Literature Review
Using machine learning in real-time system

Hang Xu

Outline

1. Historic use of ML in real-time system

2. Dimensions to distinguish ML papers

3. Relation with my work

Historic use of ML in RTS

1. Blackbox technique to deal with the uncertainty and complexity in the development of RTS

2. Narrower scope to apply ML due to the reduction of uncertainty by other static and deterministic RT analysis approaches

3. New uncertainty and complexity in RTS making ML still alive now

Use ML to predict worst execution time of real time system

1. Complexity to precisely and efficiently predict execution time:

(1) CPU hardware complexity – branch prediction,OOE,prefectching

(2) Variety on hardware, OS

(3) resource sharing among tasks or early multi-core system – cache, memory bus

→Relatively more papers on using ML to predict the WCET with full static information

1. Statistical Prediction of Task Execution Times through Analytic Benchmarking for Scheduling in a Heterogeneous Environment, M.A. Iverson ; F. Ozguner ; L.C. Potter ,Heterogeneous Computing Workshop (HCW'99)

2. Reliable Estimation of Execution Time of Embedded Software, P Giusto, G Martin, E Harcourt, Design, automation and test in Europe(DATE), 2001

3. A Data Analysis Method for Software Performance Prediction, G. Bontempi,W. Kruijtzer, (DATE 2002)

4. Accurate Software Performance Estimation Using Domain Classification and Neural Networks, Marcio Seiji Oyamada,Felipe Zschornack, Flavio Rech Wagner, symposium on Integrated circuits and system design (SBCCI 2004)

Use ML to predict worst execution time of real time system

2. Improved and mature static analysis techniques(around 2010):

(1) Abstract interpretation

(2) Well static modeling of caching effect in single-core and multi-core architectures

(3) Well static modeling of CPUs and OSes

→ Well static analysis approaches and tools (I.g. Absint: ait), few papers on using ML to predict WCET after design

1. Combining Abstract Interpretation with Model Checking for Timing Analysis of Multicore Software, Mingsong Lv, Wang Yi, Nan Guan, Ge Yu ,RTSS2010
2. A Fast and Precise Static Loop Analysis based on Abstract Interpretation Program Slicing and Polytope Models, P. Lokuciejewski, D. Cordes, H. Falk, P. Marwedel, International Symposium on Code Generation and Optimization(CGO) 2009
3. A Unified WCET Analysis Framework for Multi-core Platforms, Sudipta Chattopadhyay ; Chong Lee Kee ,etc. RTAS2012
4. Toward Static Timing Analysis of Parallel Software -- Technical Report, Andreas Gustavsson, Jan Gustafsson, and Björn Lisper

Use ML to predict worst execution time of real time system

3. Drawback of static analysis techniques

(1) prior and complete knowledge of real time systems(Source code, hardware may not be available)

(2) longer tool-chain and more manual support

→ Recent papers using ML to estimate WCET only on early stage of the development of a project

1. Huybrechts, T., et al.: A new hybrid approach on WCET analysis for real-time systems using machine learning. In: Brandner, F. (ed.) (WCET 2018)
2. Early WCET Prediction Using Machine Learning, Armelle Bonenfant and Denis Claraz and Marianne de Michiel and Pascal Sotin,17th WCET 2017
3. Early execution time-estimation through automatically generated timing models, Peter Altenbernd, Jan Gustafsson, Björn Lisper,Friedhelm Stappert, Real-Time Syst Journal(2016)

Outline

1. Evolution of ML in real-time system

2. Dimensions to distinguish ML papers

3. Relation with my work

Dimensions to distinguish existing ML papers

1. Feature selection (most are supervised learning)
2. Variety and complexity of ML techniques
3. Different application domain

Examples of features

Most based on static information from source code

- count of certain sorts of instructions
- count of certain sorts of variables
- count of certain sorts of functions

Some based on observations from measurements

- execution time
- power consumption

The selection - based on domain expert knowledge

Variety of ML Techniques

Compared to early papers(during 2000s), later ones tend to use more advanced and complex ML techniques.

