Cost-Efficient Sampling for Performance Prediction of Configurable Systems

Atri Sarkar, Jianmei Guo, Norbert Siegmund, Sven Apel, Krzysztof Czarnecki

Amanda Damasceno Santana
Reuso 2018/02, DCC-UFMG

Estrutura da Apresentação

I. Introdução
II. Uma revisão do processo de amostragem
III. Definições e exemplos
IV. Modelo de Custo
V. Amostragem Progressiva e Projetiva
VI. Geração de amostra inicial
VII. Sistemas usados na avaliação
VIII. Avaliação
IX. Conclusão

I. Introdução

Motivação: configurar suas opções ou *features* permitem com que os *stakeholders* customizem o sistema de diversas maneiras, dando origem a um enorme conjunto de variantes ou configurações de sistemas. Porém, cada *feature* pode acarretar efeitos nas propriedades funcionais e não funcionais do sistema, como por exemplo na performance e no custo.

I. Introdução

Contudo, no caso de sistemas configuráveis fazer a análise de performance não é uma tarefa trivial, pois, devido aos fatores combinatórios, o número de variantes do sistema aumenta exponencialmente conforme o número de *features* que o sistema provê. Se avaliar uma única variante pode ser custoso, como avaliar um conjunto significativos dos sistemas?

Um exemplo é o SQLite, cujas 39 *features* permitem mais de 3 Milhões de sistemas variantes.

I. Introdução

Objetivo: Propor uma estratégia de amostragem inteligente que determina dinamicamente uma “boa” amostra para um dado sistema. Uma amostra é considerada boa se é pequena o suficiente para diminuir os esforços de medição e se é grande o suficiente para aumentar a acurácia da predição (*tradeoff*).

II. Revisão do processo de amostragem

Modelos de predição são construídos iterativamente:

1. São medidas algumas configurações do sistema (treino e teste);
2. O conjunto de treino é utilizado para construir o modelo de predição;
3. Este modelo é avaliado utilizando uma métrica de avaliação no conjunto de teste;
4. Se o valor da métrica for aceitável, o processo para. Caso contrário, mais medidas serão necessárias.

II. Revisão do processo de amostragem

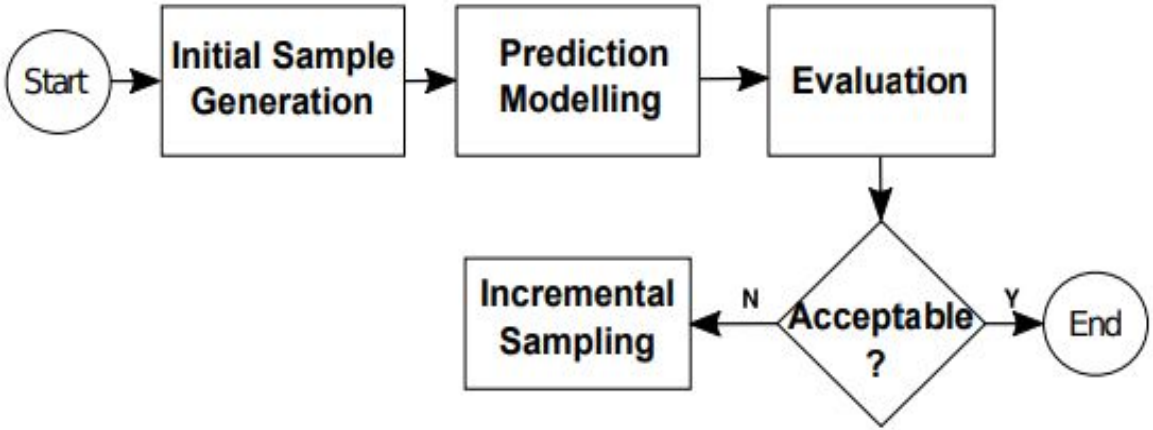


Figure 1: General process of performance prediction by sampling

II. Revisão do processo de amostragem

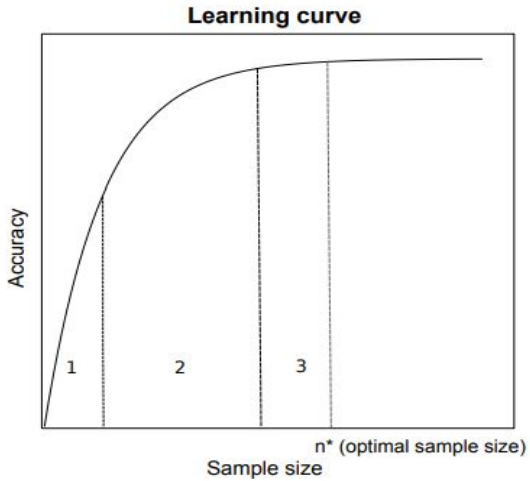


Figure 2: Regions of the learning curve in dependence of the sample size: (1) Steep incline; (2) Gradual incline; (3) Plateau; n^*: marks the optimal sample size

