scitime Documentation

Release 0.0.1

Nathan Toubiana, Gabriel Lerner

Mar 27, 2021

Contents:

Scitime is a python package that gives runtime estimations of scikit-learn algorithms

CHAPTER 1

Scitime Quick Start

Scitime currently supports:

- RandomForestClassifier
- SVC
- KMeans
- RandomForestRegressor

1.1 Installation

To install:

```
pip install scitime
```

Or:

```
conda install -c conda-forge scitime
```

1.2 Usage

Example of getting runtime estimation for KMeans:

```
from sklearn.cluster import KMeans
import numpy as np
import time

from scitime import RuntimeEstimator

# example for kmeans clustering
```

(continues on next page)

(continued from previous page)

```
estimator = RuntimeEstimator(meta_algo='RF', verbose=3)
km = KMeans()

# generating inputs for this example
X = np.random.rand(100000,10)
# run the estimation
estimation, lower_bound, upper_bound = estimator.time(km, X)
```

1.3 How Scitime works

Scitime predicts the runtime to fit by pre-trained runtime estimators, we call it meta algorithm (meta_algo), whose weights are stored in a dedicated pickle file in the package metadata. For each Scikit Learn model (if supported), you will find a corresponding meta algo pickle file in Scitime's code base. You can also train your own meta algorithm using your own generated data (see section "Use _data.py to generate data" below). More information about how the models are pre-trained can be found *here*.

1.4 Local Testing

Inside virtualenv (with pytest>=3.2.1):

```
python -m pytest
```

1.5 Use _data.py to generate data

(for contributors)

```
$ python _data.py --help

usage: _data.py [-h] [--drop_rate DROP_RATE] [--meta_algo {RF,NN}]
                [--verbose VERBOSE]
                [--algo {RandomForestRegressor,RandomForestClassifier,SVC,KMeans}]
                [--generate_data] [--fit FIT] [--save]

Gather & Persist Data of model training runtimes

optional arguments:
  -h, --help            show this help message and exit
  --drop_rate DROP_RATE
                        drop rate of number of data generated (from all param
                        combinations taken from _config.json). Default is
                        0.999
  --meta_algo {RF,NN}   meta algo used to fit the meta model (NN or RF) -
                        default is RF
  --verbose VERBOSE     verbose mode (0, 1, 2 or 3)
  --algo {RandomForestRegressor,RandomForestClassifier,SVC,KMeans}
                        algo to train data on
  --generate_data       do you want to generate & write data in a dedicated
                        csv?
  --fit FIT             do you want to fit the model? If so indicate the csv
```

(continues on next page)

(continued from previous page)

```
                        name
  --save                (only used for model fit) do you want to save /
                        overwrite the meta model from this fit?
```

1.6 Contribute

The preferred way to contribute to scitime is to fork the main repository on GitHub, then submit a "pull request" (PR) - as done for scikit-learn contributions:

- Create an account on GitHub if you do not already have one.
- Fork the project repository: click on the 'Fork' button near the top of the page. This creates a copy of the code under your account on the GitHub user account.
- Clone your fork of the scitime repo from your GitHub account to your local disk:

```
git clone git@github.com:YourLogin/scitime.git
cd scitime
# Install library in editable mode:
pip install --editable .
```

- Create a branch to hold your development changes:

```
git checkout -b my-feature
```

and start making changes. Always use a feature branch. It's good practice to never work on the masterbranch!

- Develop the feature on your feature branch on your computer, using Git to do the version control. When you're done editing, add changed files using git add and then git commit files:

```
git add modified_files
git commit
```

- to record your changes in Git, then push the changes to your GitHub account with:

```
git push -u origin my-feature
```

- Follow GitHub instructions to create a pull request from your fork.

Some quick additional notes: - We use appveyor and travis.ci for our tests - We try to follow the PEP8 guidelines (using flake8, ignoring codes E501 and F401)

CHAPTER 2

Estimator (main class)

class scitime.**RuntimeEstimator**(*meta_algo='RF'*, *verbose=3*, *confidence=0.95*)

Bases: scitime._log.LogMixin

Estimator class arguments

Parameters

- **meta_algo** – meta algorithm (RF or NN)
- **verbose** – log output (0, 1, 2 or 3)
- **confidence** – confidence for intervals

time(*algo*, *X*, *y=None*)

predicts training runtime for a given training

Parameters

- **algo** – algo whose runtime the user wants to predict
- **X** – np.array of inputs to be trained
- **y** – np.array of outputs to be trained (set to None is unsupervised algo)

Returns predicted runtime, low and high values of the confidence interval

Return type float

CHAPTER 3

Model (for contributors)

class scitime._model.**RuntimeModelBuilder**(*drop_rate=0.9*, *meta_algo='RF'*, *algo='RandomForestRegressor'*, *verbose=0*, *bins=None*)

Bases: scitime.estimate.RuntimeEstimator, scitime._log.LogMixin

Model class arguments

Parameters

- **drop_rate** – drop rate over generating data loop
- **meta_algo** – meta algorithm (RF or NN)
- **algo** – algo chosen for generating data / fitting meta model
- **verbose** – log output (0, 1, 2 or 3)

model_fit(***kw*)

builds the actual training time estimator (currently we only support NN or RF) the data is either generated from scratch or taken as input if specified, the meta algo is saved as a pkl file along with associated metadata (column names, mse per bin)

Parameters

- **generate_data** – bool (if set to True, calls _generate_data)
- **inputs** – pd.DataFrame chosen as input
- **outputs** – pd.DataFrame chosen as output
- **csv_name** – name if csv in case we fetch data from csv
- **save_model** – boolean set to True if the model needs to be saved
- **meta_algo_params** – params of the meta algo
- **compress** – value between 1 and 9 to compress the pkl model (the higher the more compressed)

Returns meta_algo

Return type scikit learn model

model_validate(**kw)

measures training runtimes and compares to actual runtimes once the model has been trained

Returns results dataframe and error rate

Return type pd.DataFrame and float

params

retrieves the estimated algorithm's parameters if the algo is supported else, return KeyError

Returns dictionary

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Index

M

P

R

T