- KNN, SVM, ANN, DT, regression, ensembling model, etc.

(genetic algorithm may not be considered ML but randomness based AI in my opinion)

1. Methods for Prediction, Simulation and Verification of Real-Time Software Architectural Design based on Machine Learning Algorithms, Mostafa Anwar Taie, Ibrahim El-Faramawy, Mohamed Elmawazini, SAE International tenchnique report 2015
2. Intelligent Prediction of Execution Times, D Tetzlaff, S Glesner - International Conference on Informatics & Applications (ICIA), 2013
3. Development of machine learning-based real time scheduling systems: using ensemble based on wrapper feature selection approach, Y.R. Shiue, R.S. Guh, K.C. Lee International Journal of Production Research, (2012)
4. On the use of machine learning to predict the time and resources consumed by Applicationss, Andréa Matsunaga, José A.B. Fortes, CCGRID 2010

Different Application Domain

Scheduling optimization;

Anomaly detection;

Power management;

1. Real-time scheduling via reinforcement learning,R. Glaubius, T. Tidwell, C. D. Gill, W. D. Smart, Uncertainty in Artificial Intelligence (UAI2010)
2. J. Song, G. Fry, C. Wu, G. Parmer, "CAML: Machine learning-based predictable system-level anomaly detection", Proc. Workshop Secur. Dependability Crit. Embedded Real-Time Syst., (CERTS), 2016.
3. Task aware hybrid DVFS for multi-core real-time systems using machine learning, F.M.M. Islama, M.Lin L.T.Yang K.R. Choo, Information sciences. 2018

Outline

1. Evolution of ML in real-time system

2. Dimensions to distinguish ML papers

3. Relation with my work

My work

1. Find uncertainty and complexity in previous work

- consider influence among set of tasks
- online learning instead of offline learning finally
- predict more instantaneous timing information(instant execution times on the fly of the task set) instead of the WCET bound

2. Use advanced ML techniques

- ANN + RNN (take into account the temporal dependence)

Mining temporal intervals from real-time system traces, Sean Kauffman, Sebastian Fischmeister, SoftwareMining 2017

This work generates the user defined “interesting” interval for real time system traces, which could be the pre-stage of using RNN.

3. The maximumly tangible data input for online learning: time stamps, ordering of task sets, other states of task sets.

1. Huybrechts, T., et al.: A new hybrid approach on WCET analysis for real-time systems using machine learning. In: Brandner, F. (ed.) (WCET 2018)

their method is for embedded system:

-find blocks in code - consecutive, one entry one exit,

-timing measurement on each block

-statically combine measurements

-supervised learning; selected features

Table 1 List of code attributes extracted with the *Feature Selector*.

No. of Additive operations	No. of Multiplicative operations	No. of Division operations
No. of Modulo operations	No. of Logic operations	No. of Bitwise operations
No. of Assign operations	No. of Shift operations	No. of Comparison operations
Return statement present	No. of Evaluation operations	No. of Local variables access
No. of Local array access	No. of Global variables access	No. of Global array access

their purpose is for insight of WCET in early development cycle However, in the early "project definition" stages where there is little or no source code available to perform analysis on.

features – count of operators , count of access

The attributes in this experiment are initially selected by visually inspecting the blocks and identifying which code characteristics would have a significant impact on the execution time. Later, the selection is from a feature pool by a feature selector. The feature selector picks up the features with high correlation with the WCET data.

training feature and data is obtained statically;

(no temporal relationship in these source features and source data)

future work: 1. improve ML model (more model options and polished existing model)

2. feature selection 3. bigger blocks 4. additional features from hardware/toolchain

Results of Paper1

Different ML techniques perform in huge variation on different benchmarks. No one gives uniformly dominant performance over benchmarks.

The experiments are conducted on TACLe-benchmark.

Table 3 Prediction error of the hybrid ML approach on three TACLeBench application for each trained regression model.

