School of Computer Science

UNIVERSITY OF BIRMINGHAM

Intelligent Software Engineering
Lab 2: Configuration Performance Learning

Dr. Tao Chen

This is lab2 for the module, which can be chosen as the problem for the coursework. If you need specific support in terms of programming, please attend one of the lab sessions and ask the TAs for help.

The Content

For Lab2, you will build a **baseline tool** using Linear Regression ($\mathcal{M}$) to automatically predict the likely performance achieved by a given configuration of a software system:

$$\mathbf{p} = \mathcal{M}(\mathbf{x}) \tag{1}$$

whereby $\mathbf{x}$ is the set of configuration values of the relevant options; $\mathbf{p}$ is the value of the predicted performance, which can be latency, throughput, energy consumption etc.

You are welcome to research the internal working mechanism of Linear Regression and its training, but some existing implementations are available from the well-known library: `https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html`.

The Datasets of Systems

This lab covers the measured datasets for a range of widely studied configurable systems. The selection aligns with this study to ensure consistency and comprehensiveness. Specifically, these systems are carefully chosen to span a variety of application domains, performance objectives, and programming languages, including both Java and C/C++, thereby providing a solid foundation for our investigation into systems with diverse characteristics, as shown in Table **??**.

Table 1: Subject system characteristics. **#O:** No. of options; **#C:** No. of configurations; **#W:** No. of workloads tested.

System	Lang.	Domain	Perf.	Version	#O	#C	#W
JUMP3R	Java	Audio Encoder	Runtime	1.0.4	16	4196	6
KANZI	Java	File Compressor	Runtime	1.9	24	4112	9
DCONVERT	Java	Image Scaling	Runtime	1.0.0-α7	18	6764	12
H2	Java	Database	Throughput	1.4.200	16	1954	8
BATLIK	Java	SVG Rasterizer	Runtime	1.14	10	1919	11
XZ	C/C++	File Compressor	Runtime	5.2.0	33	1999	13
LRZIP	C/C++	File Compressor	Runtime	0.651	11	190	13
X264	C/C++	Video Encoder	Runtime	baee400...	25	3113	9
Z3	C/C++	SMT Solver	Runtime	4.8.14	12	1011	12

The workloads we studied are diverse and domain-specific. For example, for SMT solver `Z3`, the workload can be different SMT instances that are of diverse complexity, e.g., `QF_RDL_orb08` and `QF_UF_PEQ018`; for database system `H2`, the workloads are requests with different rates and types (e.g., read-only or read-write), such as `tpcc-2` and `ycsb-2400`.

Each system features a distinct configuration space, covering different option types (e.g., integers, boolean, and enumerates) and dimensions.

The link and details of the datasets can be accessed here: https://github.com/ideas-labo/ISE/tree/main/lab2.

How to use the Datasets?

For this lab, each system workload is a dataset. For example, `x264` combined with one of its workloads would be one dataset for you to train and test your model. Of course, the total number of datasets you can test equals the sum of all workloads.

The normal procedure is that, for each dataset, you would randomly select 70% samples to train the baseline and use the remaining 30% for testing. The above process is repeated, e.g., 30 times, to avoid stochastic bias. Note that in each repeat, the training/testing sample of the configuration performance is randomly selected for all approaches you wish to compare, if any. You can then examine the results against the metrics below.

The Metrics

Commonly, for regression problems, we can use different metrics:

- **mean absolute percentage error (MAPE)**:

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \cdot 100\% \tag{2}$$

  where n is the number of testing samples; $\hat{y}_i$ and y_i correspond to the model-predicted and true performance value of a configuration, respectively. Since MAPE measures the error, the smaller the MAPE, the better the accuracy.

- **mean absolute error (MAE)**:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{3}$$

  where n is the number of testing samples; $\hat{y}_i$ and y_i correspond to the model-predicted and true performance value of a configuration, respectively. Since MAE measures the error, the smaller the MAPE, the better the accuracy.

- **root mean square error (RMSE)**:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{4}$$

  where n is the number of testing samples; $\hat{y}_i$ and y_i correspond to the model-predicted and true performance value of a configuration, respectively. Since RMSE measures the error, the smaller the MAPE, the better the accuracy.

Some existing implementations of the metrics can be found here:

- `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean_absolute_percentage_error.html`
- `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean_absolute_error.html`
- `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.root_mean_squared_error.html`

The Statistical Test

If there is nothing to compare, then you might not need the statistical test. The link to an existing statistical test library can be found here: `https://docs.scipy.org/doc/scipy/reference/stats.html`. You will find implementations of the tests mentioned in the module and those that we have not talked about.