Hybrid Machine Learning/Analytical Models for Performance Prediction

Diego Didona and Paolo Romano
INESC-ID / Instituto Superior Técnico

6th ACM/SPEC International Conference on Performance Engineering (ICPE)
Feb 1st, 2015
Outline

• Base techniques for performance modeling
  – White box modeling
  – Black box modeling
  – Modeling and optimization on two case studies

• Hybrid modeling techniques
  – Divide et impera
  – Bootstrapping
  – Ensemble

• Closing remarks
Modeling a system

INPUT FEATURES → SYSTEM → KEY PERFORMANCE INDICATORS
Modeling a system

INPUT FEATURES
- Workload:
  - Intensity, small vs large jobs
- Infrastructure
  - # servers, type of servers
- Application-specific
  - Replication

KEY PERFORMANCE INDICATORS
- Throughput
  - Max jobs / sec
- Response time
  - Exec. time of a job
- Consumed energy
  - Joules / job
What is a performance model?

- Approximator of a KPI function
- Relates input to target output
- Can be implemented in different ways
  - White box
  - Black box
Applications of Performance Modeling

• Capacity planning
  – Avoid overload in datacenters

• Anomaly detection
  – Model “normalcy” to detect anomalies

• Self-tuning
  – Maximize performance

• Resource provisioning
  – Elastic scaling in the Cloud
Accuracy of a performance model

• Approximation accuracy metrics
  – MAPE (Mean Absolute Percentage Error)
    \[
    \frac{1}{N} \sum_{i=1}^{N} \frac{|real_i - pred_i|}{real_i}
    \]
  – RMSE (Root Mean Square Error)
    \[
    \sqrt{\frac{1}{N} \sum_{i=1}^{N} (real_i - pred_i)^2}
    \]
White/Black Box Modeling 101
White box performance modeling

💡 Leverage on knowledge about target app’s internals

• Formalize a mapping between
  – Application, hosting platform and
  – Performance

• Formalization can be
  – Analytical (e.g., Queueing Theory) [45]
  – Simulation, e.g., [36]
Queueing Theory

A resource is modeled as a server + a queue

• Possible target KPIs
  – Resource utilization
  – Throughput
  – Response time

• Key factors impacting queue’s performance
  – Arrival of jobs
  – Service demands
  – Service policy (e.g., FCFS)
  – Load generation model (e.g., open vs closed)
From single queues to networks
Queueing Theory pros and cons

- **Accurate for wide spectrum of input parameters**
- **Specifically crafted for target app**
- **Analytical tractability often requires**
  - Assumptions (e.g., independent job flows)
  - Approximations
  - Simplifications (e.g., Poisson arrival)
Simulation

💡 Encode system dynamics via a computer program

• Alternative w.r.t. analytical modeling
  - 🟢 Simpler (code vs equations)
  - 🟢 May rely on less assumptions
  - 🔴 Slower to produce output

🔴 Similar trade-offs w.r.t analytical modeling
  - 🔴 Still uses simplifications to avoid overly complex code
Black box performance modeling

• Definition
• Taxonomy (Offline vs Online, supervised vs unsupervised, regression vs classification)
• Examples (DT, SVM, ANN, KNN, UCB, Gradient),
• Ensemble
• Optimization
Building black box models

💡 Infer performance model from behavior

• Machine Learning [8]
  – Observe Y corresponding to different X
  – Obtain a statistical performance model
Machine Learning pros and cons

- No need for domain knowledge
- High accuracy in interpolation
  - i.e., for input values close to the observed ones
- Curse of dimensionality
  - # required samples grows exp. with input size
  - Long training phase to build model
- Poor accuracy in extrapolation
  - i.e., for input values far away from the observed ones
Black box modeling taxonomy

• Target output feature $y$
  – Classification (discrete $y$) vs Regression ($y$ in $R$)

• Training phase timing
  – Online vs Offline

• Predict or find hidden structures
  – Supervised vs unsupervised learning
OFF-LINE SUPERVISED LEARNING