III. Definições e exemplo

$X = \{ x_1, x_2, ..., x_N \}$;

$x_i = \{0 \vee 1\}$;

N número de *features* no sistema;

Cada configuração de um sistema é uma N-tupla;

O conjunto de todas configurações válidas é denotada como **X**.

III. Definições e exemplo

A curva de aprendizado é um mapeamento entre o tamanho do conjunto de treino e a acurácia em se treinar com este mesmo conjunto.

$\lambda_n = (n, \epsilon_n)$

n = tamanho do conjunto de treino

ϵ_n = erro de predição do modelo dado que você o construiu com um conjunto de treino de tamanho n

IV. Modelo de Custo

Motivação: Quanto maior é o conjunto de treino, maior a acurácia da predição. Contudo, obter um grande conjunto de treino muitas vezes é inviável, devido ao custo de se obter as medições.

Avaliar a performance levando em consideração o custo de medição e da acurácia da predição.

IV - Modelo de Custo

$$\text{Custo Total} = \text{Custo}_{\text{Medição(Treino)}} + \text{Custo}_{\text{Medição(Teste)}} + \text{Custo}_{\text{ConstruçãoDoModelo}} + \text{Custo}_{\text{ErroDePredição}}$$

IV. Modelo de Custo

Como os autores usam o método CART (Classification and Regression Tree) para construir o modelo de predição de performance e, também, dividem o conjunto de treino e de teste em partes iguais (50:50), a expressão se simplifica para

Custo Total = $2n + \epsilon_n * |S| * R$

2n = número de amostras no conjunto de treino e teste
ϵ_n = erro de predição da performance do modelo construído com n configurações
|S| = o número de configurações cujos valores de performance serão avaliados pelo modelo
R = parâmetro que controla o quão custoso predizer a performance errado

V. Amostragem Progressiva

A ideia central deste algoritmo é adicionar em cada iteração n_i amostras. n_0, n_1, n_2, ..., n_k

Exemplo do Apache:
Dado a = 10 e n_0 = 10
Aritmética:

$$10_{(\text{interação } 0)} + 10_{(\text{interação } 1)} + 10_{(\text{interação } 2)} + 10_{(\text{interação } 3)} + 10_{(\text{interação } 4)}$$

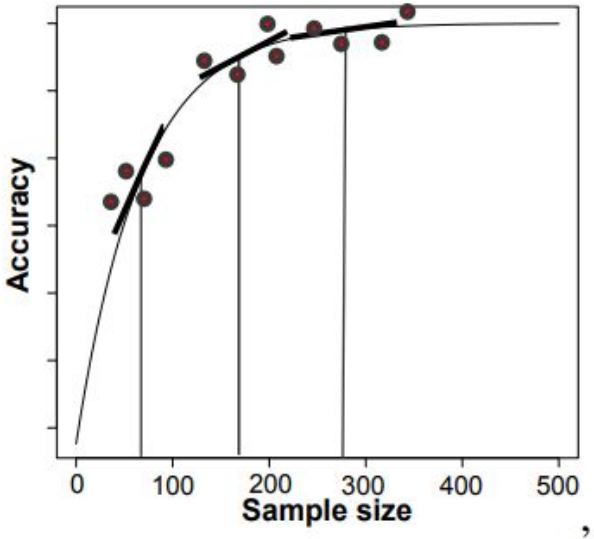
Geométrica:

$$10_{(\text{interação } 0)} + 10_{(\text{interação } 1)} + 20_{(\text{interação } 2)} + 40_{(\text{interação } 3)}$$

