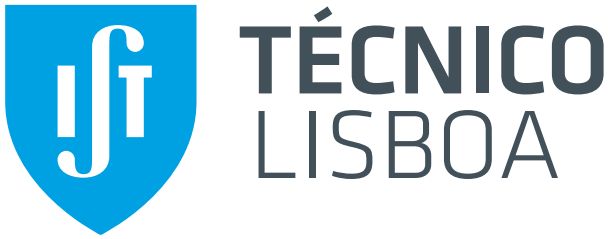


Self-tuning of complex computing systems: reconciling analytical modelling & machine learning

Paolo Romano

INESC-ID

Instituto Superior Técnico, Lisbon University

Autonomic computing: the inception

- “Dealing with complexity is the single most important challenge facing the IT industry”, IBM VP Research, **2001**
  - *60%-75% of databases’ TCO in spent in administration*
  - *40% outages caused by (skilled) humans’ errors*
  - *30%-50% of IT budget spent preventing or recovering from crashes*
  - *IT labour costs exceed equipment costs by up to 18:1*

P4 1.3GHZ ~500x less transistors than 2018 Xeon
High-end system specs: 256MB RAM, 40 GB HD

Autonomic computing: the inception

- “Dealing with complexity is the single most important challenge facing the IT industry”, IBM VP Research, **2001**

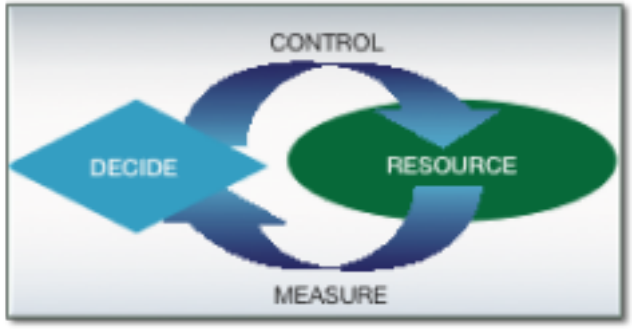
- Solution inspired to *autonomic nervous system*:
  - free *conscious brain* from low-level tasks (breathing, heating, etc…)

Autonomic computing: the inception

- "Dealing with complexity is the single most important challenge facing the IT industry", IBM VP Research, **2001**

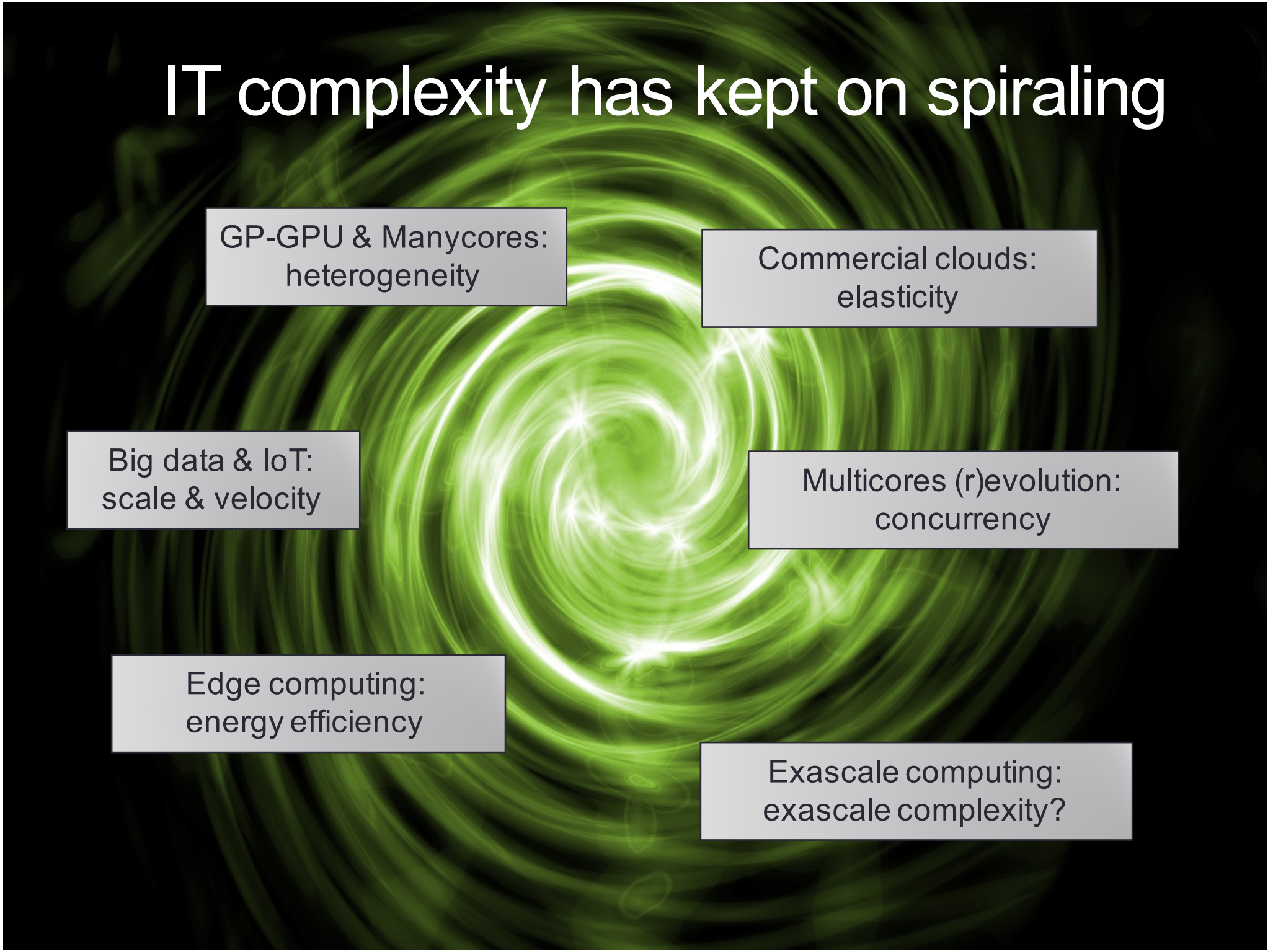
- Solution inspired to *autonomic nervous system*:
  - free *conscious brain* from low-level tasks (breathing, heating, etc…)

