AI2ES Coding Standards

Group Lead: David John Gagne

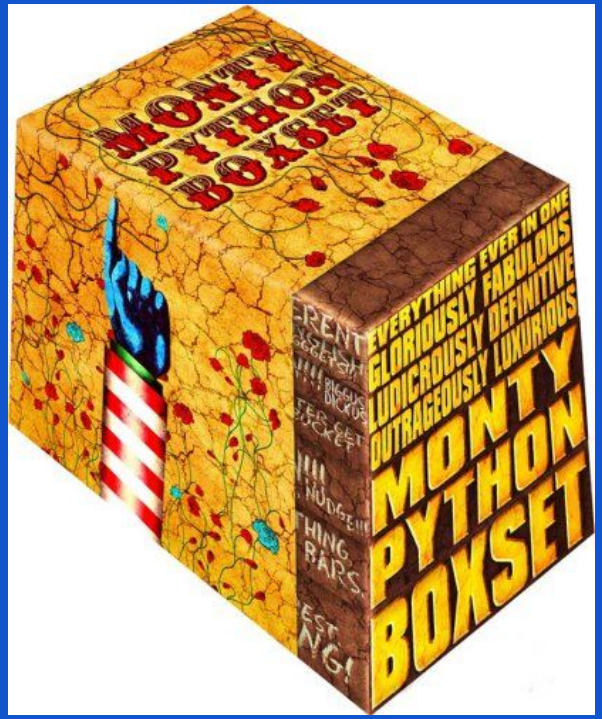
December 16, 2020

Introduction

- AI2ES members will be developing software libraries collaboratively across multiple groups and institutions
- Ideally we want to share software across the institute and release packages to the public
- Coding standards will enable us to encourage/enforce a level of software quality across all shared repositories
- However, some standards are easier to implement and enforce than others
- Goals:
  - Discuss potential types of standards we should aim to encourage across the institute
  - What is necessary vs. nice to have vs. overly burdensome?

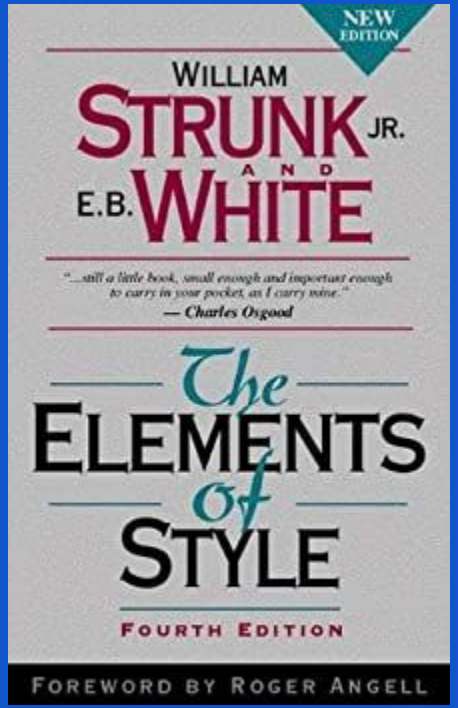
Python Packaging Structure

- package-name/
  - README.md: Contains description of package, installation and use instructions
  - setup.py: Script for installing the package
  - LICENSE: License text (CC0)
  - environment.yml: contains list of dependencies for the conda installer
  - requirements.txt: list of packages for pip installer
  - package/: Directory containing all python module files (.py)
    - test/: Contains all unit test files
  - doc/: Documentation directory
  - scripts/: Contains helper scripts and executable programs
  - notebooks/: Contains jupyter notebooks



Source: Amazon

Coding Style



- Goal: ensure that all code follows the same formatting conventions for a consistent look and meaning across packages
- Python style guide: PEP8
- Major style areas
  - Variable naming convention: instance_or_function, ClassName
  - Equation spacing: c = a + b not c=a+b
  - Whitespace: indents are 4 spaces
- Style can be checked and corrected with programs like PyCharm or with linter programs

Version Control

https://xkcd.com/1597/

- Version control: software that keeps track of changes to files and merges changed files together
- Git: distributed version control software
- Github: website that stores git repositories in a central location and provides project management and organization tools
- Why use version control:
  - Keep track of changes in case you make a mistake and need to recover old code
  - Synchronize changes across multiple computers (edit code on laptop and sync with supercomputer)
  - Merge changes from different collaborators
  - Work on new ideas in different branches of same repository

Testing

- Code should be tested to ensure it works properly and to catch changes the break existing functionality
- Types of tests
  - Unit tests: check functionality of single component
  - Integration tests: ensures components work together properly
- Testing framework: pytest
- Challenges
  - Writing good tests can be challenging
  - Needing data to test ML/data loading
  - Tests can’t cover all ways things go wrong

```python
def test_MixedConvNet():
    scalar_input_shape = 10
    conv_input_shape = (96, 96, 12)
    scn_norm = MixedConvNet(min_filters=16, output_type="gaussian", data_format="channels_last",
                            loss="crps_norm", pooling_width=2)
    scn_norm.build_network(scalar_input_shape, conv_input_shape, 1)
    scn_norm.compile_model()
    print(scn_norm.model_.summary())
    assert scn_norm.model_.input[0].get_shape()[1] == scalar_input_shape
    assert scn_norm.model_.input[1].get_shape()[1] == conv_input_shape[0]
    assert scn_norm.model_.output.get_shape()[1] == 2
```