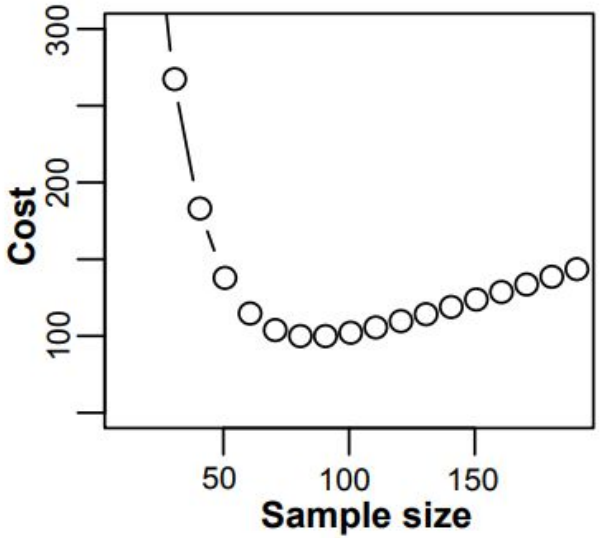
Table I: Learning curve points (λ_n) for Apache (n : sample set size, ϵ_n : relative error %)

n	ϵ_n	n	ϵ_n
10	19.26	60	7.52
20	11.12	70	7.44
30	8.62	80	7.25
40	8.23	90	7.17
50	7.76		

V. Amostragem Progressiva



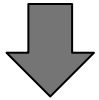
(a) Gradient-based



(b) Cost minimization

V. Amostragem Projetiva

Motivação: Amostragem progressiva pode convergir depois de várias iterações com um tamanho de amostra grande, podendo, no ponto de convergência, ter custo e acurácia não satisfatórias para o usuário.



Aproximar a curva de aprendizado usando um conjunto inicial mínimo de amostras.

V. Amostragem Projetiva

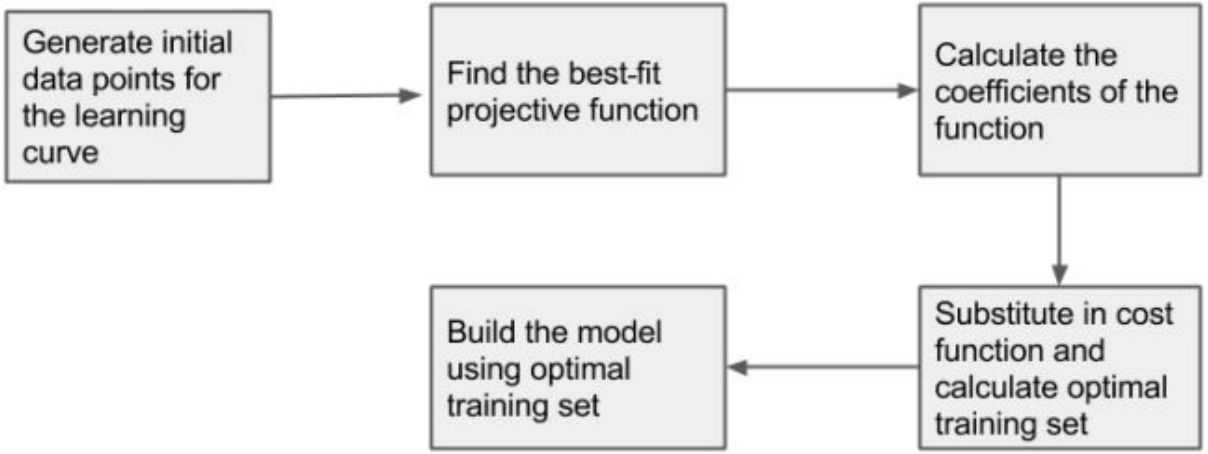


Figure 4: Overview of projective sampling

VI. Amostragem Projetiva

Table II: Projective functions of learning curves

Name	Equation	Optimal Sample Size
Logarithmic	$err(n) = a + b.log(n)$	$n^* = -(R \cdot \lvert S \rvert \cdot b)/2$
Weiss and Tian	$err(n) = a + bn/(n+1)$	$n^* = \sqrt{(-R \cdot \lvert S \rvert \cdot b)/2}$
Power Law	$err(n) = an^b$	$n^* = (\frac{-2}{R \cdot \lvert S \rvert \cdot a \cdot b})^{\frac{1}{b-1}}$
Exponential	$err(n) = ab^n$	$n^* = \log_b \left(\frac{-2}{R \cdot \lvert S \rvert \cdot a \cdot \ln b} \right)$

VI. Geração de Amostra Inicial

Objetivo: Dado um ponto amostral λ_i de uma curva de aprendizado, amostrar um conjunto $\Lambda_\delta = \{\lambda_1, \lambda_2, ..., \lambda_\delta\}$ tal que Λ_δ possa ser usado para gerar a curva de aprendizado projetada acuradamente.

