Reproducibility checklist

A clear description of the mathematical setting, algorithm, and/or model: provided in Section 1.

An analysis of the complexity (time, space, sample size) of any algorithm. Theoretical Section 2 and empirical Section 4.

A link to a downloadable source code, with specification of all dependencies, including external libraries. In Section 6.

For any theoretical claim, check if you include:

A statement of the result: Theorems 1–2.

A clear explanation of any assumptions: provided before/in Theorems 1–2.

A complete proof of the claim: provided immediately after Theorems 1–2.

For all figures and tables that present empirical results, check if you include:

A complete description of the data collection process, including sample size. A complete description of the data collection process, including sample size. Timing Figures 2–3 data collection described in Sections 2–4; sample size (5) described in captions. Accuracy Figure 4 data collection described in Section 5; sample size (4) also described in y axis label.

A link to a downloadable version of the dataset or simulation environment. Neuroblastoma data were used in Figure 2: `https://cran.r-project.org/package=neuroblastoma` UCI chipseq data were used in Figures 3–4: `https://archive.ics.uci.edu/ml/datasets/chipseq#` Loss values in large data set used for timings (Figure 3) came from `https://rcdata.nau.edu/genomic-ml/fullpath/db-loss.tsv` Processed versions of UCI chipseq data used for accuracy analysis (Figure 4) available from `https://github.com/tdhock/feature-learning-benchmark` e.g. `https://github.com/tdhock/feature-learning-benchmark/blob/master/labeled_problems_targets.csv`

An explanation of any data that were excluded, description of any pre-processing step. The proposed algorithm works on optimal loss values L_t, which were computed from the raw data using the following algorithms. `jointseg::Fpsn` R/C++ implementation used for PDPA algorithm (Section 4), `PeakSegDisk::PeakSegFPOP_disk` R/C++ implementation used to compute constrained changepoint models for UCI chipseq data (Figures 3–Figure 4).

An explanation of how samples were allocated for training / validation / testing. In Section 5 we use 4-fold cross-validation. For each fold the corresponding train data were passed to the L1-regularized linear learning algorithm (`penaltyLearning::IntervalRegressionCV` function in R), which uses internally generated train/validation splits to select the optimal degree of L1-regularization. The resulting model was used for predictions on the test set in the 4-fold cross-validation.

The range of hyper-parameters considered, method to select the best hyper-parameter configuration, and specification of all hyper-parameters used to generate results. For the approximate grid search algorithm in section 5, we used a log-scale grid of G penalty values $\lambda \in \{10^{-15}, \ldots, 10^{22}\}$, where $G \in \{2^1, \ldots, 2^{10} = 1024\}$.

The exact number of evaluation runs. 5 timings, 4 CV folds as mentioned above.

A description of how experiments were run. Provided in Sections 2–4.

A clear definition of the specific measure or statistics used to report results. The specific accuracy measure reported in Section 5 (Figure 4) is the number/percent of correctly predicted labels, also known as the zero-one loss.

Clearly defined error bars. Error bars are mean $\pm$ SD over timings/folds, as explained in caption of Figure 3 and axis label of Figure 4.

A description of results with central tendency (e.g. mean) & variation (e.g. stddev). Provided in figures 3–4.

A description of the computing infrastructure used. Provided in Section 4.