Code Review

- Different collaborators should work on different branches while implementing new features
- When ready to share new code with everyone else, the developer should request a code review by the leads for that package
- Code review should accomplish the following items:
  - Verify the code works
  - Check the style
  - Identify areas of confusion or unclear functionality
  - Identify potential performance bottlenecks
- Art of Giving and Receiving Code Reviews Gracefully: https://www.alexandra-hill.com/2018/06/25/the-art-of-giving-and-receiving-code-reviews/
-

Pull Requests

Installation and processing changes #20

Edit | Open with

Merged djgagne merged 10 commits into `master` from `djgagne` 2 days ago

Conversation 6 | Commits 10 | Checks 0 | Files changed 10 | +394 −179

djgagne commented 13 days ago

I updated the repo with a few key changes:

- added an environment.yml file to install all dependencies with conda and a requirement file for pip.
- changed processed variable names for Idaho so they are consistent with Cabauw
- added a training config file with updated variable names for FastEddy.
- copied processed datafiles to glade at `/glade/p/cisl/aiml/fasteddy`

djgagne added 5 commits on Oct 29

Added example.cu	bc9bf88
Merge branch 'master' of github.com:NCAR/mlsurfacelayer into djgagne	d9ee3cf
Added environment.yml and requirements files	340a0cc
Minor update to setup.py	7daa5a9
Added new training file and fixed Idaho derived units.	083ea11

Reviewers
- jsauer-NCAR ✓
- charlie-becker ✓

Assignees
No one—assign yourself

Labels
None yet

Projects
None yet

Milestone
No milestone

Linked issues
Successfully merging this pull request may close these issues.

Continuous Integration

- Automated scripts that run whenever new changes are pushed to github
- Functions
  - Install all dependencies from scratch
  - Run test suite
  - Run test function
  - Check style, test coverage
  - Upload to package repository if everything passes
- Frameworks
  - TravisCI
  - CircleCI
  - Github Actions
- Benefits
  - Automatically runs after commits and pull requests
  - Can test multiple configurations of package
  - Catches breaking changes throughout pipeline
  - Emails you if something is broken
- Drawbacks
  - Requires moderate effort for initial setup
  - Only as good as the tests are
  - Can cost money for private repos or if usage quota exceeded

Documentation

- Code should be documented so people know how to use it properly and how it works
- Levels of documentation
  - Docstrings: at beginning of function that describe purpose of function, inputs, outputs, and a usage example
  - Inline comments: Describe how a section of code works or why it is used
  - Tutorial: Describes how to use package through a step-by-step guide
  - Narrative documentation: describes motivation for code, history, science, broader context

API vs. Narrative Documentation

3.2.4.3.4. sklearn.ensemble.ExtraTreesRegressor

class sklearn.ensemble.**ExtraTreesRegressor**(*n_estimators=100, *, criterion='mse', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, bootstrap=False, oob_score=False, n_jobs=None, random_state=None, verbose=0, warm_start=False, ccp_alpha=0.0, max_samples=None*) [source]

An extra-trees regressor.

This class implements a meta estimator that fits a number of randomized decision trees (a.k.a. extra-trees) on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

Read more in the User Guide.

Parameters:

n_estimators : *int, default=100*
The number of trees in the forest.

Changed in version 0.22: The default value of `n_estimators` changed from 10 to 100 in 0.22.

criterion : *{"mse", "mae"}, default="mse"*
The function to measure the quality of a split. Supported criteria are "mse" for the mean squared error, which is equal to variance reduction as feature selection criterion, and "mae" for the mean absolute error.

New in version 0.18: Mean Absolute Error (MAE) criterion.

max_depth : *int, default=None*
The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

min_samples_split : *int or float, default=2*
The minimum number of samples required to split an internal node:

- If int, then consider `min_samples_split` as the minimum number.
- If float, then `min_samples_split` is a fraction and `ceil(min_samples_split * n_samples)` are the minimum number of samples for each split.

Changed in version 0.18: Added float values for fractions.

min_samples_leaf : *int or float, default=1*
The minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least `min_samples_leaf` training samples in each of the left and right branches. This may have the effect of smoothing the model, especially in regression.

- If int, then consider `min_samples_leaf` as the minimum number.
- If float, then `min_samples_leaf` is a fraction and `ceil(min_samples_leaf * n_samples)` are the minimum number of samples for each node.

