Final Project, STAT 243

Aparimit Kasliwal, Treves Li, Yuyang Wu

1. Functional Programming Implementation of `ARS`

Our implementation of Adaptive Rejection Sampling, based on the `Gilkis et.al.` is based on the following modular functions;

(a) `construct_envelope` function: Here, we

(b) `update_envelope` function: Here, we

(c) `construct_squeezing` function: Here, we

(d) `update_squeezing` function: Here, we

(e) `calculate_piecewise_linear` function: Here, we

(f) `sample_piecewise_linear` function: Here, we

(g) `adaptive_search_domain` function: Here, we

(h) `init_points` function: Here, we

(i) `check_overflow_underflow` function: Here, we

(j) `h_log` function: Here, we

(k) `h_cached` function: Here, we

(l) `is_log_concave` function: Here, we

(m) `compare_samples_to_distribution` function: Here, we

(n) `ars` function: Here, we

2. File Structure and Modules

We propose the following file structure, into which we collapse all the functions defined above. As visualized below, our main functional code is present in the `ars` directory within the `gradients.py`, `sampler.py`, `utils.py` and `validation.py` files.

On the other hand, we have a directory called `tests` which contains different files focusing on various aspects of testing our code - all of which can be executed using `pytest` with the following command:

```
pytest tests/.
```

This project is organized as follows:

```
  README.md                      # Project overview and instructions
  ars                            # Main ARS package
      __init__.py                # Package initializer
      gradients.py               # Gradient computation utilities
      sampler.py                 # ARS sampling implementation
      utils.py                   # General utility functions
      validation.py              # Validation functions for ARS
  debug_and_compare.ipynb        # Jupyter notebook for debugging and comparison
  requirements.txt               # Python dependencies for the project
  tests                          # Unit tests for the project
      test_sampler.py            # Tests for sampler module
      test_utils.py              # Tests for utils module
      test_validation.py         # Tests for validation module
```

3. Installation of the package and code exceution

We've included specific files called `requirements.txt` and `setup.py` which allow transforming our developed code into an installable package through the following command;

```
pip install .
```

4. Testing

---

5. Statement of Contribution