Installing the development version of scikit-learn

This section introduces how to install the **master branch** of scikit-learn. This can be done by either installing a nightly build or building from source.

Installing nightly builds

The continuous integration servers of the scikit-learn project build, test and upload wheel packages for the most recent Python version on a nightly basis.

Installing a nightly build is the quickest way to:

- try a new feature that will be shipped in the next release (that is, a feature from a pull-request that was recently merged to the master branch);
- check whether a bug you encountered has been fixed since the last release.

```
pip install --pre --extra-index https://pypi.anaconda.org/scipy-wheels-nightly/simple scikit-learn
```

Building from source

Building from source is required to work on a contribution (bug fix, new feature, code or documentation improvement).

1. Use Git to check out the latest source from the scikit-learn repository on Github.:

   ```
   git clone git://github.com/scikit-learn/scikit-learn.git
   cd scikit-learn
   ```

   If you plan on submitting a pull-request, you should clone from your fork instead.

2. Install a compiler with OpenMP support for your platform. See instructions for Windows, macOS, Linux and FreeBSD.

3. Optional (but recommended): create and activate a dedicated virtualenv or conda environment.

4. Install Cython and build the project with pip in Editable mode:

   ```
   pip install cython
   pip install --verbose --editable .
   ```

5. Check that the installed scikit-learn has a version number ending with `.dev0`:

   ```
   python -c "import sklearn; sklearn.show_versions()"
   ```

6. Please refer to the Developer's Guide and Useful pytest aliases and flags to run the tests on the module of your choice.

Note: You will have to re-run the `pip install --editable .` command every time the source code of a Cython file is updated (ending in `.pyx` or `.pxd`).

Dependencies

Runtime dependencies

Scikit-learn requires the following dependencies both at build time and at runtime:

- Python (>= 3.5),
- NumPy (>= 1.11),
- SciPy (>= 0.17),
- Joblib (>= 0.11).

Those dependencies are **automatically installed by pip** if they were missing when building scikit-learn from source.

Note: For running on PyPy, PyPy3-v5.10+, Numpy 1.14.0+, and scipy 1.1.0+ are required. For PyPy, only installation instructions with pip apply.

Build dependencies

Building Scikit-learn also requires:

- Cython >= 0.28.5
- A C/C++ compiler and a matching OpenMP runtime library. See the platform system specific instructions for more details.

Note: If OpenMP is not supported by the compiler, the build will be done with OpenMP functionalities disabled. This is not recommended since it will force some estimators to run in sequential mode instead of leveraging thread-based parallelism. Setting the `SKLEARN_FAIL_NO_OPENMP` environment variable (before cythonization) will force the build to fail if OpenMP is not supported.

Since version 0.21, scikit-learn automatically detects and use the linear algebrea library used by SciPy **at runtime**. Scikit-learn has therefore no build dependency on BLAS/LAPACK implementations such as OpenBlas, Atlas, Blis or MKL.

Test dependencies

Running tests requires:

- pytest >=4.6.2

Some tests also require pandas.

Building a specific version from a tag

If you want to build a stable version, you can `git checkout <VERSION>` to get the code for that particular version, or download an zip archive of the version from github.

Editable mode

If you run the development version, it is cumbersome to reinstall the package each time you update the sources. Therefore it is recommended that you install in with the `pip install --editable .` command, which allows you to edit the code in-place. This builds the extension in place and creates a link to the development directory (see the pip docs).

This is fundamentally similar to using the command `python setup.py develop` (see the setuptool docs). It is however preferred to use pip.

On Unix-like systems, you can equivalently type `make in` from the top-level folder. Have a look at the `Makefile` for additional utilities.

Platform-specific instructions

Here are instructions to install a working C/C++ compiler with OpenMP support to build scikit-learn Cython extensions for each supported platform.

Windows

First, install Build Tools for Visual Studio 2019.

Warning: You DO NOT need to install Visual Studio 2019. You only need the “Build Tools for Visual Studio 2019”, under “All downloads” -> “Tools for Visual Studio 2019”.

Secondly, find out if you are running 64-bit or 32-bit Python. The building command depends on the architecture of the Python interpreter. You can check the architecture by running the following in `cmd` or `powershell` console:

```
python -c "import struct; print(struct.calcsize('P') * 8)"
```

For 64-bit Python, configure the build environment with:

```
SET DISTUTILS_USE_SDK=1
"C:\Program Files (x86)\Microsoft Visual Studio\2019\BuildTools\VC\Auxiliary\Build\vcvarsall.bat" x64
```

Replace `x64` by `x86` to build for 32-bit Python.

Please be aware that the path above might be different from user to user. The aim is to point to the “vcvarsall.bat” file that will set the necessary environment variables in the current command prompt.