Regression models	Bitonic	Bsort	Recursion
Linear Regression	-49.3%	102.2%	-0.2%
Polynomial Regression (2nd Degree)	100.2%	-266.3%	-10.9%
Tree Regression	18.1%	18%	-52.8%
Random Forest Regression	-11.8%	113.7%	-14.6%
Support Vector Regression (Linear Kernel)	-24%	8.5%	-55.3%
Support Vector Regression (RBF Kernel)	-31.9%	-36.6%	-45.6%
K-Nearest Neighbours Regression	-45,9%	38.5%	-54.1%
Ridge Regression	-47,1%	56.8%	0.5%

2. Intelligent Prediction of Execution Times, D Tetzlaff, S Glesner - International Conference on Informatics & Applications (ICIA), 2013

Machine Learning techniques based on supervised learning

Assumption: that there exists a linear relationship between the amount of (classified) machine code instructions to be executed by the function and its execution time. Therefore, they consider linear regression modeling, for which several learning algorithms exist.

- decision trees,
- k-Nearest Neighbor (k-NN),
- Ordinary Least Squares Estimation (OLS),
- the Iterated Reweighted Least Squares (IWLS)
- Support Vector Machine (SVM)
- Predicting Query Run-time 2 (PQR2) technique that is based on a decision tree

TABLE I: Static code features for learning execution times

	No.	Name	Description
Tests	1	bitwise	operations on bits
	2	comp	comparison between data values
Arithmetic	3	calc	integer arithmetic calculations
	4	convert	conversion between integer data types
	5	fcalc	floating point arithmetic calculations
	6	fconvert	conversion between floating point and integer values
Control	7	docall	function calls
	8	param	push actual parameters to stack
	9	jump	jumps in the control flow
Copy	10	addr	load effective address
	11	load	load of values into registers
	12	move	move values between registers
	13	store	storage of values to memory

use static code features of applications

More sorts of and more complex ML methods are used and compared.

Results of Paper2

The Figure 3a does not have a correct unit for y-axis (the error of WCET prediction).

Figure 3b shows none of the prediction algorithms perform accurate enough.

Figure 3c only shows the accuracy of prediction in the training phase. So overall, the result section is problematic.

3. Statistical Prediction of Task Execution Times through Analytic Benchmarking for Scheduling in a Heterogeneous Environment, M.A. Iverson ; F. Ozguner ; L.C. Potter ,Heterogeneous Computing Workshop (HCW'99)

Predict the execution time of tasks in a distributed parallel computing system with heterogeneous computing environments (such as different machines).

Execution time is treated as a random variable and is statistically estimated from past observations. A set of past observations is kept for each machine and used to make new execution time predictions.

KNN is used; Nonparametric Regression

Most Relevant Results of Paper 3

Experiments are conducted on 16 heterogeneous machines. They compare three variations of kNN methods. The best one shows less than 10% average prediction error.

