The guide to Pycle (**Py**thon **C**ompresive **Le**arning toolbox)

Vincent Schellekens

November 28, 2019

Abstract

This is the guide to Pycle, a toolbox for Compressive Learning. It is structured as follows: first we shortly explain the theoretical methods this toolbox implements. Then, we explain how the toolbox is structured, and the main steps that a user should follow to use it. The detailed documentation of all the functionalities in the toolbox is then provided, followed by some practical examples to get started easily.

1 What is Compressive Learning?

$$\begin{array}{ccccc} X & \longrightarrow & \boldsymbol{z}_X & \longrightarrow & \Theta \\ \text{Dataset} & & \text{Sketch} & & \text{Parameters} \end{array}$$

Figure 1: Compressive learning .

$$\boldsymbol{z}_X := \frac{1}{n} \sum_{i=1}^{n} \Phi(\boldsymbol{x}_i) \tag{1}$$

See [1] for a complete introduction to compressive learning.

2 An overview of Pycle

2.1 Requirements

The Pycle package builds on a set of standard Python libraries, that are required to run it:

- `numpy`
- `scipy`
- `matplotlib`

2.2 Typical workflow

A typical use of Pycle follows the following steps:

1. Design a sketch operator, then sketch the dataset using the `sketching.py` module.

2. Extract a model from the sketch by a compressive learning method contained in the `compressive_learning.py` module.

$$X \longrightarrow \boldsymbol{z}_X \longrightarrow \Theta$$

Figure 2: Flowchart of a typical compressive learning execution with Pycle.

3 A tutorial tour of `pycle`

3.1 Sketching

To use the `sketching` submodule, you first need to import it (I personally like `sk` as shorthand). Usual sketching as defined in (1) can then be done by calling `sk.computeSketch` as follows.

```
import pycle.sketching as sk

X = ...   # load a numpy array of dimension (n,d)
Phi = ... # sketch feature map, see later

z = sk.computeSketch(X,Phi)
```

As you might have guessed, `sk.computeSketch(dataset,featureMap)` requires two arguments, the dataset encoded as a numpy array, and the feature map Φ.

Note: Dataset representation conventions. In `pycle`, we follow the mainstream convention for datasets, where a collection of a vectors in dimension b is encoded as a numpy array of size (a, b). Thus, the dataset $X = (\boldsymbol{x}_i \in \mathbb{R}^d)_{i=1}^n$ is a (n, d) numpy array (even when $d = 1$) which means that `X[0]` references $\boldsymbol{x_1}$. Similarly, the projection matrix $\Omega = (\boldsymbol{\omega}_j \in \mathbb{R}^d)_{j=1}^m$ is a (m, d) array, etc.

The feature map argument can be specified in one of the two following ways. Either you give an instance of a `FeatureMap` object, which we explain below, or you directly provide a custom callable function (e.g., to compute the second-order moments for the sketch, write `Phi = lambda x: x**2`).

Note: FeatureMap objects. At this point, you might be wondering why we bother to construct a `FeatureMap` object instead of a function to represent... well, a (mathematical) function, namely Φ. The reason is that actual learning algorithms (from `pycle.compressive_learning`) or even other sketching functionalities (e.g., privacy preservation) require not just the ability to evaluate Φ but other information about it (e.g., its target dimension m, its jacobian $\nabla\Phi$, whether it belongs to a set of standard maps etc.). Those informations are packaged in the `FeatureMap` object for your convenience.

3.2 Learning

3.3 Utilities

4 Documentation

4.1 Sketching methods

4.2 Learning tools

4.3 Utilities

5 Examples

References

[1] R. Gribonval, G. Blanchard, N. Keriven, and Y. Traonmilin, "Compressive statistical learning with random feature moments," *arXiv preprint arXiv:1706.07180*, 2017.