• Supervised
  – Known inputs $x$ have a corresponding known $y = f(x)$

• Offline
  – Model built on a training dataset
  – Dataset $\{<x,y> : y = f(x)>\}$
  – Learn $f' : f'(x) \sim f(x)$
    • While being able to generalize outside the known dataset
Decision Trees [55]

💡 Predictive model is a tree-like graph
  • Intermediate nodes are predicate
  • Classifications: leaves are classes
  • Regression: leaves are functions
    – Piecewise approximation of nonlinear functions
DT: an example

Input features
• Income range
• Criminal records
• # years in present job
• use credit card
Support Vector Machines [16]

- A tuple is a point in a multidimensional space
- Find hyperplane s.t. different classes are as much distant as possible

Credits to Erik Kim for SVM-related images, http://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html
Support Vector Machines

What if points are not linearly separable?
SVM: the kernel trick, I

💡 Map points to a higher dimensional space

👍 In that space, points are linearly separable

- Here, kernel is $f(x, y) = (x, y, x^2 + y^2)$
SVM: the kernel trick, II

- Nonlinear separation in original domain
Artificial Neural Network [79]

- Inner model is a graph
- Resembles neurons connections in brain
ANN internals

• Neuron structure

• Weighted sum of inputs

• Activation 0/1 function as output

Credits to http://www.theprojectspot.com/tutorial-post/introduction-to-artificial-neural-networks-part-1/7
Building an ANN

• Determining its structure
  – # layers
  – # neurons per layer

• Activation function per neuron

• Iteratively learn weights depending on error
K Nearest Neighbors [2]

💡 Predict based on closest known values to target

- Proximity given by a function

\[ \sqrt{\sum_{i=1}^{k} (x_i - y_i)^2} \]

- Euclidean

\[ \sum_{i=1}^{k} |x_i - y_i| \]

- Manhattan

\[ \left( \sum_{i=1}^{k} (|x_i - y_i|)^q \right)^{1/q} \]

- Minkowski

K Nearest Neighbors

• Classification:
  – Class of X is the most common in neighborhood

• Regression
  – Value for X is a function of the values in the neigb.
ONLINE LEARNING

• We consider Reinforcement Learning [70]
  – Training set not available (nor stored)
  – Given a set of <State, Action> pairs
  Find sequence of actions that maximizes payoff (reward)
  – Collect feedback from system

• Tradeoff between
  – Exploration (try new actions)
  – Exploitation (use good known actions)
Multi-armed bandit (MAB)

• Inspired by gambling at slot machines. Find
  – Which arm to play
  – How many times
  – In which order
Upper Confidence Bound [3]

• Popular set of algorithms for MAB

💡 At any time choose the arm that
  1. maximizes reward, while...
  2. minimizing regret:
     • utility loss due to sub-optimal choices

• Efficiency: regret is logarithmic in the # of trials
Hill Climbing

• Not really “learning”, but online optimization
  Explore function in the direction that increases/decreases its value
• Possibly coupled with randomization to avoid local max/min

NO FREE LUNCH THEOREM FOR ML

• There is no “absolute best learner”

• Best learner and parameters depend on data

• When working in extrapolation, there are no a priori distinctions between learning algorithms [80]
ML optimization

⚠️ A ML algorithm has meta-parameters
   - # features of the input data
   - # min of cases per leaf in DT
   - Kernel and its parameters in SVM
   - Neurons, layers, activation functions in ANN

• How to choose them to maximize accuracy?
  - It depends on the problem at hand!
Features selection [40]

💡 Identify features of inputs that are correlated the most with target output

✅ Speedup in building the model

✅ Increase accuracy by reducing noise
Features selection

• Wrapper: use target ML with different combinations of features
  – Forward selection, Backward elimination, ...

