Overview

It would be promising to manage test time and further optimize the whole test system if there exists an algorithm to predict the bench test time with a combination chosen from roughly 400 features.

We aim to design a model with input as interested feature combinations and output as the correspondent bench test time.

Data

- **Data Size**
  - 4209 Training Sets and 4209 Test Size.
  - 376 Logistic features and 8 Classification features.

Feature Preprocessing

- **Feature Selection**
  - 4209 Training Sets and 4209 Test Size.
  - 376 Logistic features and 8 Classification features.

Feature Extraction

The goal of feature extraction is to minimize the effect from sparse signals and separate classification features for further processing.

Figure 1. Feature distribution

Figure 2. Feature selection. For the data distribution, we remove the outlier from raw input data.

Model

- **Model Candidates for Selection**
  - XGBoost
  - K-Nearest Neighbors
  - Random Forest
  - Support Vector Machine
  - Decision Tree
  - Gradient Boosting

For XGBoost, the model is tree ensemble including a set of classification and regression trees. We can write the model in the form

\[ \hat{y}_i = \sum f_k(x_i) \]

where \( k \) is the number of trees, \( f_k \) is a function in the functional space \( \mathcal{F} \), and \( F \) is the set of all possible CARTs. Therefore, our objective to optimize can be written as

\[ \text{obj}(\theta) = \sum_i \text{obj}_i(\hat{y}_i, y_i, \sum_k f_k(x_i)) \]

The next step is tree boosting, including regularization, additive training, compute loss function. Then we get

\[ \text{obj}(\theta) = \sum_i \left[ \sum_j \left( \frac{1}{2} \left( y_i - h_j(x_i) \right)^2 + \frac{1}{2} \lambda \| \theta_j \|^2 \right) \right] \]

Renormalizing the tree model, we can write the objective value with the t-th tree as

\[ \text{obj}(\theta) = \sum_i \left[ \sum_j \left( \frac{1}{2} \left( y_i^n - h_j^n(x_i) \right)^2 + \frac{1}{2} \lambda \| \theta_j^n \|^2 \right) \right] \]

Where \( y_i^n = \{ (y_i, h_j^n(x_i)) | j \} \) is the set of indices of data points assigned to the j-th leaf. We can have \( G_j = \sum_i h_j \) and \( h_j = \sum_i h_j \)

\[ \text{obj}(\theta) \]

The last equation measures how good a tree structure \( \theta(x) \) is. The smaller the score is, the better the structure.

Result

- **Test Data**
  - Divide raw data into 75% training set and 25% cross-validation set.
  - 4209 test samples.

- **Model Evaluation**
  - We choose the R-square to assess our model.

\[ R^2 = 1 - \frac{\text{SSres}}{\text{SSTot}} \]

Future Work

- Using Deep Neural Network for data feature.
- Get Feature Hierarchy by training each layer.
- The training is based on previous layer’s output.
- Detect the latent structures of our data, with feature hierarchy.
- Deep Neural Network enables the automatic feature extraction.
- Feature Extraction without DNN vs. with DNN