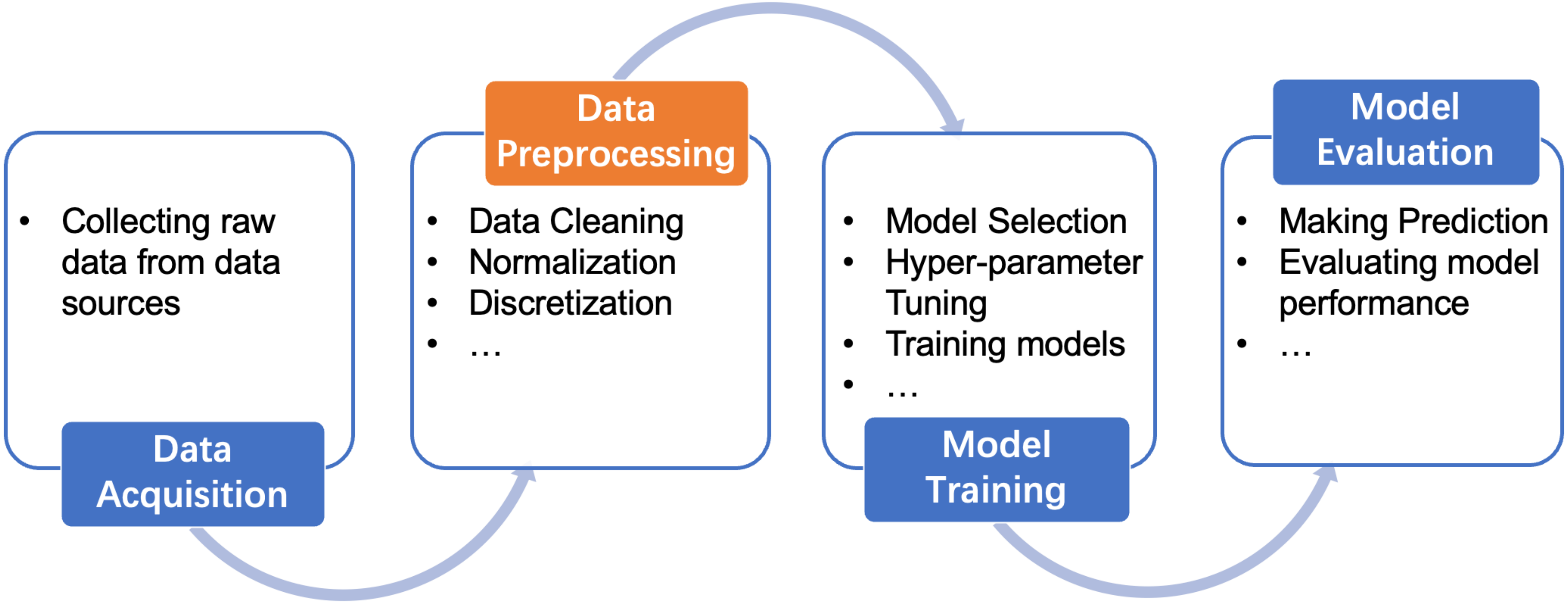
DiffPrep: Differentiable Data Preprocessing Pipeline Search for Learning over Tabular Data

Peng Li, Zhiyi Chen, Xu Chu, Kexin Rong
Georgia Institute of Technology

SIGMOD 2023

Data preprocessing is an essential step in ML

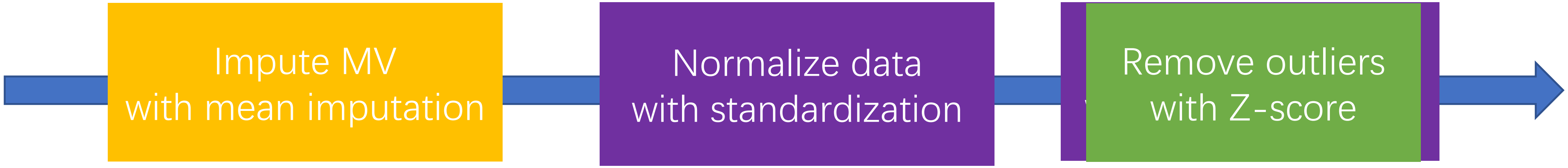
- Raw data collected from data sources can rarely be used directly by ML models due to the existence of data issues (e.g., data errors, different feature scales).



A Typical ML Workflow

Designing a data preprocessing pipeline is hard

- **Data Preprocessing Pipeline** is a sequence of operators, where each operator tackles one specific data issue.