• Filter: independent of the target ML
  – E.g., discard 1 between 2 highly correlated variables

• Dimensionality reduction (PCA, SVD)
  – Find features that account for most of the variance
Hyperparameters optimization

• Find hyper-parameters that maximize accuracy
• Based on cross-validation
  – Use part of the set as training and part as test
• Different approaches
  – Grid search
  – Random search [6]
  – Bayesian optimization [74]
1. Uniformly discretize features’ domain

2. Take the Cartesian product of features
Random search

• Include randomness
  – Increase sampling granularity of important param.
ENSEMBLING

• Solution to counter NFL theorem
• Employ multiple learners together
• Bagging [9]
  – Train learners on different training sets
• Boosting [66]
  – Generate 1 strong learner from N weak ones
• Stacking [79]
  – Combine output of learners depending on input
Bagging

💡 Average output of sub-models

- Generate $N$ sets of size $D'$
  - Draw uniformly at random with repetition from $D$
- Generate $N$ black box models
  - Voting for classification
  - Averaging for regression

- Cannot improve predictive power (in extrapol.) ...
- Can reduce variance (i.e., better interp. accuracy)
Bagging example

- 100 bootstrapped learners
- Reduce variance and overfit w.r.t. single models
Boosting

💡 Build a strong learner from many weak ones

• Stage-wise training phase
  – Training at stage i depends on output of i-1

• 0/1 Adaboost
  – Base learners $B_i$: can classify correctly with $p > \frac{1}{2}$
  – Iteratively try to classify better mis-classified samples
  – At stage i, drawn training set according to dist. $D_i$
  – $D_{i+1}$ s.t. mis-classified samples have higher relevance
  – Output weighted average of weak learners
Adaboost, training

Credits to Kihwan Kim. http://www.cc.gatech.edu/~kihwan23/imageCV/Final2005/FinalProject_KH.htm
Adaboost, result

\[ H(x) = \text{sign}(\alpha_1 h_1(x) + \alpha_2 h_2(x) + \alpha_3 h_3(x)) \]

A meta-learner combines output of ML

- Partition D into D', D''
- Train 1...N learners on D'
- $\text{ML}_{N+1}$ trained on $\text{ML}_1$,...,$\text{ML}_N$ predictions on D''
Introduction and Modeling of Main Case Studies
Background Case studies

• Total Order Broadcast primitive
  – Analytical model
  – Black box online optimization

• Distributed NoSQL transactional data grid
  – Simulation model
  – Black box offline supervised learning
Total Order Broadcast case study

• TOB allows a set of nodes to deliver broadcast messages in the same order

• Incarnates the popular consensus problem
  – Fundamental abstraction for dependable computing

• We consider Sequencer-based TOB
  – Messages are broadcast normally
  – A Sequencer node decides the delivery order
Sequencer-Based TOB

Broadcast messages

Broadcast Seq. No.

Ordered deliver

N₀ (Sequencer)

N₁

N₂

M₁, M₂

M₁, M₂

M₁
Performance of STOB

- STOB minimizes messages exchange, but...
- The sequencer may become the bottleneck
- Possible solution: batching
- The sequencer
  - Waits to receive $N > 1$ msgs
  - Send a single, bigger seq. msg for the $N$ msgs instead of $N$ smaller
Batching in STOB

• At high load batching
  – Allows for amortizing msgs sequencing cost
  – Increases sequencer capacity and throughput

• At load load batching
  – Introduces useless delays
  – The sequencer waits too much and wastes time
The need for self-tuning STOB Batching

- Optimal batching depending on msgs rate
Tuning the batching level

• White box approaches
  – Forecast the impact of batching given workload

• Black box approaches
  – On-line optimization
STOB white box modeling

- Focus on performance on sequencer
- It is representative of the whole system
STOB model input

- $m =$ messages generation rate
- $b =$ batching level
- $T_{1} =$ time to process $1^{st}$ message in batch
- $T_{\text{Add}} =$ time to process additional msgs
  - Batching makes sense when $T_{1}>T_{\text{Add}}$
STOB analytical model [59]

• Sequencer = M/M/1 queue

\[ T(b, m) = \frac{1}{\mu(b, m) - \lambda(b, m)} \]

• Batch generation rate

\[ \lambda(b, m) = \frac{m}{b} \]

• Batch service rate

\[ \mu(b, m) = \frac{1}{T_{1st} + \frac{(b-1)}{2m} + T_{add}(b-1)} \]

• Taking derivatives, optimal b is computed
STOB model’s accuracy

• Assumptions and simplifications
  – Exponential arrival rate and service rate (M/M/1)
  – In computing arrivals and computation overlapping
STOB black box optimization [24]