- Tame IT complexity via self-*:
  - express system's behavior via high-level policies
  - pursue these goals via automatic control loops



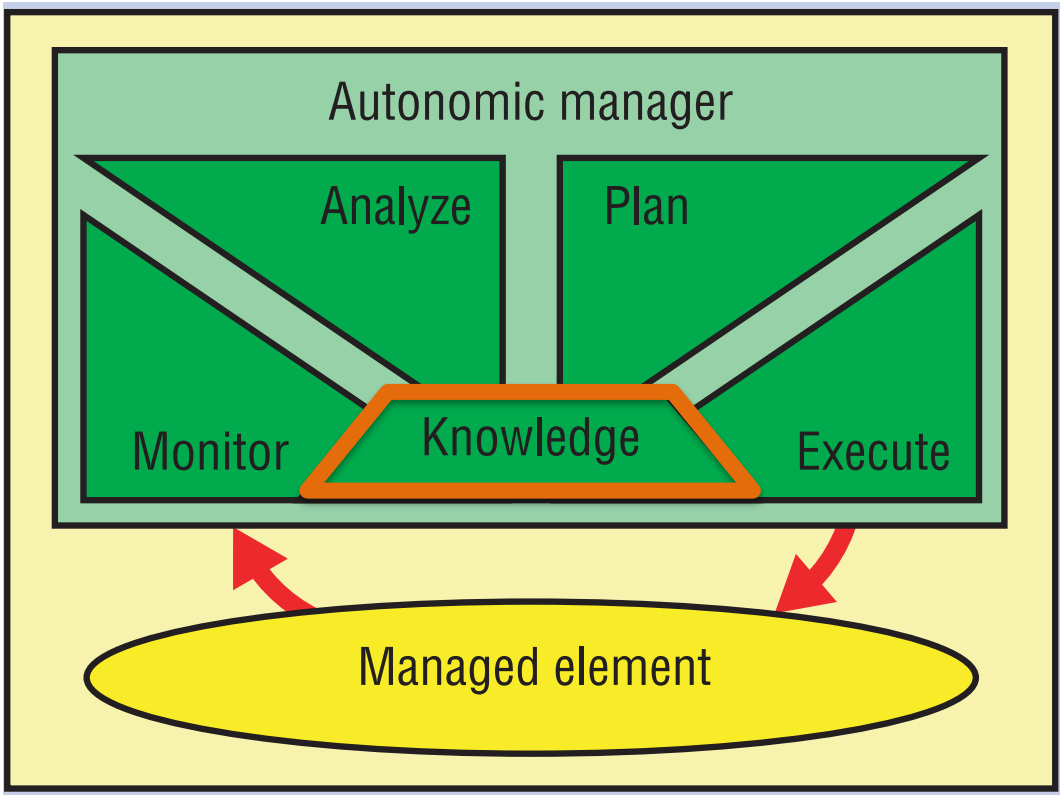
What happened next?