An example data preprocessing pipeline

- **Complex design decisions:**
  - **Types:** which type of transformations to use?
  - **Operators:** which operator to use for each transformation?
  - **Order:** which order of transformations to use?
  - **Feature-wise:** different features may need different pipelines.

Limitations in Existing Methods

- **ML Developers**
  - Use a default pipeline or trial-and-error methods.
- **Traditional Data Cleaning Work**
  - Design pipelines that optimize data quality independently of ML.
  - Data quality may not be accessible, and it may not lead to the optimal ML performance.
- **Existing AutoML Systems**
  - Limited search space.
  - Train model multiple times.

Systems	Operator	Types	Order	Feature-wise	Optimization Method
H2O	×	×	×	×	Random Search
Azure	✓	×	×	×	Bayesian Optimization
Auto-Sklearn	✓	✓	×	×	Bayesian Optimization
Learn2Clean	✓	✓	✓	×	Q-Learning
DiffPrep-Fix	✓	✓	×	✓	**Bi-level Optimization with Gradient Descent**
DiffPrep-Flex	✓	✓	✓	✓	

Problem Definition

- **Goal:** automatically and efficiently select a data preprocessing pipeline from the search space such that the model performance (validation accuracy) is maximized.

Outer level: $$\underset{pipline}{\operatorname{argmin}} Loss\ (D_{val}, pipeline, model^*)$$

Inner level: $$s.t.\ model^* = \underset{model}{\operatorname{argmin}} Loss\ (D_{train}, pipeline, model)$$

"Bi-level Optimization"

- **Compared with existing AutoML systems:**
  - Explore the entire design space of data preprocessing pipelines (types, operators, order, feature-wise).
  - Only need to train ML model once.

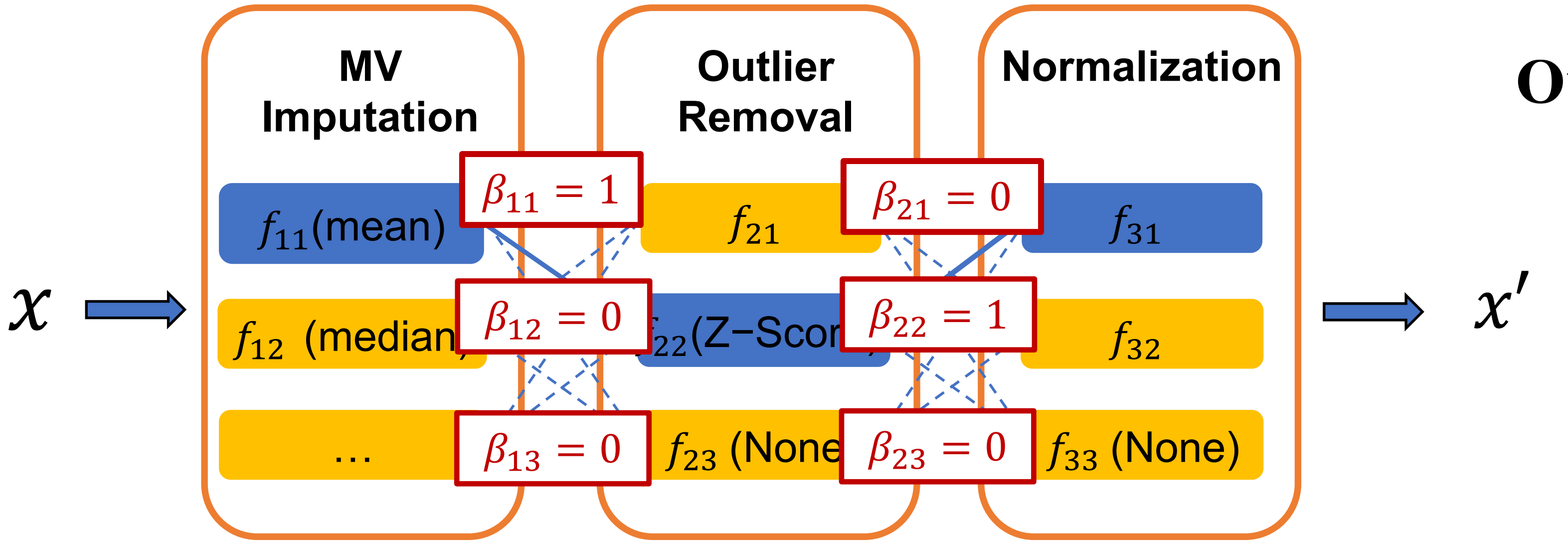
Gradient-based Bi-level Optimization

- **Naive Approach:** we train a ML model with every possible pipeline and select the best one. --- Assume a pipeline contains s transformations and m choices for each transformation. There are m^{sc} feature-wise pipelines for a dataset with c features.
- **Gradient-based approach:** we alternatively and iteratively solve the outer-level and inner-level optimization using gradient descent.
- **Key issue:** the search space of pipelines is discrete and non-differentiable.

