A Boosting Tree Based AutoML System with Concept Drift Adaptation

Meta_Learners:

Zheng Xiong, Jiyan Jiang, Wenpeng Zhang

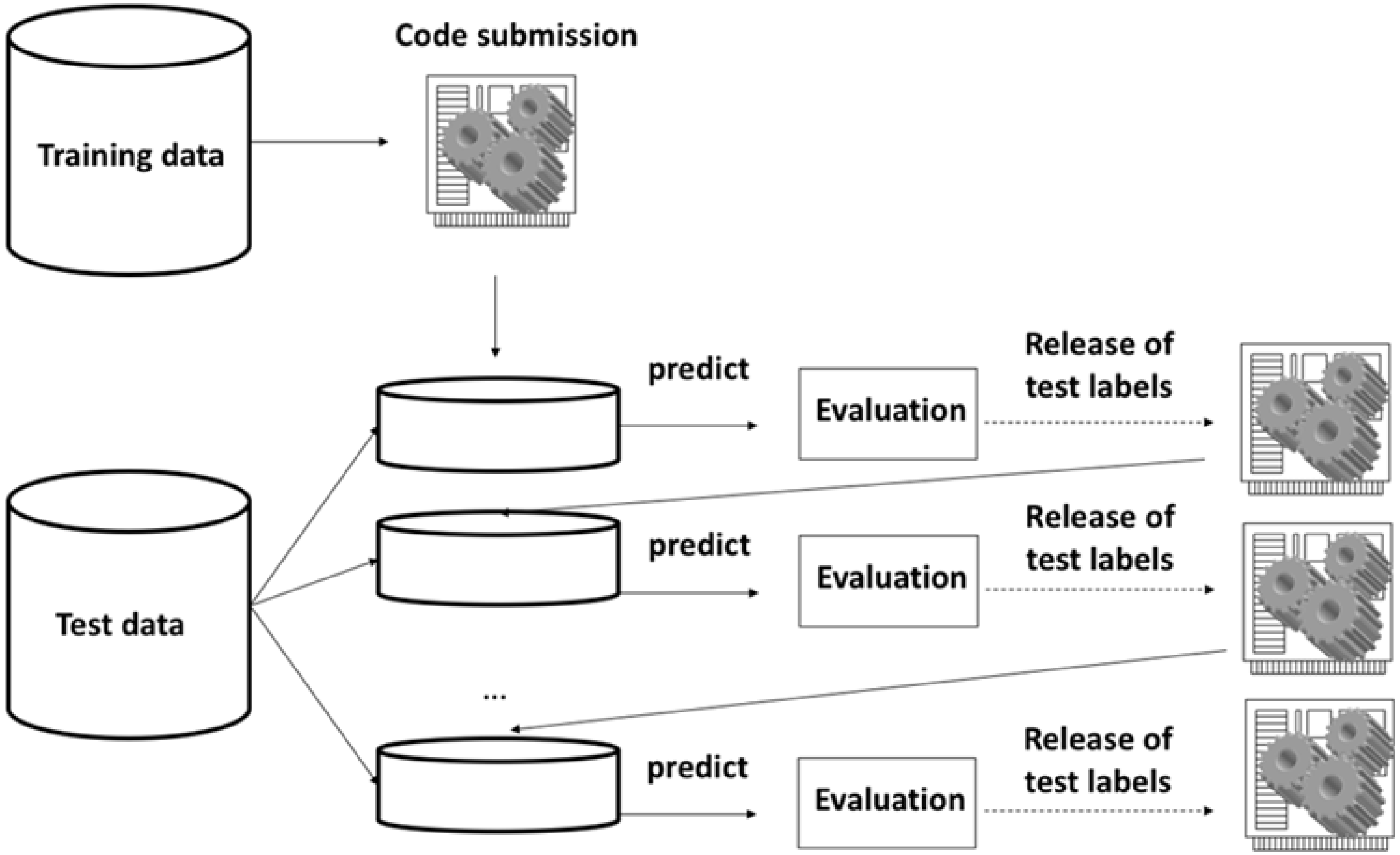
Advisor: Prof. Wenwu Zhu

Department of Computer Science, Tsinghua University, Beijing

Outline

- Problem Statement
- Key Challenges
- System Framework
- Feature Engineering
- Concept Drift Adaptation
- Resource Management

Problem Statement



https://www.4paradigm.com/competition/nips2018

AutoML: the final submission of the feedback phase is blindly tested on 5 unseen new datasets without human intervention

Concept drift: data comes in stream with data distribution changing between batches