• Learn optimal waiting time for a batch of size $b$
  – Computed at the sequencer

• Hill climbing for each value of $b$
  – In/decrease wait time @$b$ depending on feedback

• When delivering a batch of size $b$
  – Confirm previous decision if delivery time is lower
  – Revert previous decision if delivery time is higher
• But limited expressiveness:
  – Self-tuning at the cost of no predictability
Transactional NoSQL store case study

- Distributed transactional data store
  - Nodes maintain elements of a dataset
    - Full vs partial replication (# copies per item)
  - Transactional -- ACI(D) -- manipulation of data
    - Concurrency control scheme (enforce isolation)
    - Replication protocol (disseminate modifications)
Replication protocols: which one?

Transactional data

Consistency protocols

Single master (primary-backup)

Multi master

Total order-based

2PC-based

State machine replication

Certification

Non-voting

Voting

BFC

No one size fits all solution
DSTM Performance

- Heterogeneous, nonlinear scalability trends!
Factors limiting scalability

Network latency in commit phase

Aborted transactions because of conflicts
White box modeling

Client
- begin
- put
- get
- commit
- begin_return
- put_return
- get_return
- commit_return
- abort
- next_tx (for open systems only)

Server
- Transaction Manager (TM)
- CPU
- Concurrency Control (CC)
- Distribution Manager (DM)

Events:
- Event
- Function call

From other cache servers:
- prepare_reply
- read_done
- commit_done

To other cache servers:
- remote_get
- remote_prepare
- abort
- commit
- CPU_complete
Simulator [21]

• Assumptions and approximations
  – CPU = G/M/K
  – Fixed point to point network latency

• Accuracy / resolution time trade-off
Black box modeling

- MorphR [20]
  - Automatic switching among replication protocols
- Decision tree classifier (C5.0)
- Workload characterization
  - Xact mix, #ops, throughput, abort rate
- Physical resource usage
  - CPU, memory, commit latency
- Output: optimal replication protocol
MorphR in action

Throughput (committed tx/sec)

Time (minutes)

TW1 TW2 TW3

PB 2PC TOB MorphR
Gray Box Modeling
Gray box modeling

• Combine WB and BB modeling
  – Lower training time thx to WBM
  – Incremental learning thx to BBM

• Techniques in this tutorial
  – Divide et impera
  – Bootstrapping
  – Hybrid ensembling
Gray box modeling

• Techniques in this tutorial
  – Divide et impera
  – Bootstrapping
  – Hybrid ensembling
Divide et impera

💡 Modular approach
  – WBM of what is observable/easy to model
  – BBM of what is un-observable or too complex

• Reconcile their output in a single function

👍 Higher accuracy in extrapolation thx to WBM
👍 Apply BBM only to sub-problem
  – Less features, lower training time
NoSQL optimization in the Cloud

• Important to model network-bound ops but…

행위 실패
Cloud hides detail about network 😞
   – No topology info
   – No service demand info
   – Additional overhead of virtualization layer

💡 BBM of network-bound ops performance
   – Train ML on the target platform
TAS/PROMPT [28,30]

• Analytical modeling
  – Concurrency control scheme
    • E.g., encounter time vs commit time locking
  – Replication protocol
    • E.g., PB vs 2PC
  – Replication scheme
    • Partial vs full
  – CPU

• Machine Learning
  – Network bound op (prepare, remote gets)
  – Decision tree regressor
Analytical model in TAS/PROMPT

• Concurrency control scheme (lock-based)
  – A lock is a M/G/1 server
  – Conflict prob = utilization of the server
• Replication protocol
  – 2PC: all nodes are similar → one model
  – PR: primary vs backups → two models
• Replication scheme
  – Probability of accessing remote data
  – # nodes involved in commit
Machine Learning in TAS/PROMPT

- Decision tree regressor
- Operation-specific models
  - Latency during prepare
  - Latency to retrieve remote data
- Input
  - Operations rate (prepare, commit, remote get...)
  - Size of messages
  - # nodes involved in commit
ML accuracy for network bound ops

Seamlessly portable across infrastructures
– Here, private cloud and Amazon EC2
AM and ML coupling

At training time, all features are monitorable.