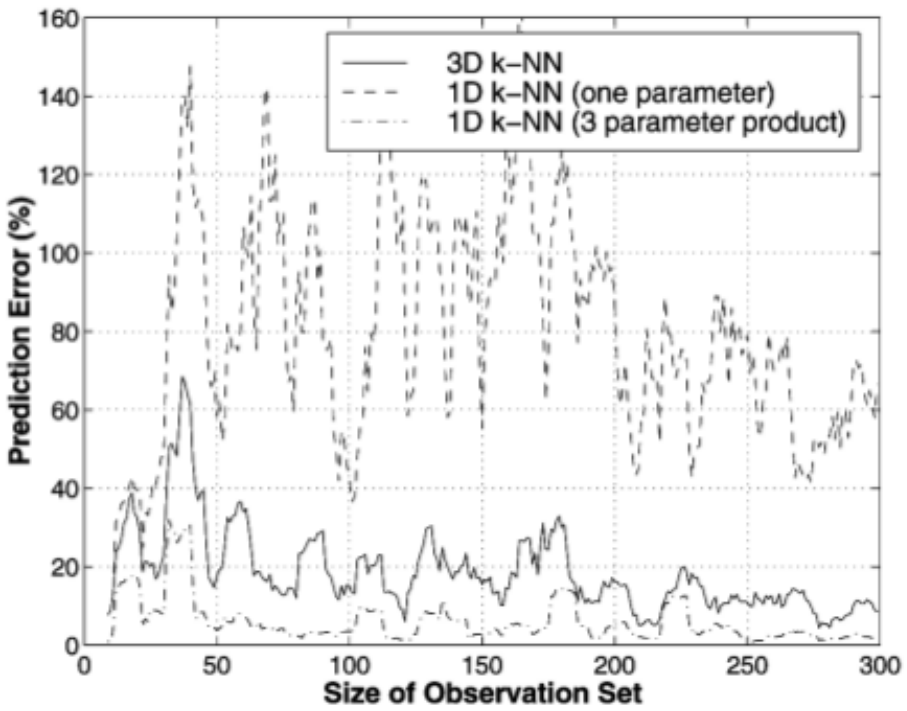

Fig. 3. Prediction error of matrix multiplication algorithm versus size of the observation set.

4. Reliable Estimation of Execution Time of Embedded Software, P Giusto, G Martin, E Harcourt, Design, automation and test in Europe(DATE), 2001

This work models both the target system (CPU instruction set, target compiler, etc.) and the structure of the software program at an abstraction level that makes the estimate of execution time reasonable without losing too much accuracy.

A source-based approach analyzes the original C source code.

Use a stepwise multiple linear regression approach, along with multiple basic linear regression models and correlation analysis.

Results of Paper 4

The static information and linear regression based prediction shows poor performance on the accuracy. The percentage of absolute error is never smaller than 20% on dynamically branching control code benchmarks except for 1% absolute error on FFT benchmark – a mathematical operation dominated benchmark.

5. Accurate Software Performance Estimation Using Domain Classification and Neural Networks, Marcio Seiji Oyamada,Felipe Zschornack, Flavio Rech Wagner, symposium on Integrated circuits and system design (SBCCI 2004)

Use neural network to predict WCET on embedded systems.

Two steps: (1)Domain classification for applications into different NN. This is done by comparing the metric of CFG_weight.

(2) NN: inputs are the number of executed instructions of different instruction types, while the expected result is the number of cycles consumed by the embedded application. A cycle-accurate simulator is required, to extract the number of executed instructions and the total number of cycles consumed by the application. So the data is obtained through simulation first.

Results of Paper 5

The paper trains generic model and domain-specific model based on different division of training dataset. The two models do not show remarkable difference in the accuracy of prediction and neither shows highly reliable prediction Max relative error at least 24%.

Table 1: Estimation results of 41-benchmark set

	Branch, load/store, integer, float	Backward branch, forward branch, load/store, integer, float
Mean error	9.26%	7.90%
Std dev	10.64%	9.11%
Max over	41.01%	33%
Max under	-20.69%	-31%

Table 3: Estimation results using domain-specific estimators

	Domain A	Domain B	Cross-test	
			(A vs B)	(B vs A)
Mean error	7.62%	6.41%	17.65%	55.12%
Std dev	12.46%	9.45%	12.39%	38.25%
Max over	24.96%	25.87%	42.34%	163.78%
Max under	-49.37%	-32.41%	-28.49%	-95.70%

6. Early execution time-estimation through automatically generated timing models, Peter Altenbernd, Jan Gustafsson, Björn Lisper,Friedhelm Stappert, Real-Time Syst Journal(2016)

Predict the execution time of software through an early, source-level timing analysis at the early stage of the development of RTS.

Methods:

(1)It is based on a set of virtual instructions (arithmetic/logic operations, branch, function call/return, etc.) defining an abstract machine able to execute the source code.

(2)A source-level timing model is automatically generated for the given compiler-hardware combination.

(3)The prediction model is linear and consists of fixed costs for the virtual instructions.

