Reproducibility README for 'Adaptive Debiased Lasso in High-dimensional GLMs with Streaming Data'

This repository contains the source code for the **Approximated Debiased Lasso (ADL)** algorithm, designed for online statistical inference in high-dimensional generalized linear models (GLMs) with streaming data. The algorithm is particularly useful for scenarios where data arrives sequentially, and efficient, real-time inference is required.

Repository Structure

The repository is organized as follows:

Demonstrations for Reproducing Numerical Results

Below is a list of execution files for reproducing numerical results of ADL presented in **Table 2**, **Table 3**, **Figure 3** in the main text, and **Table S.1 - Table S.4** in the supplementary. For detailed procedures, please refer to **Section 4** of the main text and **Section S.3** of the supplementary material. Since we provide numerous simulations in Section S.3 of the supplementary material, we have included the first four simulations in this repository to faciliate easier understanding and usage. These configurations are representative of the broader set of settings thereafter.

- [run_table2.py]: This execution file contains the codes necessary to reproduce the column "ADL" of Table 2 in the main text. This simulation is conducted over 500 replications with $n = 200, p = 500, s_0 = 6, \mathbf{\Sigma} = 0.1 \times \{0.5^{|i-j|}\}_{i,j=1,\ldots,p}$. The confidence intervals are constructed for three randomly selected parameters from each category of $\boldsymbol{\beta}^*$.

- [run_table3.py]: This execution file contains the codes necessary to reproduce the column "ADL" of Table 3 in the main text. This simulation is conducted over 500 replications with $n = 200, p = 500, s_0 = 6, \mathbf{\Sigma} = \{0.5^{|i-j|}\}_{i,j=1,\ldots,p}$. The confidence intervals are constructed for three randomly selected parameters from each category of $\boldsymbol{\beta}^*$.

- [run_figure3.py]: This execution file contains the codes necessary to reproduce Figure 3 in the main text. In this demonstration, the dimension p is raised to 20000 with $n = 1000$, $s_0 = 20$, $\mathbf{\Sigma} = \{0.5^{|i-j|}\}_{i,j=1,\ldots,p}$. The confidence intervals are constructed for three randomly selected parameters from each category of $\boldsymbol{\beta}^*$.

- [run_tableS.1.py]: This execution file contains the codes necessary to reproduce the column "ADL" of Table S.1 (independent covariates) in the supplementary material. This simulation is conducted over 500 replications with $n = 200, p = 500, s_0 = 6, \mathbf{\Sigma} = \boldsymbol{I}$. The support set S is randomly generated, with three elements equal to 1 and another three equal to -1.

- [run_tableS.2.py]: This execution file contains the codes necessary to reproduce the column "ADL" of Table S.2 (highly-correlated covariates case) in the supplementary material. This simulation is conducted over 500 replications with $n = 200, p = 500, s_0 = 6, \mathbf{\Sigma} = \{0.8^{|i-j|}\}_{i,j=1,\ldots,p}$. The support set S is randomly generated, with three elements equal to 1 and another three equal to -1.

- [run_tableS.3.py]: This execution file contains the codes necessary to reproduce the column "ADL" of Table S.3 (block-wise Toeplitz correlated covariates case) in the supplementary material. This simulation is conducted over 500 replications with $n = 200, p = 500, s_0 = 6$, and $\mathbf{\Sigma}$ is one block-wise Toeplitz covariance matrix . The support set S is randomly generated, with three elements equal to 0.5 and another three equal to -0.5.

- [run_tableS.4.py]: This execution file contains the codes necessary to reproduce the column "ADL" of Table S.4 (high collinearity case) in the supplementary material. This simulation is conducted over 500 replications with $n = 200, p = 500, s_0 = 6$, and $\mathbf{\Sigma}$ is one block-wise Toeplitz covariance matrix. The support set $S = \{1, 2, \cdots, 6\}$. Specifically, $\boldsymbol{\beta}_1 = \boldsymbol{\beta}_3 = \boldsymbol{\beta}_5 = 0.5$ and $\boldsymbol{\beta}_2 = \boldsymbol{\beta}_4 = \boldsymbol{\beta}_6 = -0.5$。

Real Data Example

- [run_realdata.py]: Execution file for real data example. Please read through **Real Data Example** section in this README file for detailed procedures.

Algorithm Core Functions

- [adl.py]: This file implements the **Approximated Debiased Lasso (ADL) algorithm**, the main method proposed in the paper for online statistical inference in high-dimensional GLMs.

- [adl_realdata.py]: This file implements the ADL algorithm for real data analysis, which is compatible with sparse arrays.

- [radar.py]: This file contains the implementation of the **Regularization Annealed Epoch Dual Averaging (RADAR)** and **Adaptive RADAR algorithms**, which are core components of the ADL algorithm.

Helper Functions

- [cal.py]: This file contains utility functions for generating synthetic data, calculating summaries, and visualizing results.

- [process.py]: This script processes the raw dataset (`combined_data.csv`) to extract uni-gram and bi-gram features from text data, and transforming it into a sparse matrix format for efficient storage and analysis.

Dependencies

To reproduce the numerical results, we suggest install the the following dependencies in your Python environment:

- `numpy` , version 2.2
- `scipy` , version 1.15
- `matplotlib` , version 3.10
- `pandas` , version 2.2.3

Reproducing Numerical Results

To run the simulations included in this repository, execute the following files:

```
python run_table2.py
python run_table3.py
python run_figure3.py
python run_tableS.1.py
python run_tableS.2.py
python run_tableS.3.py
python run_tableS.4.py
```

For each simulation, the online estimates, confidence intervals, and trace plots will be saved in a dedicated folder. For example, the results for Table 2 will be saved in the folder `./table2`. For the ease of presentation, the summaries required for reporting in the manuscript are saved in a text file named `summary.txt` within the corresponding folder.

Real Data Example

Dataset

For the real data example, the dataset `combined_data.csv` is required. This dataset can be downloaded from:

- Email Spam Classification Dataset on Kaggle

To extract uni-gram and bi-gram features from the raw data, ensure that the raw dataset is placed in the root directory of the repository before running the real data analysis scripts. Then, execute the following command:

```
python process.py
```

The processed data will be saved as sparse matrices in `bigram_X.npz` and `bigram_y.npz` for further analysis. For convenience and to facilitate an easier walkthrough of the code, we have included these two processed data files in the repository. Users may skip the feature extraction step and proceed directly to online inference if desired.

As described in Section 5 of the main text, we selected three terms of interest: “investment”, “schedule”, and “per cent” for statistical inference. These terms correspond to feature indices 6795, 7856, and 22608, respectively. Users can specify which feature to analyze by modifying line 13 of the script file `run_realdata.py` .py. The trace plot and test prediction error will be saved in a folder, for example, `./realdata_result/feature6795`. Online estimates and confidence intervals will also be saved in the corresponding folder. To conduct online statistical inference on the processed data, run the following script:

```
python run_realdata.py
```