At query time they are NOT!

EXAMPLE

- Current config: 5 nodes, full replication
  - Contact all 5 nodes at commit

- Query config: 10 nodes, partial replication
  - How many contacted nodes at commit??
Model resolution

💡 AM can provide (estimates of) missing input
  • Iterative coupling scheme

ML takes some input parameters from AM

AM takes latencies forecast by ML as input parameter
TOP: PB, only master node. BOTTOM: 2PC. FULL REPL.
YCSB (transactified) workloads while varying
- # operations/tx
- Transactional mix
- Scale
- Replication degree

Mean relative error

Percentage of additional training set

0 0.1 0.2 0.3 0.4 0.5 0.6 0.7

Cubist
M5R
SMOReg
MLP
Divide et Impera
COMPARISON WITH PURE BLACK, II

- ML trained with TPCC-R and queried for TPCC-W
- Pure ML blunders when faced with new workloads
TAS/PROMPT integration

- TAS/PROMPT are baseline AM for case studies

⚠️ We will use TAS/PROMPT as pure white AM
  - Trained with fixed network model
  - i.e., we do not retrain it as new data are collected
    (But it is possible)
  - Representative of pure white box models
Gray box modeling

• Techniques in this tutorial
  – Divide et impera
  – Bootstrap
  – Hybrid ensembling
BOOTSTRAPPING [27]

💡 Obtain zero-training-time ML via initial AM

1. Initial (synthetic) training set of ML from AM
2. Retrain periodically with “real” samples
How many synthetic samples?

- **Important tradeoff**
  - Higher # → lower fitting error over the AM output
  - Lower # → higher density of real samples in dataset
How to update

• Merge: simply add real samples to synthetic set

• Replace only the nearest neighbor (RNN)

• Replace neighbors in a given region (RNR)
  – Two variants
Real vs AM function
Real vs learnt

• Assuming enough point to perfectly learn AM
• Add real samples to synthetic
• Problem: same/near samples have diff. output
Replace Nearest Neighbor (RNN)

- Remove nearest neighbor
Replace Nearest Neighbor (RNN)

- Preserve distribution...
Replace Nearest Neighbor (RNN)

• ... but may induce alternating outputs
Replace Nearest Region (RNR)

- Add real and **remove** synth. samples in a radius
Replace Nearest Region (RNR)

- $R = \text{radius defining neighborhood}$
Replace Nearest Region (RNR)

• $R = \text{radius defining neighborhood}$
Replace Nearest Region (RNR)

- Skew samples’ distribution
Replace Nearest Region 2 (RNR2)

- **Replace** all synthetic samples in a radius R
Replace Nearest Region2 (RNN2)

- Maintain distribution, piecewise approximation
Weighting

• Give more relevance to some samples

👍 Fit better the model around **real** samples
  – “Trust” **real** samples more than synthetic ones
  – Useful especially in Merge

👎 Too high can cause over-fitting!
  – Learner too specialized only in some regions
• Weighting more real samples reduces error
• Examples of over-fitting
MERGE VS REPLACE (TOB)

- Assuming optimal parameterization
- Merge and Replace seem *very* similar...
Impact of base model (TOB)

- ... BUT replace is better if base model is poor
  - It evicts synthetic samples more aggressively
Visualizing the correction (STOB)

BASE MODEL

PURE ML (70% TS)

BOOTSTRAPPED ML (70% TS)
Visualizing the correction (KVS)

**BASE MODEL**

![Graph showing write percentage and absolute percentage error for different numbers of nodes.]

**PURE ML (70% TS)**

![Graph showing write percentage and absolute percentage error for different numbers of nodes.]

**BOOTSTRAPPED ML (70% TS)**

![Graph showing write percentage and absolute percentage error for different numbers of nodes.]