IT complexity has kept on spiraling

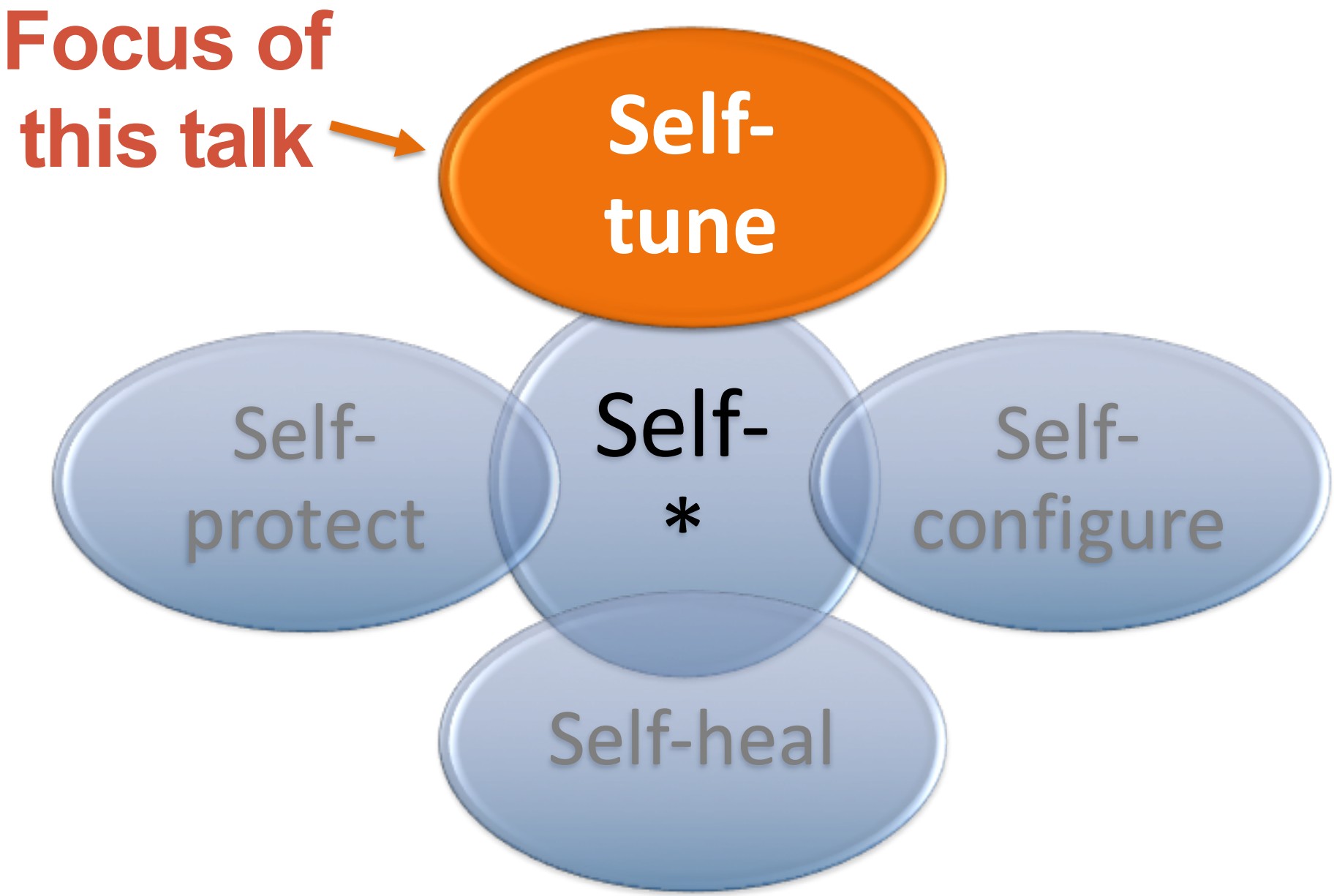


Today, autonomic computing is a key tool to cope with IT complexity

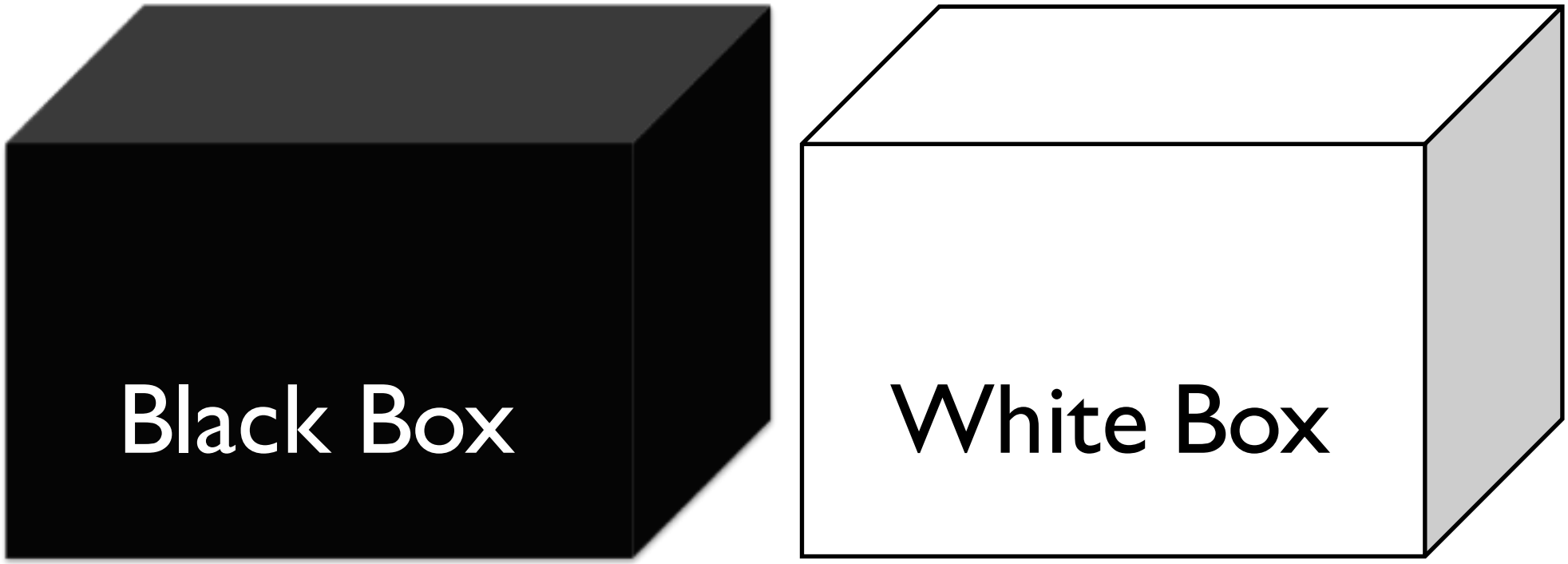
Google Cloud Platform



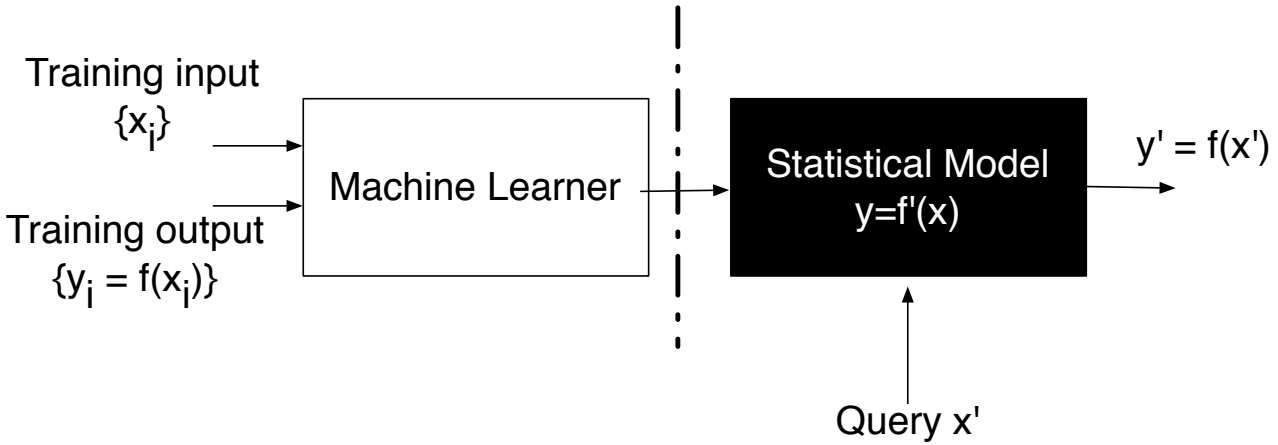
Surge of IT complexity challenges our ability to **model** their behavior


Focus of this talk
Self-tune
Self-protect
Self-*
Self-configure
Self-heal

Approaches to (performance) modelling of computing systems



Black box modelling



PROS

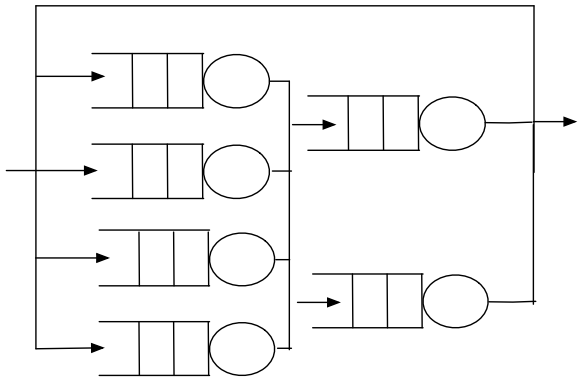
- High accuracy in areas already observed (interpolation)
- No knowledge on system's internals

