Performance Prediction of Configurable Software Systems by Fourier Learning

Yi Zhang, Jianmei Guo,
Eric Blais, Krzysztof Czarnecki

UNIVERSITY OF WATERLOO

Overview

- The Problem
- The Tool
- The Solution
- Evaluation

The Problem

- Given configurable software systems with n (binary) features
- Each configuration is a set of features
- Each configuration has a performance value, e.g. execution time
- **Goal:** Predict the performance of all (valid) configurations by measuring a (small) sample of configurations.

The Problem: Example

Feature_1	Feature_2	Feature_3	Performance
1	0	0	7.0
0	1	1	5.9
1	1	0	8.1
0	0	1	?
...	...	...	?
1	1	1	?

The Problem: Alternative

x			f(x)
x_1	x_2	x_3	
1	0	0	7.0
0	1	1	5.9
1	1	0	8.1
0	0	1	?
...	...	...	?
1	1	1	?

The Problem: Alternative

- Given *n,* the number of features
- Configuration: bit vector $x \in \{0,1\}^n$
- Performance: *f(x)*
- **Goal:** Estimate *f(x)* for all $x \in \{0,1\}^n$

  i.e. learn the function f

The Challenge

This is impossible
for arbitrary *f* !

f(101)
has nothing to do with
f(110)

The Good News:

Functions representing **real** software systems **have structure**.

The Tool: Fourier Analysis

Given a function: $f:\{0,1\}^n \rightarrow \mathbb{R}$

Can write f as:

$$f(x) = \sum_{z\in\{0,1\}^n} \hat{f}(z)\chi_z(x)$$

where:

$$\chi_z(x) := \begin{cases} +1 & \text{if } \sum_{i=1}^n z_i x_i = 0 \mod 2 \\ -1 & \text{if } \sum_{i=1}^n z_i x_i = 1 \mod 2 \end{cases}$$

The Tool: Observations

- For a function $f:\{0,1\}^n \rightarrow \mathbb{R}$

  There are 2^n Fourier coefficients

- Knowing the coefficients is

  equivalent to knowing the function itself.

The Tool: Example

x		*f(x)*
0	0	3
0	1	2
1	0	4
1	1	1

$$f(x) = 2.5 \cdot \chi_{00}(x) + 1 \cdot \chi_{01}(x)$$

$$+ \mathbf{0} \cdot \chi_{10}(x) + (-0.5) \cdot \chi_{11}(x)$$

The Tool: Fourier Analysis

$$\hat{f}(z) = \frac{1}{2^n} \sum_{x \in \{0,1\}^n} f(x) \cdot \chi_z(x)$$

For example:

$$\hat{f}(01) = \frac{1}{4}\big(f(00)\cdot\chi_{01}(00) + f(01)\cdot\chi_{01}(01)$$
$$+ f(10)\cdot\chi_{01}(10) + f(11)\cdot\chi_{01}(11)\big)$$

$$\hat{f}(01) = \frac{1}{4}(3 - 2 + 4 - 1) = 1$$

The Good News:

Functions representing **real** software systems

~~**have structure**~~

are Fourier sparse!

(when normalized)

i.e. many coefficients are (close to) **0**.

The Problem: Final

- Given *n,* the number of features
- Configuration: bit vector $x \in \{0,1\}^n$
- Performance: *f(x)*
- **Goal:** ~~Estimate *f(x)* for all $x \in \{0,1\}^n$~~

  Estimate all (large) Fourier coefficients of *f*.

The Solution: Idea

$$\hat{f}(z) = \frac{1}{2^n} \sum_{x \in \{0,1\}^n} f(x) \cdot \chi_z(x)$$

Take random sample *S*:

$$\hat{h}(z) \approx \frac{1}{|S|} \sum_{x \in S} f(x) \cdot \chi_z(x) \qquad (*)$$

Use $\hat{h}(z)$ to construct h

The Solution: Theorem (Hoeffding)

Given *f* is Fourier-sparse, if *S* is large, then *h* is close to *f* with high probability.

The Solution: Theorem

Given $f:\{0,1\}^n \rightarrow \mathbb{R}$ is Fourier t-sparse, with

$$\frac{2}{\epsilon^2}\left(\left(n+1\right)\log\left(2\right)+\log\left(\frac{1}{\delta}\right)\right)$$

samples, our estimation *h* can achieve:

$$\|f-h\|^2 < t\cdot\epsilon^2$$

with probability $1-\delta$.

The Solution: Algorithm

1) User specify error bound γ and confidence level
2) Assume t = 1 (*f* is 1-sparse), and calculate number of samples required
3) Take the measurements and calculate Fourier coefficients using (*), obtain *h*
4) Take more samples and estimate the distance between h and f
5) If not within the specified bound, increase t and repeat

Evaluation: Systems

TABLE II. SUMMARY OF ORIGINAL SOFTWARE SYSTEMS

System	Domain	Lang.	$\|D\|$	n
Apache	Web Server	C	192	8
x264	Encoder	C	1,152	13
LLVM	Compiler	C++	1,024	10
Berkeley DB	Database	C	2,560	16
Berkeley DB	Database	Java	180	17

|D| = # Total configurations.

Original systems too small.

Evaluation: Hybrid-systems

System x*y

x		y		f(x*y)
x_1	x_2	y_1	y_2	f(x)+f(y)
0	0	0	0	3+2
0	0	0	1	3+4
0	0	1	0	3+1
...	...	...	...	...
1	1	1	1	5+3

Evaluation: Systems

TABLE III. SUMMARY OF CONSTRUCTED HYBRID-SYSTEMS

System	Component	$\|D\|$	n
A	Apache + x264	221184	21
B	LLVM + x264	1179648	23
C	x264 + x264	1327104	26
D	LLVM + LLVM	1048576	20

|D| = # Total configurations.

Evaluation: Results

1) Confidence level set to be 80%

2) Run 10 times for each setting

System	n	γ	Samples	mean	max
A	21	0.2	24236 (11%)	0.083	0.084
		0.15	43307 (20%)	0.081	0.081
		0.1	97440 (44%)	0.074	0.074
B	23	0.2	26332 (2.2%)	0.035	0.036
		0.15	46812 (4.0%)	0.026	0.026
		0.1	105326 (8.9%)	0.0068	0.0069
C	26	0.2	29289 (2.2%)	0.084	0.085
		0.15	52070 (3.9%)	0.084	0.084
		0.1	117156 (8.8%)	0.080	0.082
D	20	0.2	23375 (2.2%)	0.074	0.080
		0.15	41555 (4.0%)	0.034	0.037
		0.1	93497 (8.9%)	0.024	0.024

Evaluation: Comparison

1) SPLConqueror from Siegmund et. al.(2012) uses feature interaction to predict performance.

2) CART from Guo et. al. (2013) uses machine learning techniques.

	SPLConqueror	CART	Fourier
Accuracy	~ 95%	~ 94%	Arbitrary*
Sample Size	$O\left(n^2\right)$	Any	$O\left(n, 1/\gamma^2\right)$
Sampling	Specific	Random	Random
Error Control	No	No	Yes
System	Any	Any	Large

Summary

1) Fourier learning predicts software performance with guaranteed accuracy and confidence level

2) May require large systems and run time may be slow

3) Future: reduce exponential number of Fourier coefficient estimations

4) Future: testing Fourier sparse-ness of systems

Thank you.