Results of Paper 6

The experiment is conducted based on training data collected from GEM5 simulator.

The predictions are accurate (relative error < 10%) in about 2/3 of the benchmarks. The worst accuracy could reach 77%.

Table 3 Predicted versus simulated running times for the benchmark programs (ARM7)

Name	Prediction	Simulation	Overestimation		Underestimation	
bs	4732	4718	14	0%		
cover	7163	7832			669	9%
crc	71555	62271	9284	15%		
duff	7184	6374	810	13%		
fdct	10410	8512	1898	22%		
fibcall	5253	5082	171	3%		
insertsort10	6324	5951	373	6%		
insertsort15	4874	4922			48	1%
insertsort20	4974	5037			63	1%
insertsort30	5174	5267			93	2%
insertsort	6324	5941	383	6%		
janne_complex	5035	4996	39	1%		
jfdctint	10503	9845	658	7%		
loop3	14119	13477	642	5%		
ludcmp	9771	11152			1381	12%
matmult	326289	265830	60459	23%		
minmax	4697	4651	46	1%		
ns	19273	24087			4814	20%
prime	19357	16742	2615	16%		
qsort-exam	6677	6752			75	1%
select	6728	6993			265	4%
sqrt	5235	4970	265	5%		
statemate	5753	6475			722	11%
Average deviation				9%		7%

Table 4 Predicted versus simulated running times for the benchmark programs (ARM4)

Name	Prediction	Simulation	Overestimation		Underestimati	
bs	4721	4682	39	1%		
cover	8192	8019	173	2%		
crc	69730	51855	17875	34%		
duff	7154	6375	779	12%		
fdct	8498	8390	108	1%		
fibcall	5183	5082	101	2%		
insertsort10	6321	6047	274	5%		
insertsort15	4901	4863	38	1%		
insertsort20	5021	4958	63	1%		
insertsort30	5261	5148	113	2%		
insertsort	6321	6037	284	5%		
janne_complex	5042	4990	52	1%		
jfdctint	11461	9366	2095	22%		
loop3	15690	13303	2387	18%		
ludcmp	44594	44537	57	0%		
matmult	376532	232217	144315	62%		
minmax	4633	4650			17	0
ns	25015	20961	4054	19%		
prime	22738	99421			76683	7
qsort-exam	9743	9544	199	2%		
select	9254	8895	359	4%		
sqrt	13795	13156	639	5%		
statemate	5260	5732			472	8
Average deviation				10%		2

7. Nonlinear approach for estimating WCET during programming phase, F.Meng,X.Su, Z. Qu, Cluster Computing (2016)

This paper employs least square support vector machine as nonlinear model to estimate WCET of embedded systems.

This ML model is supervised and based on two sorts of input data features.

(1) static feature in the object code (fingerprints of blocks of object code)

(2) dynamic feature – counts of instructions(static in many other papers) and simulated execution time (simplescalar)

(3) labeled WCET data for training is obtained from simplescalar simulator

They use different metrics of similarity for different groups of sample code as input data in the evaluation:

(1) levenshtein_distance (2) cossine similarity and maxquotient

Results of Paper 7

The results demonstrate that the more similar the testing code is to the training code, the higher accuracy the estimation could achieve.

Table 6 Errors distribution of WCET estimation with different thresholds

error	test1	test2	test3	test4	test5	test6	test7	test8	test9	test10
δ	50	80	80	80	80	0	0	50	80	80
θ	0	0	90	0	99.5	0	99.5	99.5	99.5	99.5
σ	10,000	10,000	10,000	50	10,000	10,000	50	10	10	50
$\leq \pm 10\,\%$	83.17 %	85.86 %	94.51 %	97.65 %	96.61 %	49.15 %	89.02 %	94.74 %	100.00 %	100.00 %
$\leq \pm 15\,\%$	87.13 %	87.88 %	94.51 %	97.65 %	96.61 %	57.63 %	93.90 %	96.49 %	100.00 %	100.00 %
$\leq \pm 30\,\%$	90.10 %	89.90 %	97.80 %	98.82 %	96.61 %	75.42 %	96.34 %	96.49 %	100.00 %	100.00 %
$\geq \pm 100\,\%$	5.94 %	5.05 %	2.20 %	0.00 %	3.39 %	18.64 %	0.00 %	0.00 %	0.00 %	0.00 %