CONS

- Poor accuracy in non-observed areas (extrapolation)
- Curse of dimensionality
  - Extensive training phases

White box modelling

- Exploit knowledge on internal system dynamics
  - ✧ model dynamics analytically or via simulation



PROS

- Minimal or no learning phase
- Good extrapolation power

CONS

- Simplifying assumptions
  - ➢ reduced accuracy
- Knowledge of system internals often unavailable

Key Observation & Questions

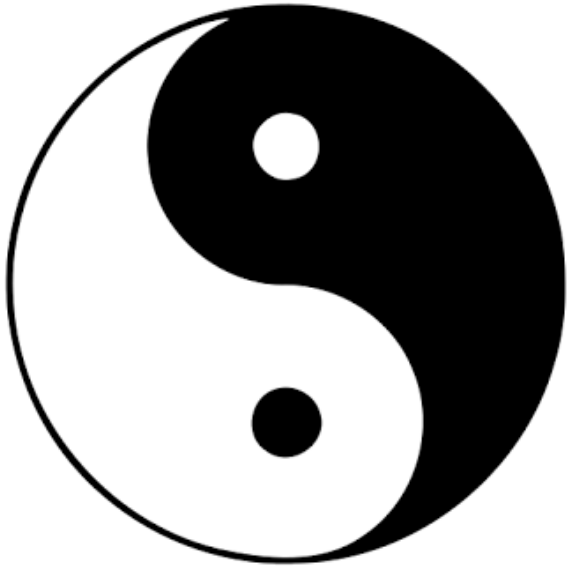
Pros of white-box are cons of black-box & vicev.

Can we achieve the best of the two worlds?

Can black and white box modelling be reconciled ?

Gray box modeling

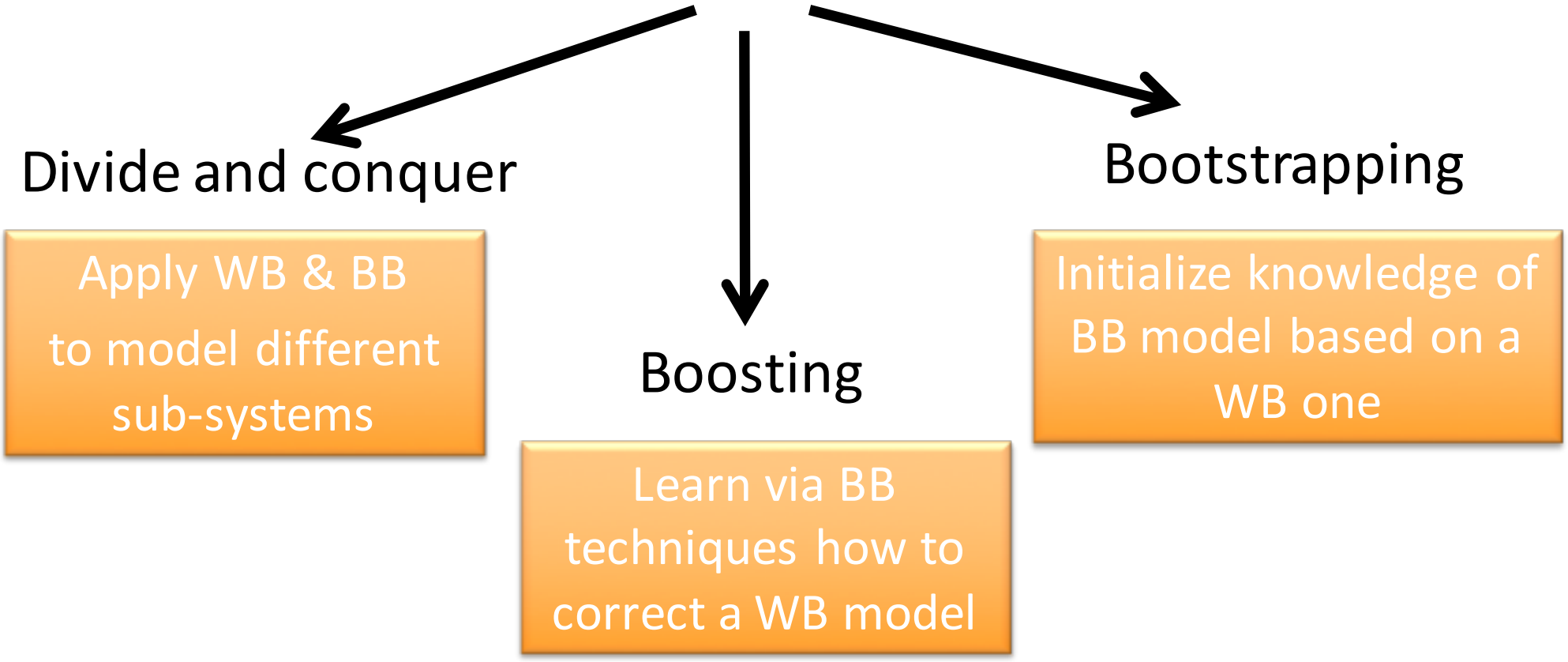
- Combine WB and BB modeling



- Enhance **robustness & reduce cost**
  - Lower training time & cost thanks to White box models
  - Incremental learning thanks to Black box techniques

Gray box modeling

- I will present three methodologies:

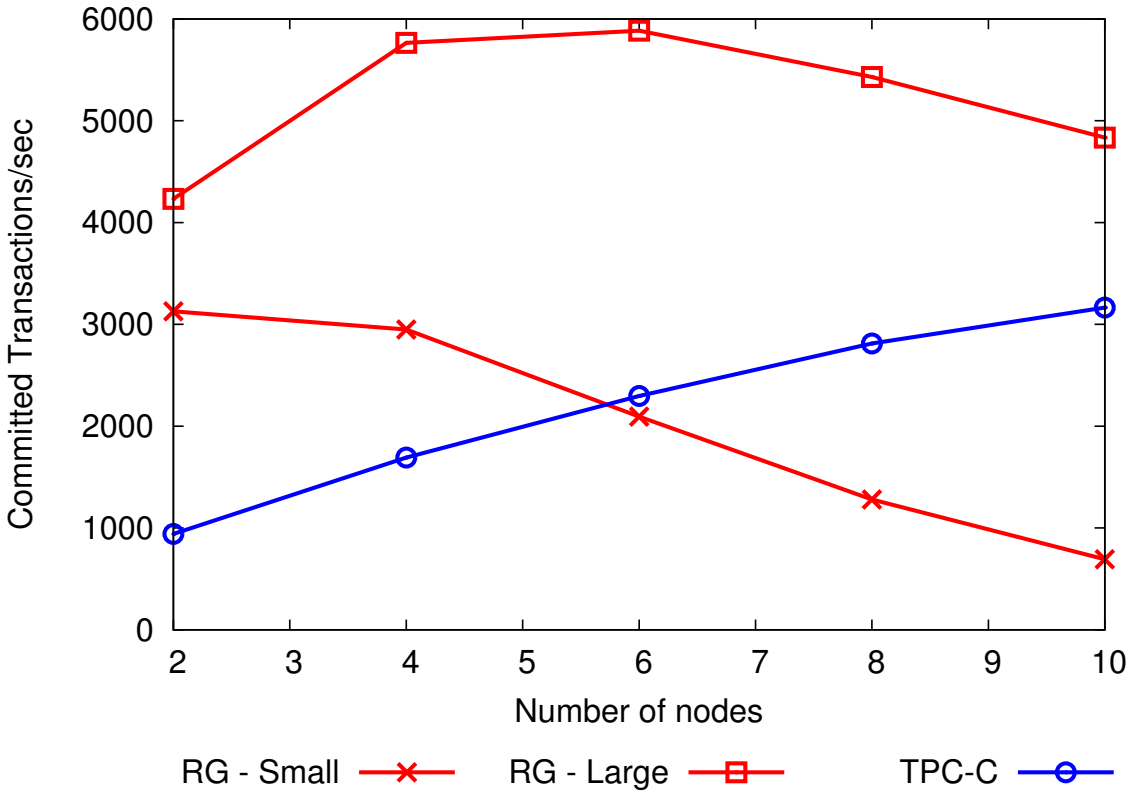


Case study:
Self-tuning of Transactional Data Grids

Infinispan

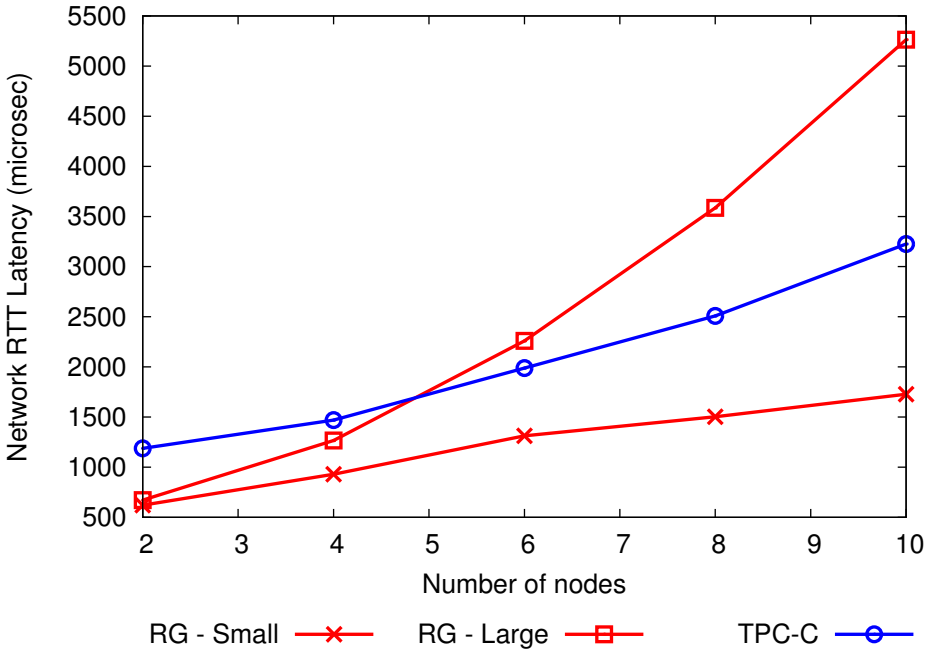
- In-memory transactional data-grid:
  - Data scattered across elastic distributed platform
    - Full vs partial replication
  - Transactional --ACI(D)– manipulation of data
  - Pervasive support for dynamic reconfiguration:
    - elastic scaling, data placement, replication protocol, locking strategy,…

Transactional Data Grids: performance

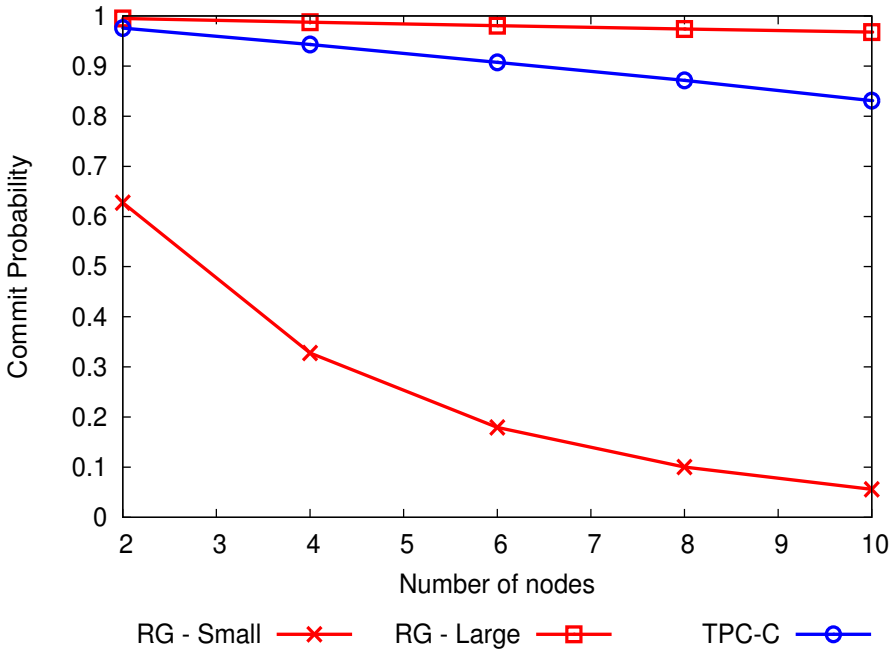


- Heterogeneous, nonlinear scalability trends!