1.11.2.2. Extremely Randomized Trees

In extremely randomized trees (see `ExtraTreesClassifier` and `ExtraTreesRegressor` classes), randomness goes one step further in the way splits are computed. As in random forests, a random subset of candidate features is used, but instead of looking for the most discriminative thresholds, thresholds are drawn at random for each candidate feature and the best of these randomly-generated thresholds is picked as the splitting rule. This usually allows to reduce the variance of the model a bit more, at the expense of a slightly greater increase in bias:

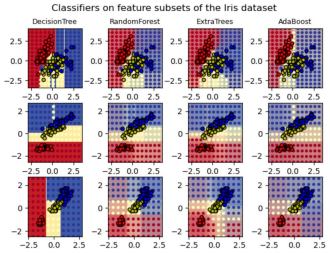
```
>>> from sklearn.model_selection import cross_val_score
>>> from sklearn.datasets import make_blobs
>>> from sklearn.ensemble import RandomForestClassifier
>>> from sklearn.ensemble import ExtraTreesClassifier
>>> from sklearn.tree import DecisionTreeClassifier

>>> X, y = make_blobs(n_samples=10000, n_features=10, centers=100,
...     random_state=0)

>>> clf = DecisionTreeClassifier(max_depth=None, min_samples_split=2,
...     random_state=0)
>>> scores = cross_val_score(clf, X, y, cv=5)
>>> scores.mean()
0.98...

>>> clf = RandomForestClassifier(n_estimators=10, max_depth=None,
...     min_samples_split=2, random_state=0)
>>> scores = cross_val_score(clf, X, y, cv=5)
>>> scores.mean()
0.999...

>>> clf = ExtraTreesClassifier(n_estimators=10, max_depth=None,
...     min_samples_split=2, random_state=0)
>>> scores = cross_val_score(clf, X, y, cv=5)
>>> scores.mean() > 0.999
True
```

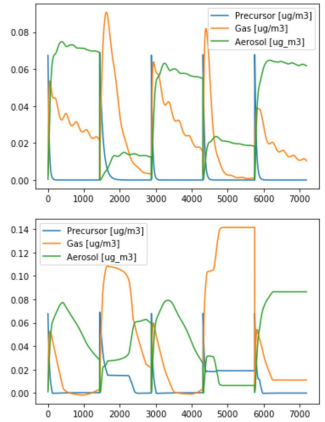


Jupyter Notebooks

```
In [14]: print('Metrics for Box Emulator:')
         evaluate_mod(true_box, pred_box)

         ## Quick plot to see if emulator is capturing the patterns
         true_box.iloc[:,0:3].plot()
         pred_box.iloc[:,0:3].plot()

         Metrics for Box Emulator:
         RMSE: Precursor: 0.00946, Gas: 0.06098, Aerosols: 0.02118
         R2: Precursor: 0.53770, Gas: 0.00260, Aerosols: 0.47804
         Hellenger Distance: Precursor: 0.11655, Gas: 0.21057, Aerosols: 0.33689
Out[14]: <matplotlib.axes._subplots.AxesSubplot at 0x2b46f7601a50>
```



As you can see, it is relatively easy to train the base network to predict t+1 values. It is far more challenging to model the length of the experiment (~1440 timesteps). Furthermore, you may discover that better performance on the neural network may not mean better performance with the box emulator.

Some potential ideas for better emulator performance: adding some noise to the training data to prevent overfitting, or using a recurrent neural network/LSTM to utilize more than one timestep to inform prediction of the next.

Model Type	**Metric**	**Variable**		
Baseline DNN		Precursor	Gas	Aerosols
	RMSE:	**0.00019**	**0.00035**	**0.00014**

- Interactive coding and visualization interface
- Benefits
  - Load data and interact with it on multiple computing platforms
  - Merge docs and code together
  - Great for tutorials
  - Can run locally, HPC, cloud
  - Can convert notebooks to packages (nbdev)
- Drawbacks
  - Can encourage spaghetti code
  - Errors caused by order of running cells
  - Need Python environment setup correctly to work

Challenges

- Participation in code review
  - Less experienced people can be intimidated from commenting on pull requests
  - Need to encourage comments and have a positive environment for commenting
  - Avoid gatekeeping behavior or overly harsh criticism
- Documentation
  - Always needed but can be tedious to write
  - Need feedback on documentation priorities
  - Documentation needs can be quickly evident by having a beginner try to use the code
- Teaching Coding Standards
  - Can point everyone to tutorials
  - People need to get in habit of practicing tasks
- Changing Standards
  - Software recommendation and fashions change with time
  - Balancing consistent guidance with adapting to new effective practices

Summary and Questions

- More detailed documents in AI2ES coding standards working group folder
- Add practices to this document
- Questions:
  - What other coding practices should we use?
  - What is essential, nice to have, or overly burdensome?
  - How to incorporate science workflow and priorities into coding process?
  - Who wants to join the group?
- Email me: dgagne@ucar.edu