Finally, build scikit-learn from this command prompt:

```
pip install --verbose --editable .
```

macOS

The default C compiler on macOS, Apple clang (confusingly aliased as `/usr/bin/gcc`), does not directly support OpenMP. We present two alternatives to enable OpenMP support:

- either install `conda-forge::compilers` with conda;
- or install `libomp` with Homebrew to extend the default Apple clang compiler.

macOS compilers from conda-forge

If you use the conda package manager (version >= 4.7), you can install the `compilers` meta-package from the conda-forge channel, which provides OpenMP-enabled C/C++ compilers based on the llvm toolchain.

First install the macOS command line tools:

```
xcode-select --install
```

It is recommended to use a dedicated conda environment to build scikit-learn from source:

```
conda create -n sklearn-dev python numpy scipy cython joblib pytest \
    conda-forge::compilers conda-forge::llvm-openmp
conda activate sklearn-dev
make clean
pip install --verbose --editable .
```

Note: If you get any conflicting dependency error message, try commenting out any custom conda configuration in the `$HOME/.condarc` file. In particular the `channel_priority: strict` directive is known to cause problems for this setup.

You can check that the custom compilers are properly installed from conda forge using the following command:

```
conda list compilers llvm-openmp
```

The compilers meta-package will automatically set custom environment variables:

```
echo $CC
echo $CXX
echo $CFLAGS
echo $CXXFLAGS
echo $LDFLAGS
```

They point to files and folders from your `sklearn-dev` conda environment (in particular in the bin/, include/ and lib/ subfolders). For instance `-L/path/to/conda/envs/sklearn-dev/lib` should appear in `LDFLAGS`.

In the log, you should see the compiled extension being built with the clang and clang++ compilers installed by conda with the `-fopenmp` command line flag.

macOS compilers from Homebrew

Another solution is to enable OpenMP support for the clang compiler shipped by default on macOS.

First install the macOS command line tools:

```
xcode-select --install
```

Install the Homebrew package manager for macOS.

Install the LLVM OpenMP library:

```
brew install libomp
```

Set the following environment variables:

```
export CC=/usr/bin/clang
export CXX=/usr/bin/clang++
export CPPFLAGS="$CPPFLAGS -Xpreprocessor -fopenmp"
export CFLAGS="$CFLAGS -I/usr/local/opt/libomp/include"
export CXXFLAGS="$CXXFLAGS -I/usr/local/opt/libomp/include"
export LDFLAGS="$LDFLAGS -Wl,-rpath,/usr/local/opt/libomp/lib -L/usr/local/opt/libomp/lib -lomp"
```

Finally, build scikit-learn in verbose mode (to check for the presence of the `-fopenmp` flag in the compiler commands):

```
make clean
pip install --verbose --editable .
```

Linux

Linux compilers from the system

Installing scikit-learn from source without using conda requires you to have installed the scikit-learn Python development headers and a working C/C++ compiler with OpenMP support (typically the GCC toolchain).

Install build dependencies for Debian-based operating systems, e.g. Ubuntu:

```
sudo apt-get install build-essential python3-dev python3-pip
```

then proceed as usual:

```
pip3 install cython
pip3 install --verbose --editable .
```

Cython and the pre-compiled wheels for the runtime dependencies (numpy, scipy and joblib) should automatically be installed in `$HOME/.local/lib/pythonX.Y/site-packages`. Alternatively you can run the above commands from a virtualenv or a conda environment to get full isolation from the Python packages installed via the system packager. When using an isolated environment, `pip3` should be replaced by `pip` in the above commands.

When precompiled wheels of the runtime dependencies are not avalaible for your architecture (e.g. ARM), you can install the system versions:

```
sudo apt-get install cython3 python3-numpy python3-scipy
```

On Red Hat and clones (e.g. CentOS), install the dependencies using:

```
sudo yum -y install gcc gcc-c++ python3-devel numpy scipy
```

Linux compilers from conda-forge

Alternatively, install a recent version of the GNU C Compiler toolchain (GCC) in the user folder using conda:

```
conda create -n sklearn-dev numpy scipy joblib cython conda-forge::compilers
conda activate sklearn-dev
pip install --verbose --editable .
```

FreeBSD

The clang compiler included in FreeBSD 12.0 and 11.2 base systems does not include OpenMP support. You need to install the `openmp` library from packages (or ports):

```
sudo pkg install openmp
```

This will install header files in `/usr/local/include` and libs in `/usr/local/lib`. Since these directories are not searched by default, you can set the environment variables to these locations:

```
export CFLAGS="$CFLAGS -I/usr/local/include"
export CXXFLAGS="$CXXFLAGS -I/usr/local/include"
export LDFLAGS="$LDFLAGS -Wl,-rpath,/usr/local/lib -L/usr/local/lib -lomp"
```

Finally, build the package using the standard command:

```
pip install --verbose --editable .
```

For the upcoming FreeBSD 12.1 and 11.3 versions, OpenMP will be included in the base system and these steps will not be necessary.

Toggle Menu