Legend:
- Write percentage
- Absolute Percentage Error
- Number of nodes
BOOTSTRAPPING in RL [59]

- Optimize batching level in STOB
- Base AM already presented
Hybrid RL in STOB

• UCB: find optimal batch size ($b^*$) for a given msg. arrival rate ($m$)
  – Discretize $m$ domain into $M=\{m_{\text{min}}...m_{\text{max}}\}$
  – A UCB instance for each $m_i$
  – For each instance, a lever for each $b$

• Initial rewards are determined via AM
  – Convergence speed of UCB insufficient at high arr.:
    • Enhance convergence speed using initial knowledge of AM
Bootstrapped model

- Enhance response time by better batching
- Faster convergence than UCB (& no thrashing)
Gray box modeling

- Techniques in this tutorial
  - Divide et impera
  - Bootstrapping
  - Hybrid ensembling
Hybrid Ensemble [26]

- Combine output of AM and ML
  - Hybrid boosting: correct errors of single models
  - KNN: select best model depending on query
  - Probing: train ML only where AM is not accurate
Hybrid Boosting

• Implements Logistic Additive Regression

• Chain composed by AM + cascade of ML

• $ML_1$ trained over residual error of AM

• $ML_i$, $i > 1$ trained over residual error of $ML_{i-1}$
• Chain of 3 BBMs (> 3 were useless here)
  – DT, ANN, SVM
Online variant of HyBoost

• Self-correcting Transactional Auto Scaler (SC-TAS) [28]

identifying optimal level of parallelism in a distributed NoSQL transactional store
  – # nodes in the platforms
  – # threads active on each node
Parallelism tuning in DTM

Why not using a simpler exploration based approach, e.g. hill-climbing?

- Adapting number of threads per node is simple and effective
- Changing # nodes is costly: state transfer!

Model-based solution

- Input: workload, # nodes, # threads/node
- Output: throughput

Obtain: highest-throughput configuration

Final Workshop of the Euro-TM COST Action, Amsterdam, 2015 Jan. 19th
Implemented solution: SC-TAS

💡 Exploration + modeling + Machine Learning
1. Explore to gather feedback on model’s accuracy
2. **LEARN** corrective functions to “patch” model

- Try to avoid global reconfiguration (# nodes)
  - Rely on local # threads exploration (cheap)

Increase accuracy

Final Workshop of the Euro-TM COST Action, Amsterdam, 2015 Jan. 19th
SC-TAS control loop

- **Local** scaling
- **Global** scaling

- Explore locally
- End Exploration?
- Update SC-TAS

- Collect Data
- Invoke SC-TAS

- YES
- NO

- Workload, #thread, #nodes, TAS’ error
SC-TAS control loop

1. Collect Data
2. Update SC-TAS
3. End Exploration?
   - NO: Explore locally
   - YES: Invoke SC-TAS

Local scaling
Global scaling

• Re-train hyboost “patching” ML
SC-TAS control loop

- Collect Data
  - Local scaling
    - Explore locally
  - Update SC-TAS
    - NO
      - End Exploration?
        - YES
          - Global scaling
          - Invoke SC-TAS
    - YES

- Yes if min<#thread exploration<max &&
- Accuracy of the patched model considered “OK”
SC-TAS control loop

- Local scaling
- Explore locally
- Collect Data
- Update SC-TAS
- End Exploration?
- YES: Invoke SC-TAS
- NO: Global scaling
- Change # of threads and repeat

- Patch is not enough
SC-TAS control loop

- Model is **supposedly** patched
- Invoke the patched model
Dynamics of SC-TAS

The graph shows the dynamics of Absolute Relative Error over the Number of iterations for different values of $\mu$ (minimum number of thread explorations per node). Specifically:

- $\mu = 3$ (solid black line)
- $\mu = 5$ (solid red line)
- $\mu = 7$ (solid blue line)

The data indicates that as the number of iterations increases, the Absolute Relative Error decreases, suggesting improved accuracy. The self-correcting capabilities of SC-TAS can benefit both in optimizing the multiprogramming level of DTM applications and in applications such as QoS-driven elastic scaling policies and what-if analysis of the scalability of DTM applications, requiring speculation on performance in different scale settings.

**Conclusion**

In this paper, we proposed and evaluated algorithms aimed to self-tune the multiprogramming level in two radically different types of TM systems: a shared memory STM and a distributed STM. We showed that for shared memory, a simple exploration-based hillclimbing algorithm can be extremely effective, even when faced with challenging workloads. However, in the distributed case, pure exploration-based approaches are no longer viable, as testing configurations with a different number of nodes requires triggering costly state transfer phases. We tackled this problem by introducing a novel hybrid approach that combines performance models and local exploration in order to achieve the best of the two methodologies: quick convergence towards the global optimum and robustness to possible inaccuracies of the performance models.