Factors limiting scalability



Network latency in commit phase



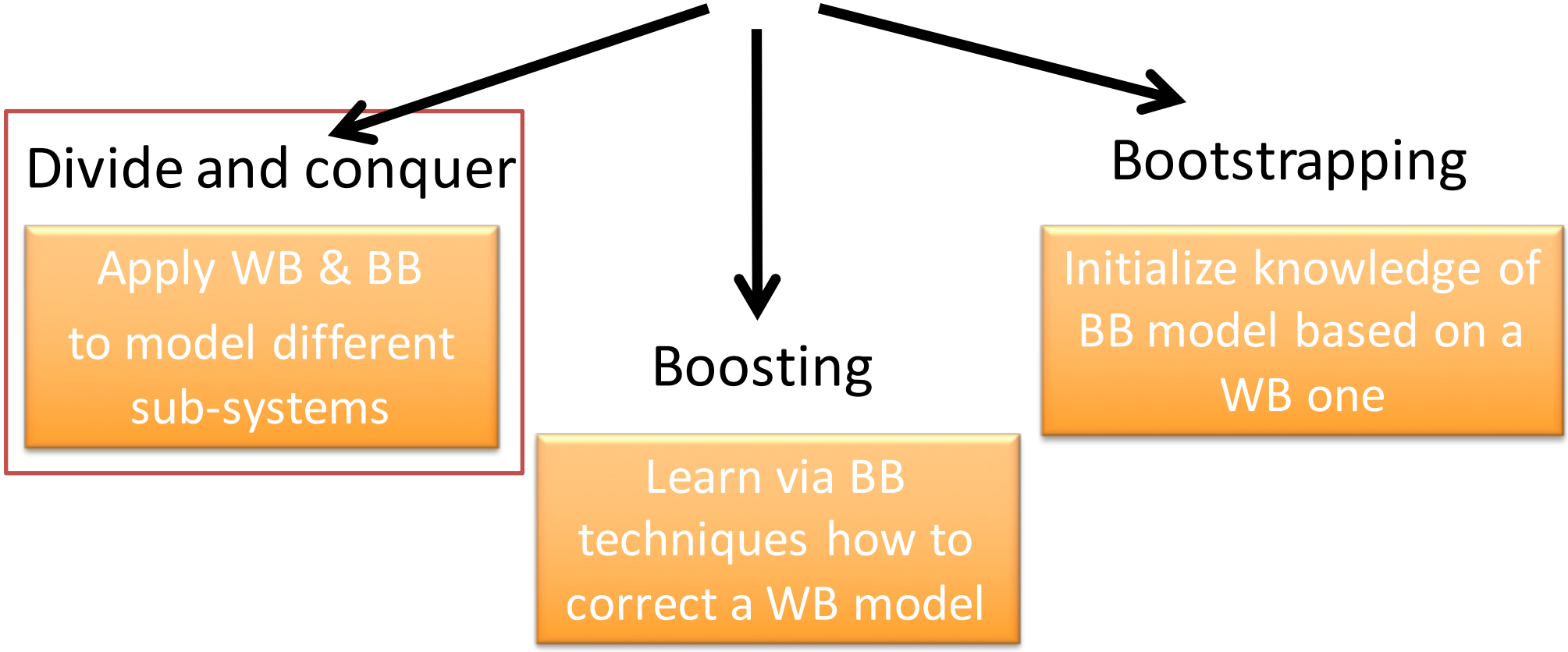
Aborted transactions because of conflicts

Transactional Data Grids: a forge of self-tuning problems

- Scale up and/or scale out [SEAMS18, ASPLOS16, ICPE15]
  - how many machines should my DTM be provisioned with?
  - how many threads should be active on each machine?
- Which distributed synchronization scheme [TPDS14]
  - single- vs multi-master, optimistic vs pessimistic
- Tuning of data replication and group communication layers:
  - quorum sizes [Middleware15], message batching [SASO12]
- Data placement [TAAS14]:
  - where should data and code be placed to maximize locality?

Gray box modeling

- I will present three methodologies:



Divide and conquer

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 via WBM

Apply BBM only to sub-problem

- Less features, lower training time & cost

Self-tuning (data grids) in the cloud: the partial observability problem

- Important to model network-bound ops but…

Cloud hides detail about network

  - No topology info
  - No load info
  - Additional overhead of virtualization layer

BBM of network-bound ops performance

  - Train ML on the target platform

TAS/PROMPT [TAAS14,Mascots14]

- Analytical modeling (queuing theory based)
  - Concurrency control scheme
    - encounter time vs commit time locking
  - Replication protocol
    - multi-master (2PC) vs single-master (Primary Backup)
  - Replication scheme
    - Partial vs full
  - CPU
- Machine Learning (Decision tree regressor)
  - Latency network bound operations (prepare, remote gets)
  - Inputs: operation rates, #nodes involved in commit

AM and ML coupling


At ML training time, all features are known


At query time they are NOT!


EXAMPLE

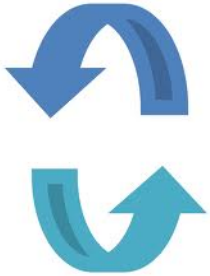
- Current config: 5 nodes, full replication
  - Contact all 5 nodes at commit
- Query config: 10 nodes, partial replication
  - How many contacted nodes at commit?

