit list is available at

commit list

Lastly, the IRC channel `#mlpack` on Freenode can be used to get help.

© 2017 GitHub, Inc. Terms Privacy Security Status Help

Contact GitHub API Training Shop Blog About