Key Challenges

- **Feature engineering**

  Hard to design encoding scheme for categorical features with high cardinality following a power-law distribution

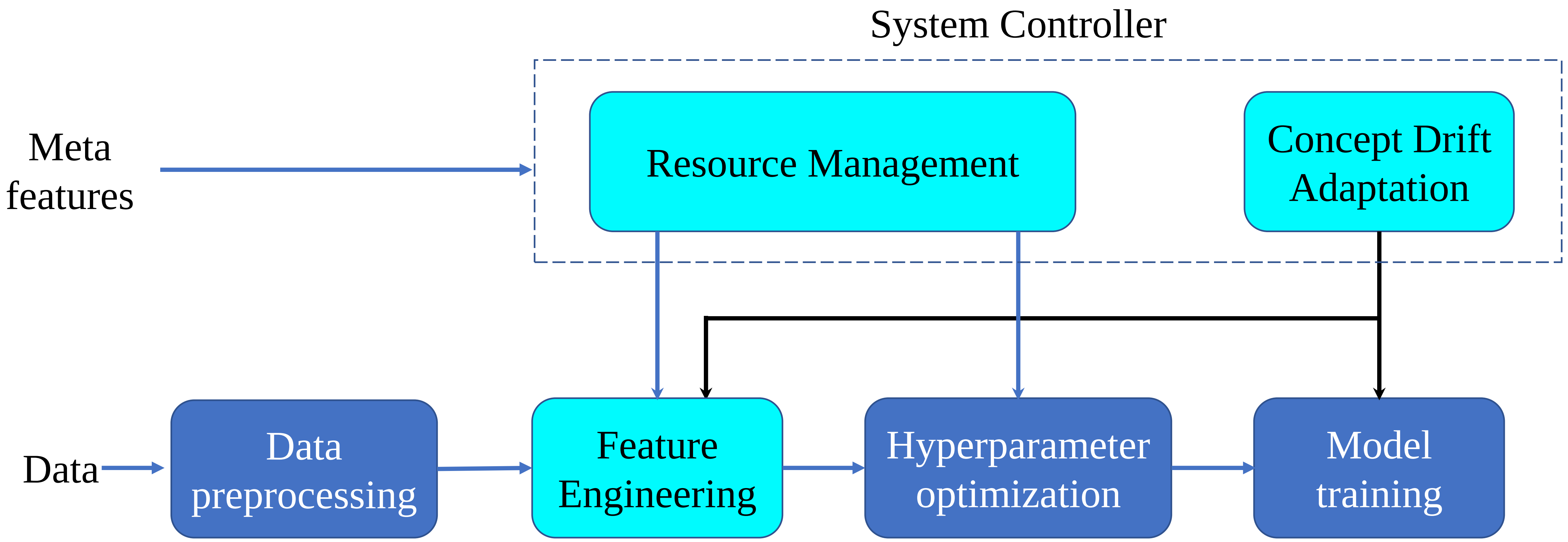
- **Concept drift**

  Data distribution changing slowly over time

- **Scalability and robustness**

  Hard to adapt to time budget and memory constraint on unseen test sets

The System Framework



Feature Engineering

Encoding schemes for categorical features

- Encoding is essential for categorical feature representation
- Count encoding and target encoding provide a compact and informative representation for categorical features with high cardinality and power-law distribution

Feature Engineering

Encoding schemes for categorical features

	Count encoding $p(x)$	**Target encoding** $p(y\|x)$
Pros	Computationally efficient Automatically deal with power-law distribution	More informative than count encoding
Cons		Easy to overfit Sensitive to concept drift