Model resolution

AM can provide (estimates of) missing input

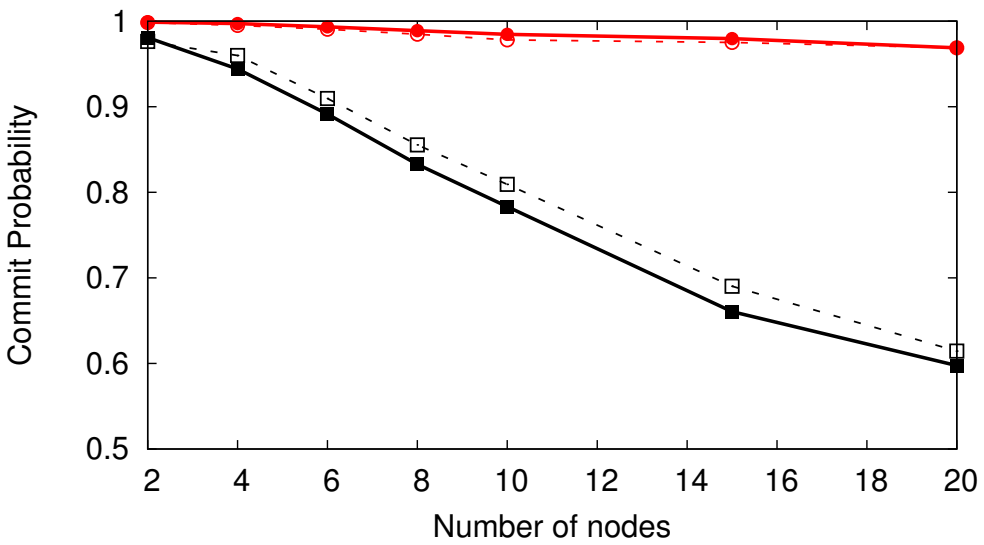
Recursive coupling scheme

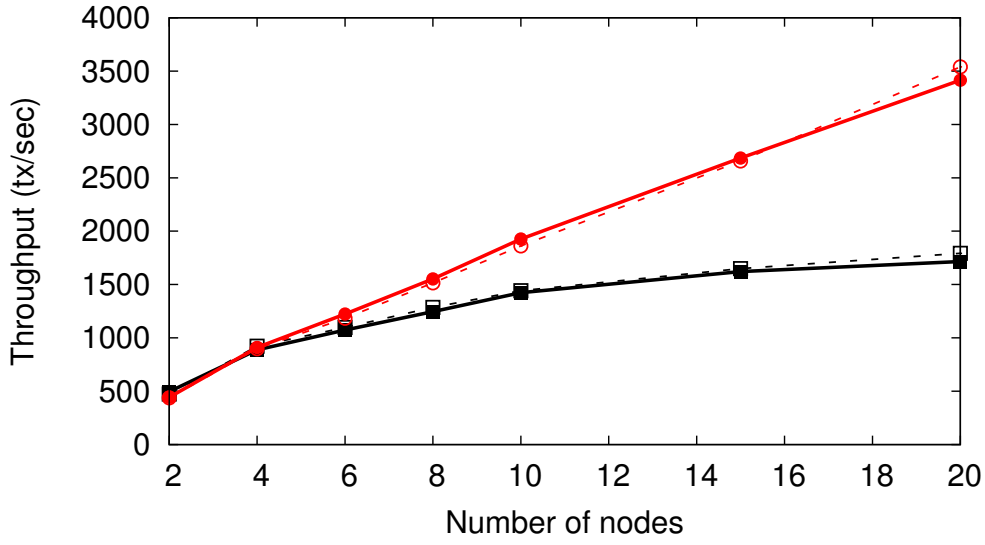
ML predicts network latencies based on AM inputs

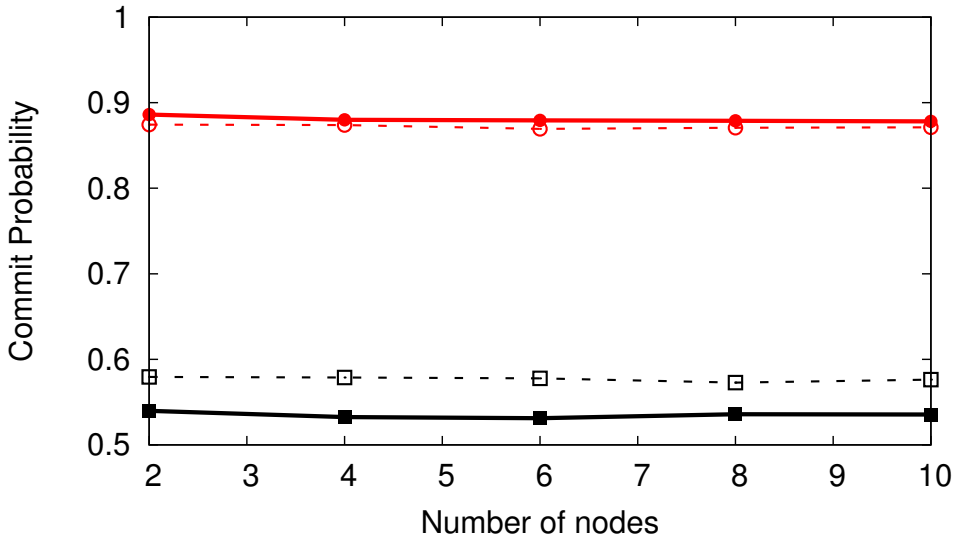


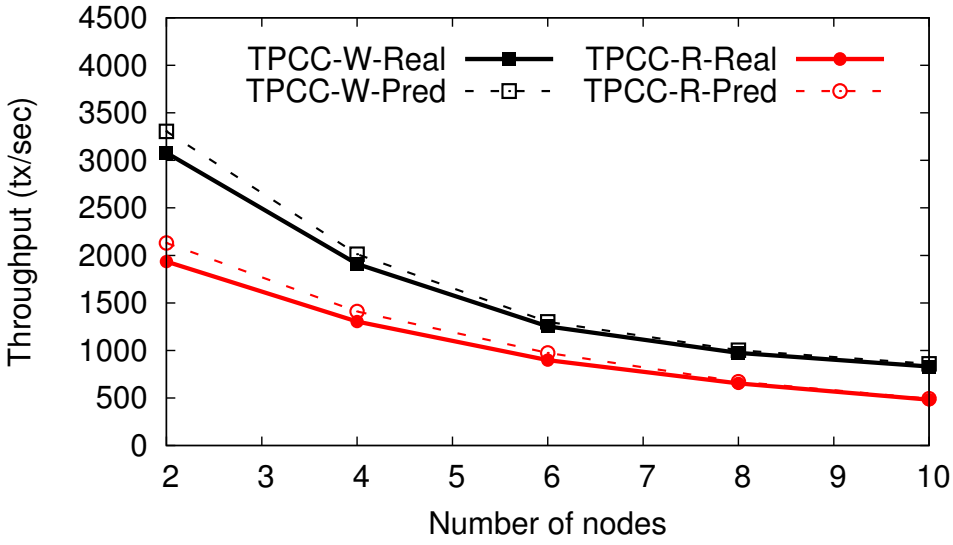
AM predicts KPIs and updates inputs for ML