Fatores a serem levados em consideração:

1. Para fazer com que a estratégia de amostragem seja eficiente em seu custo, o valor de δ deve ser menor que n^*
2. É importante que a curva projetada se aproxime da real curva de aprendizado acuradamente, já que uma escolha subótima da amostra inicial possui grande probabilidade de resultar em uma função projetiva que não é próxima do comportamento de aprendizado do modelo.

VI. Geração da Amostra Inicial

Proposta para a geração da amostra inicial:

- Usuários podem selecionar e desselecionar *features*, implicando na relação entre as *features* e a performance. Esta implicação sugere que uma boa amostra de configurações inicial Λ_{δ} deve possuir cada uma das *features*, por pelo menos em uma configuração da amostra (estas restrições só se aplicam para *features* opcionais).

VI. Geração da Amostra Inicial

Algorithm 1 Generate configurations for projective sampling

1: **while** ($curr\_freq < thresh\_freq$ **AND** $mean\_err > err\_thresh$) **OR** ($curr \leq 3$) **do**
2: $c \leftarrow$ RAND() ▷ Randomly generate a configuration c
3: $S_{curr} \leftarrow S_{curr} \cup c$
4: UPDATE(T) ▷ Update the feature-freq table $\boldsymbol{T}$
5: $\epsilon_{curr} \leftarrow$ CART(S_{curr}) ; $\varepsilon \leftarrow \varepsilon \cup \epsilon_{curr}$
6: $mean\_err$ = MEAN(ε)
7: $\lambda_{curr} = (curr, \epsilon_{curr})$; $\Lambda_\delta \leftarrow \Lambda_\delta \cup \lambda_{curr}$ ▷ Add the current learning curve sample point (λ_{curr}) to Λ_δ
8: $curr\_freq \leftarrow \min(t[1, j], t[2, j])$; $curr \leftarrow curr + 1$
9: **end while**

VII. Sistemas Usados na Avaliação

Table V: Overview of the six subject systems. Lang: language; LOC: lines of code; $|\mathbf{X}|$: number of all valid configurations; N: number of all features

	System	Domain	Lang.	LOC	$\|\mathbf{X}\|$	N
1	Apache	Web Server	C	230,277	192	9
2	LLVM	Compiler	C++	47,549	1,024	11
3	x264	Encoder	C	45,743	1,152	16
4	Berkeley DB	Database	C	219,811	2,560	18
5	Berkeley DB	Database	JAVA	42,596	400	26
6	SQLite	Database	C	312,625	3,932,160	39

VIII. Avaliação

Qual a melhor técnica de amostragem?
S = |X|/3 e R=1, acurácia: 50:50 teste/treino, em 30 experimentos

Table VI: Cost and accuracy of progressive and projective sampling

	Cost		Accuracy (%)	
	Progressive	Projective	Progressive	Projective
Apache	616	**602**	92	**92**
Berkeley DB (C)	86679	**11206**	1	**88**
Berkeley DB (Java)	355	**328**	96	**96**
LLVM	1263	**946**	96	**97**
SQLite	2517	**1828**	98	**99**
x264	2807	**2121**	92	**95**