8. Methods for Prediction, Simulation and Verification of Real-Time Software Architectural Design based on Machine Learning Algorithms, Mostafa Anwar Taie, Ibrahim El-Faramawy, Mohamed Elmawazini, SAE International tenchnique report 2015

In this paper, the WCET of OS processes are researched instead of WCET of tasks for embedded systems. Different versions of OS software are also taken into account.

Their work manages the prediction of ETs of multiple OS processes at the same time using different ML models on different levels (process level, group level).

The features are obtained from softwares of previous release. The features include static features of AUTOSAR Application (APP) and Basic Software(BSW) module, seniority of developers, hardware factors and customized features.

Different ML techniques are investigated and compared (KNN, SVM, regression, NN, Extreme Learning Machines).

Features of paper 8

1. No of very long execution time category requirements (vl)
2. No of long execution time category requirements (l)
3. No of medium execution time category requirements (m)
4. No of short execution time category requirements (s)
5. No of very short execution time category requirements (vs)

1. No of deleted very long execution time category requirements (dvl)
2. No of deleted long execution time category requirements (dl)
3. No of deleted medium execution time category requirements (dm
4. No of deleted short execution time category requirements (ds)
5. No of very short execution time category requirements (dvs)

1. Number of NATIVE blocks: The typical type of NV (Non-Volatile) blocks that means stream of bytes saved.
2. Number of REDUNDANT blocks: Used for important data that is stored twice in the NV memory.
3. Number of DATASET blocks: Used for a set of blocks with the same structure.
4. Minimum block size: The smallest block, as block size impacts the execution time.
5. Average block size: The average size for all blocks.
6. Maximum block size: The largest block.
7. Number of blocks using CRC-8: Cyclic Redundancy Check (CRC) is calculated and written with the data in the NV memory, and calculated for consistency check during the read operation, so we count the blocks that use 8-bits CRC for consistency check.
8. Number of blocks using CRC-16: The numbers of blocks that use 16-bits CRC for consistency check.
9. Number of blocks using CRC-32: The numbers of blocks that use 32-bits CRC for consistency check.
10. Number of blocks using write verification: The blocks with write verification consumes more time to perform such verification that the data is written correctly by reading the data after being written then compare to the originally sent data.

Result of Paper 8

Table 3. APP SW Component OS Processes WCET Prediction

Machine Learning Algorithm	SVM		KNN (K=1)		KNN (K=3)		Linear Regression		Additive Regression		BPNN	
	CC	RAE %	CC	RAE %	CC	RAE %	CC	RAE %	CC	RAE %	CC	RAE %
P1	0.9738	20.55	0.8695	47.81	0.8978	40.64	0.9723	21.59	0.9407	35.71	-0.2543	99.83
P2	0.9974	6.87	0.9647	20.71	0.9713	21.47	0.9976	6.48	0.9618	23.28	-0.1666	101.28
P3	0.992	10.84	0.9613	23.09	0.946	30.94	0.9918	11.61	0.9605	26.84	-0.178	101.44
P4	0.9806	19.88	0.967	24.57	0.9653	28.07	0.9862	17.89	0.9527	26.59	-0.1284	99.06
P5	0.9452	32.38	0.7851	60.99	0.8292	50.68	0.9508	29.13	0.8403	54.72	-0.288	100.05
G4 Avg.	0.9778	18.104	0.9095	35.434	0.9219	34.36	0.9797	17.34	0.9312	33.428	-0.2031	100.332

RAE: relative average error; CC: correlation coefficient

Most of the results show high unreliability of using ML to predict WCET.