**References**

SC-TAS: before and after
Hybrid Ensemble [26]

• Combine output of AM and ML

  – Hybrid boosting: correct errors of single models

  – KNN: select best model depending on query

  – Probing: train ML only where AM is not accurate
Hybrid KNN

• Split D into D’, D’’

• Train ML₁…MLₙ on D’
  – ML can differ in nature, parameters, training set...

• For a query sample x
  – Pick the K training samples in D’’ closest to x
  – Find the model with lowest error on the K samples
  – Use such model to predict x
KNN, sensitivity (TOB)

- Low cut-off && low % training $\rightarrow$ collapse to AM
- High cut-off && high % training $\rightarrow$ collapse to ML
Hybrid Ensemble [26]

• Combine output of AM and ML
  – Hybrid boosting: correct errors of single models
  – KNN: select best model depending on query
  – Probing: train ML only where AM is not accurate
PROBING

💡 Build a ML model as specialized as possible
  – Use AM where it is accurate
  – Train ML only where AM fails

⚠ Differences with KNN
  – In KNN, ML is trained on all samples:
    • Here, only when AM is found to be inaccurate
  – In KNN, voting decides on ML vs AM:
    • Here, binary classifier determines in *which regions* the AM is inaccurate
Probing at work

1. $D_{\text{ML}} = \text{empty set}$

2. Train a classifier: for each $x$ in $D$
   - If error of $\text{AM}$ on $x > \text{cut-off}$, map $x$ to $\text{ML}$ and add $x$ to $D_{\text{ML}}$
   - Else map $x$ to $\text{AM}$

3. Train $\text{ML}$ on $D_{\text{ML}}$

• QUERY for input $z$
  - If classify($z$) = $\text{AM}$, return $\text{AM}(z)$; else return $\text{ML}(z)$
Probing Sensitivity (KVS)

- High cut-off $\rightarrow$ collapses to AM
- Low cut-off $\rightarrow$ collapses to ML
NFL strikes again

- No one-size-fits-all hybrid models exist!
- Choosing best hybrid model (with right parameters) can be cast to a parameter optimization problem
Concluding remarks

WBM and BBM often conceived as antithetic

They can be leveraged on synergistically

– Increased predictive power thx to WBM
– Incremental learning capabilities thx to BBM

Gray box approaches

– Divide et impera, Bootstrapping, Hybrid ensembling
– Design, implementation and use cases

Can deliver better accuracy than pure B/W
THANK YOU

Questions?

didona@gsd.inesc-id.pt

www.gsd.inesc-id.pt/~didona
Hybrid Machine Learning/Analytical Models for Performance Prediction: Bibliography

Diego Didona and Paolo Romano
INESC-ID / Instituto Superior Técnico, Universidade de Lisboa

White box performance modeling: principles, applications and fundamental results.
[45] [49] [50] [71] [46] [4] [39] [44] [58] [51] [36]
Principles of Machine Learning.
[8][5][80] [65][66][48][34][9][79][10][70][41][3][77][62] [16][33][54] [55] [47]
[56][53] [42] [76] [2]
ML ensembling, features selection and hyper-parameter optimizations.
[32] [12][74] [6] [69] [40]
Application of ML to performance modeling.
[13] [60][18][20][15] [59][31][75][38][37][68] [1][82][81] [57]
Divide et impera.
[30] [43][28] [22]
Bootstrapping.
[73] [59] [67] [72] [61] [27]
Hybrid ensembling.
[26] [25] [14]
Case studies: introduction and performance modeling/optimization.
[11] [35] [63] [24] [18] [21] [52] [7] [29] [17] [19] [23] [64] [20] [78]
[83] [84] [85] [86] [87] [73]
References


[31] Nuno Diegues and Paolo Romano. Self-tuning intel transactional synchronization extensions. In 11th International Conference on Autonomic Com-