Can we convert the discrete search space into a ***continuous and differentiable*** space?

Step 1: Parameterization

- **Goal:** Represent each pipeline in the space using a set of parameters
- Let's first assume we have a predefined order of transformations.



Output of a transformation:

$$x_i = \sum_j \beta_{ij} f_{ij}(x_{i-1})$$

Output of pipeline:

$$x'(x, \boldsymbol{\beta})$$

Loss:

$$Loss\ (D_{val}, \boldsymbol{\beta}, \boldsymbol{w})$$

$$Loss\ (D_{train}, \boldsymbol{\beta}, \boldsymbol{w})$$

Associate each operator with a $\beta_{ij} \in \{0,1\}$

$$\beta_{ij} = \begin{cases} 1 & f_{ij} \text{ is selected} \\ 0 & \text{Otherwise} \end{cases}$$

Constraint: $\sum_j \beta_{ij} = 1$

Step 2: Relaxation

$\boldsymbol{\beta}: \beta_{ij} \in \{0,1\} \xrightarrow{\text{Relax}} \boldsymbol{\beta}: \beta_{ij} \in [0,1]$

- To retain constraints $\sum_j \beta_{ij} = 1$, use **Softmax function**, $\tau_{ij} \in \mathbb{R}$

$$\beta_{ij} = \frac{\exp(\tau_{ij})}{\sum_k \exp(\tau_{ik})}$$

We can now solve Bi-level optimization using gradient descent

Automate Order Selection

- **Question:** How to determine the order of applying transformations (e.g., Outlier Removal, Discretization, Normalization)?
  - Can we try all possible orders? --- The number of possible orders for feature-wise pipelines are exponential to the number of features.

- **Parameterization**: Any order can be represented using a binary permutation matrix α

$$\alpha_{ij} = \begin{cases} 1 & \text{the i-th type is } F_j \\ 0 & \text{Otherwise} \end{cases}$$

$$[\mathrm{O}, \mathrm{D}, \mathrm{N}] \longrightarrow \boldsymbol{\alpha} = \begin{array}{c} \begin{array}{ccc} \mathrm{N} & \mathrm{O} & \mathrm{D} \end{array} \\ \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \end{array}$$

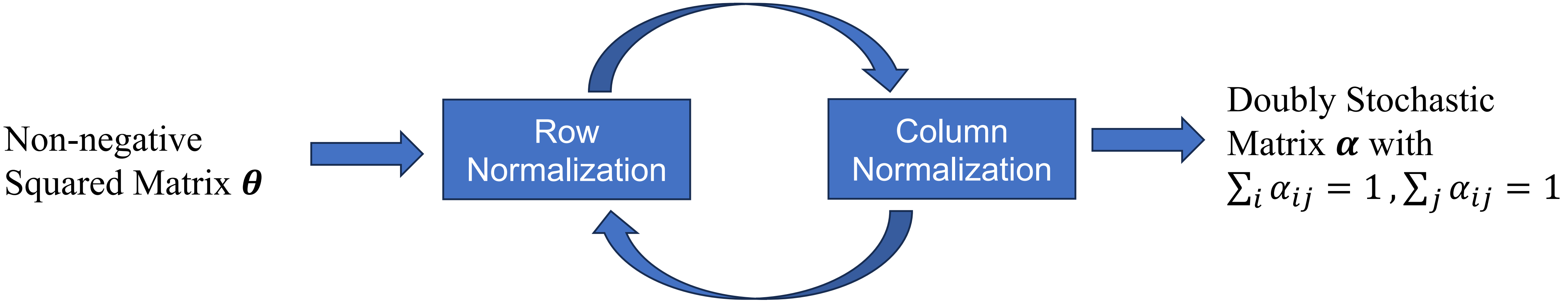
Constraints: $\sum_i \alpha_{ij} = 1 , \sum_j \alpha_{ij} = 1$

Automate Order Selection

- **Relaxation:**

$$\boldsymbol{\alpha}: \alpha_{ij} \in \{0,1\} \xrightarrow{\text{Relax}} \boldsymbol{\alpha}: \alpha_{ij} \in [0,1]$$

To retain constraints $\sum_i \alpha_{ij} = 1\,, \sum_j \alpha_{ij} = 1$, use *Sinkhorn normalization*



We can now learn optimal order, choices of operators and the model simultaneously!