Concept Drift Adaptation

Objective: consistent representation of the training set and test set

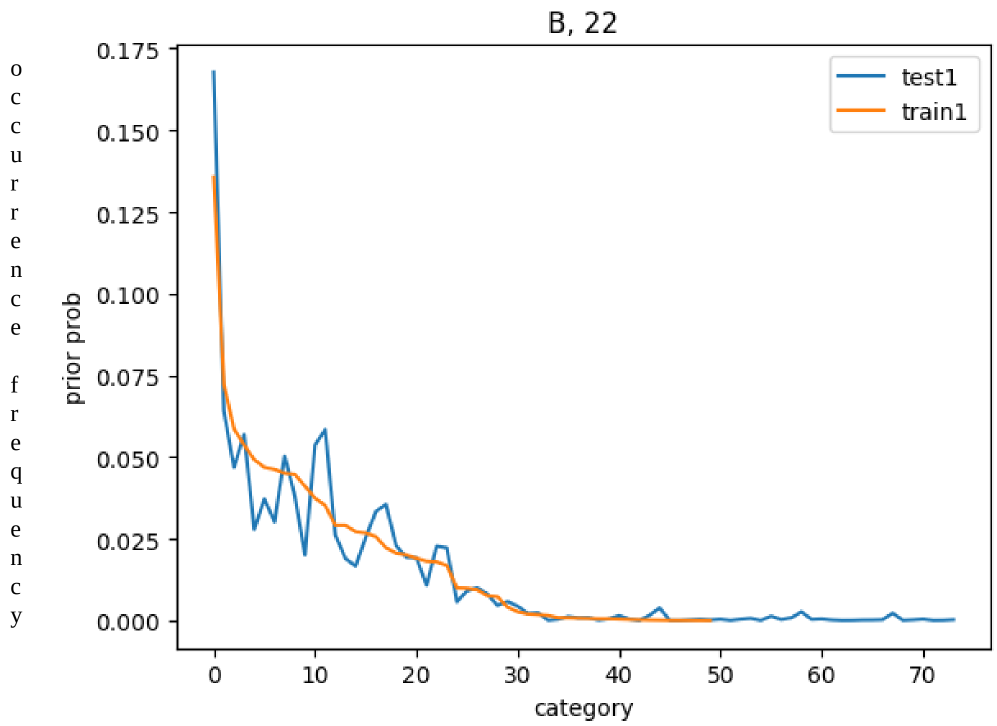
Drift-adaptive training scheme:

Retrain a boosting tree model with the last N batches for each test batch

Key points:

- Retraining vs. incremental learning
- Sliding window on training batches

Concept Drift Adaptation



Concept drift in categorical features

- Many unseen new categories may appear in the test batch
- The frequency of existing categories may change significantly between batches

Propose a streaming co-encoding method, making the feature representation consistent between training set and test set

Concept Drift Adaptation

What is the best strategy to encode categorical features?

First strategy: fit the encoder on the training data and apply it to transform both training and test data

Encoding Strategy	Problem
X_train = encoder.fit_transform(X_train) X_test = encoder.transform(X_test)	Can not deal with unseen categories properly

Concept Drift Adaptation

What is the best strategy to encode categorical features?

Second strategy: fit an encoder for training set and test set respectively

Encoding Strategy	Problem
X_train = encoder.fit_transform(X_train) X_test = encoder.transform(X_test)	Can not deal with unseen categories properly
X_train = encoder_train.fit_transform(X_train) X_test = encoder_test.fit_transform(X_test)	The representation of the same category may not be consistent across batches

Concept Drift Adaptation

What is the best strategy to encode categorical features?

Finally, a co-encoding strategy, which merges the training set and test data together and fit the encoder on this merged set

Encoding Strategy	Problem
X_train = encoder.fit_transform(X_train) X_test = encoder.transform(X_test)	Can not deal with unseen categories properly
X_train = encoder_train.fit_transform(X_train) X_test = encoder_test.fit_transform(X_test)	The representation of the same category may not be consistent across batches
X_train, X_test = encoder.fit_transform([X_train, X_test])	The distribution of training test and test set may be different

Resource Management

Adaptive time budget control

Key idea:

Estimate the computational cost of basic components

Adjust the configuration space according to the time budget

Empirical tunable configurations:

- Whether to use multi-value features or not
- The number of iterations for boosting tree model

Time budget control and hyperparameter tuning are jointly optimized

Resource Management

Adaptive time budget control

Key Steps:

1. Define time_budget_score for each dataset to determine whether or not to use multi-value features
2. Estimate the upper bound for the number of iterations in boosting model and search for the best hyperparameters on the training batch
3. Adaptively adjust the number of iterations in boosting model for each batch based on the estimation of remaining time

Resource Management

Memory control

- The space complexity of the system remains constant as new test batch arrives
- Monitor the memory curve and conduct timely garbage collection
- Automatically tune some hyperparameters to constrain memory usage (e.g. the number of multiprocessing, the number of training batches)

Summary

- Propose a Boosting Tree Based AutoML System with Concept Drift Adaptation for High Cardinality Streaming Data Classification
- Our system's key functions consist of *Feature Engineering*, *Concept Drift Adaptation*, *and Resource Management* for both time and memory constraints
- Future work: although our system is able to generate a consistent feature representation of training set and test set between different batches, the feature distribution may still vary due to concept drift. It remains future work to explore a better encoding strategy to address this problem.

Thank You!

xiongz17@mails.tsinghua.edu.cn

jiangjy17@mails.tsinghua.edu.cn

zhangwenpeng0@gmail.com