VIII. Avaliação

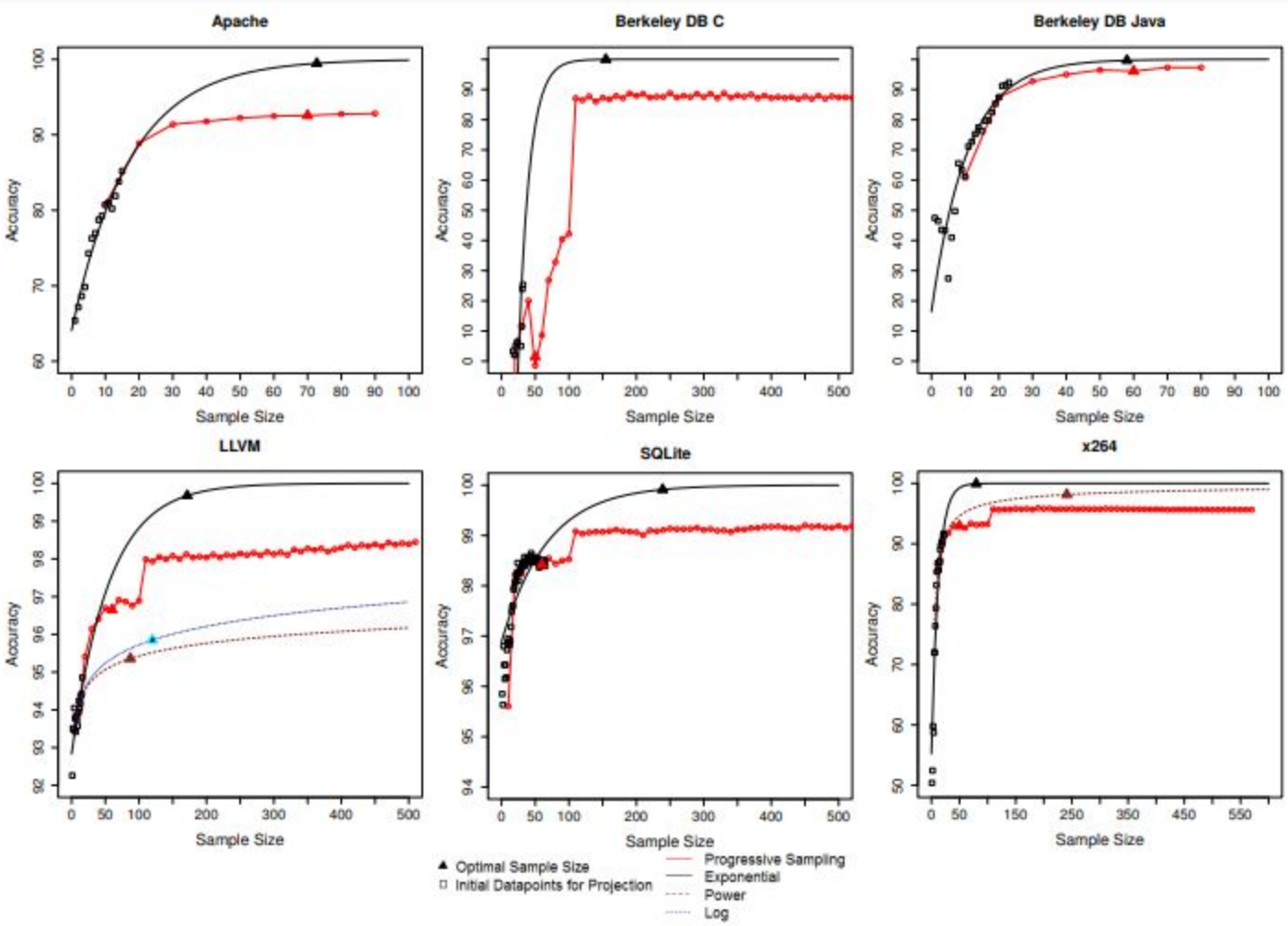


Figure 5: Learning curves for progressive and projective sampling

VIII. Avaliação

Table IX: t-way vs. feature-frequencies; r: Pearson's correlation coefficient.

		r	p *value*	**Total Cost**
Apache	2-way	0.981	0.000017	764
	3-way	0.977	0	610
	feature frequency	**0.989**	0	**602**
Berkeley DB (C)	2-way	0.858	0.000001	69572
	3-way	0.903	0	11212
	feature frequency	**0.92**	0	**11206**
Berkeley DB (Java)	2-way	0.895	0.000468	818
	3-way	0.943	0	6102
	feature frequency	**0.971**	0	**328**
LLVM	2-way	–	–	–
	3-way	0.006	0.981767	2082
	feature frequency	**0.915**	0.000001	**946**
SQLite	2-way	0.888	0	2470
	3-way	0.941	0	**1697**
	feature frequency	**0.943**	0	1828
x264	2-way	0.942	0.000005	2798
	3-way	0.968	0	2803
	feature frequency	**0.97**	0	**2121**

IX. Conclusão

- Amostragem projetiva é melhor do que a amostragem progressiva em termos de custo e de acurácia, mas sofre com a dependência de um bom conjunto inicial de amostras e de uma função projetiva que seja próxima da curva de aprendizado real;
- A heurística baseada na frequência de *features* é mais eficiente do que as outras abordagens utilizadas;
- A função exponencial foi a melhor em se aproximar das curvas reais dos sistemas escolhidos;
- É recomendado usar amostragem progressiva aritmética ao invés da geométrica.

Dúvidas?