Experiment Setup

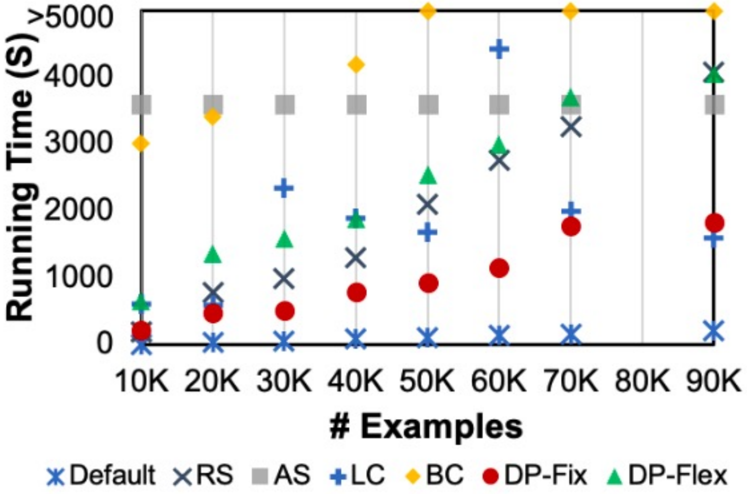
- **Datasets:** 18 real-world datasets
- **Model:** Logistic regression
- **Search Space:** Missing value imputation, Outlier removal, Discretization, Normalization
- **Methods compared:**

Our methods	Practical Methods	AutoML Systems	Advanced Data Cleaning Methods
DiffPrep-Fix (DP-Fix) DiffPrep-Flex (DP-Flex)	Default (DEF) Random Search (RS)	AutoSklearn (AS)	BoostClean (Clean) Learn2Clean (LC)

- **Evaluation Metrics:**
  - Model accuracy
  - Running time

Experiment Results

Dataset	Data Characteristics					Test Accuracy						
	#Ex.	#Feat.	#Cls.	#MVs	#Out.	DEF	RS	AS	LC	BC	DP-Fix	DP-Flex
abalone	4177	9	28	0	200	0.24	0.243	0.216	0.186	0.168	0.238	**0.255**
ada_prior	4562	15	2	88	423	0.848	0.844	0.853	0.816	0.848	**0.854**	0.846
avila	20867	11	12	0	4458	0.553	0.598	0.615	0.597	0.585	**0.638**	0.63
connect-4	67557	43	3	0	45873	0.659	0.671	0.667	0.658	0.69	**0.732**	0.701
eeg	14980	15	2	0	209	0.589	0.658	0.657	0.641	0.659	**0.678**	0.677
jungle_chess	44819	7	3	0	0	0.668	0.669	0.678	0.676	0.667	**0.682**	**0.682**
micro	20000	21	5	0	8122	0.564	0.579	0.584	0.582	0.561	0.586	**0.588**
mozilla4	15545	6	2	0	290	0.855	0.922	**0.931**	0.854	0.930	0.923	0.922
obesity	2111	17	7	0	25	0.775	0.841	0.737	0.723	0.652	0.893	**0.896**
page-blocks	5473	11	5	0	1011	0.942	0.959	**0.969**	0.92	0.951	0.957	0.967
pbcseq	1945	19	2	1445	99	0.71	0.73	0.712	0.704	0.72	0.725	**0.743**
pol	15000	49	2	0	8754	0.884	0.879	0.877	0.737	0.903	0.904	**0.919**
run_or_walk	88588	7	2	0	8548	0.719	0.829	0.851	0.728	0.835	0.907	**0.917**
shuttle	58000	10	7	0	5341	0.964	0.996	**0.998**	0.997	0.997	**0.998**	0.997
wall-robot-nav	5456	25	4	0	1871	0.697	0.872	0.869	0.69	0.9	0.898	**0.914**
google	9367	9	2	1639	109	0.586	0.627	**0.664**	0.549	0.616	0.645	0.641
house	1460	81	2	6965	617	0.928	0.938	**0.945**	0.812	0.928	0.932	**0.945**
uscensus	32561	15	2	4262	2812	0.848	0.840	0.851	0.786	0.848	**0.857**	0.852



Our methods achieve the best test accuracy on 15 out of 18 datasets!

Thank you!