Model's accuracy



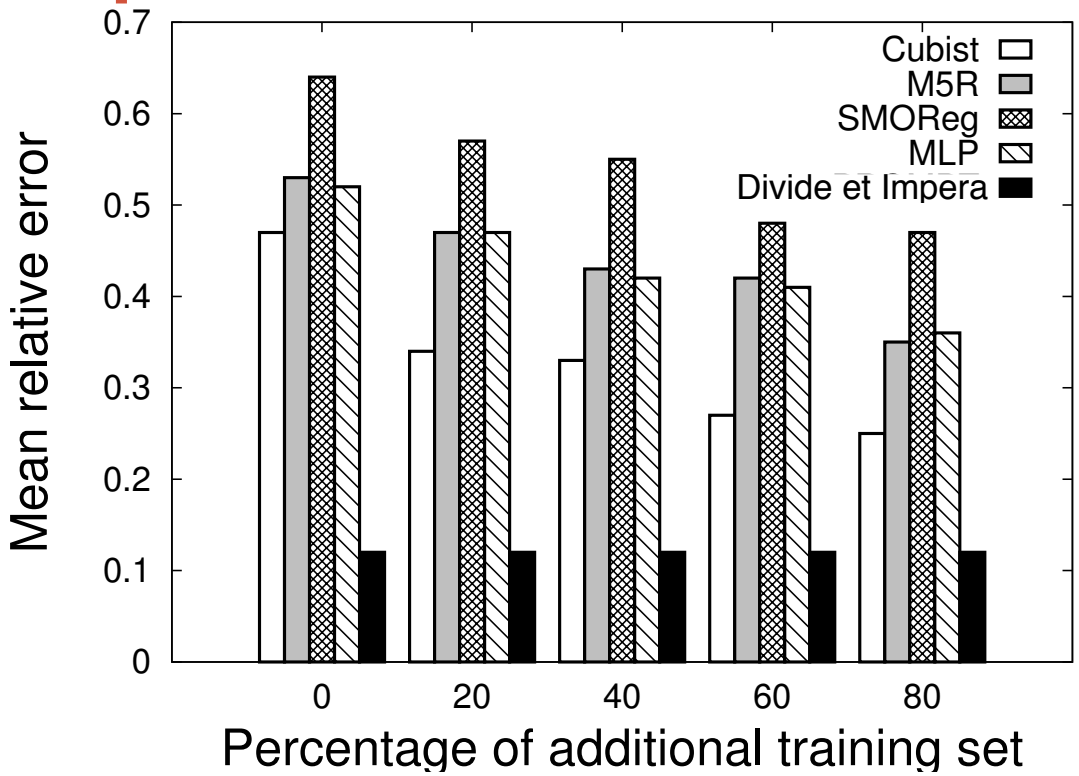






TOP: primary-backup. BOTTOM: multi-master (2PC-based)

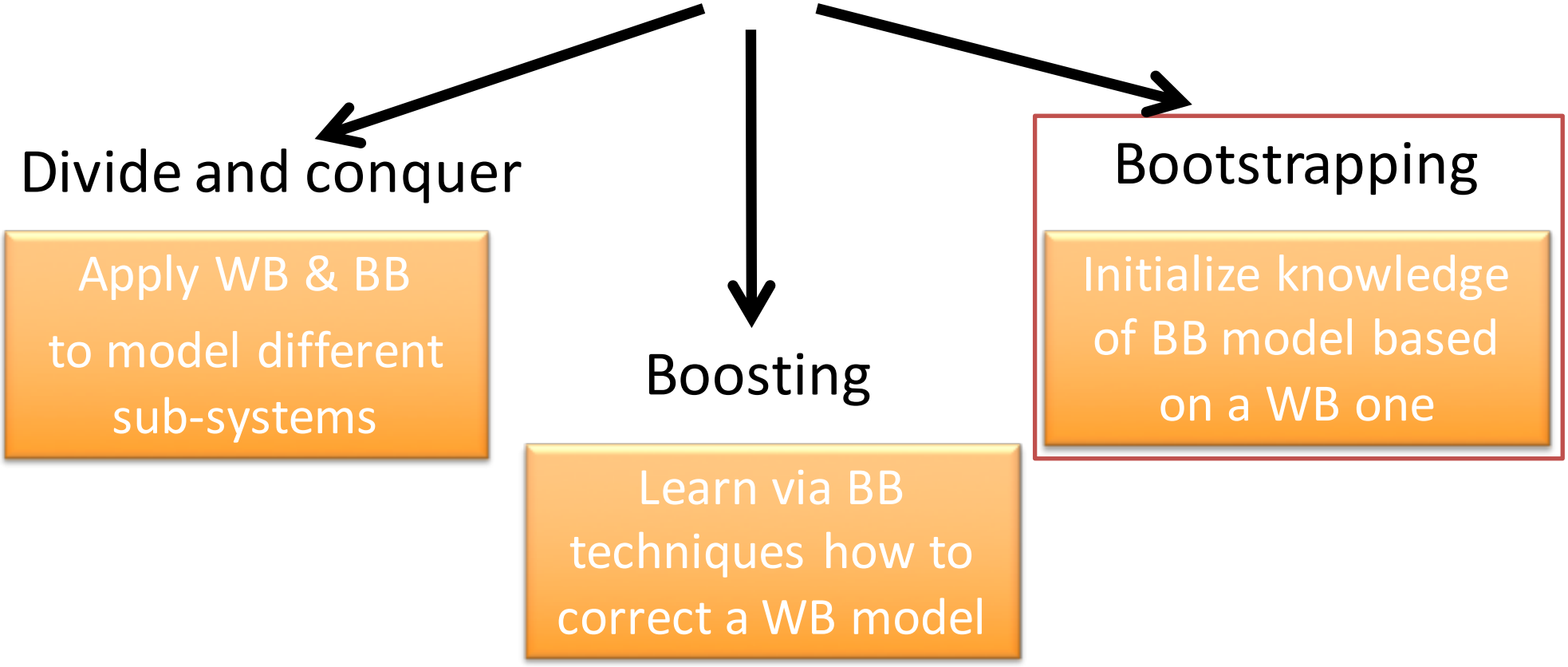
Comparison with Pure ML



- YCSB (transactified) workloads while varying
  - # operations/tx
  - Transactional mix
  - Platform Scale & replication degree

Gray box modeling

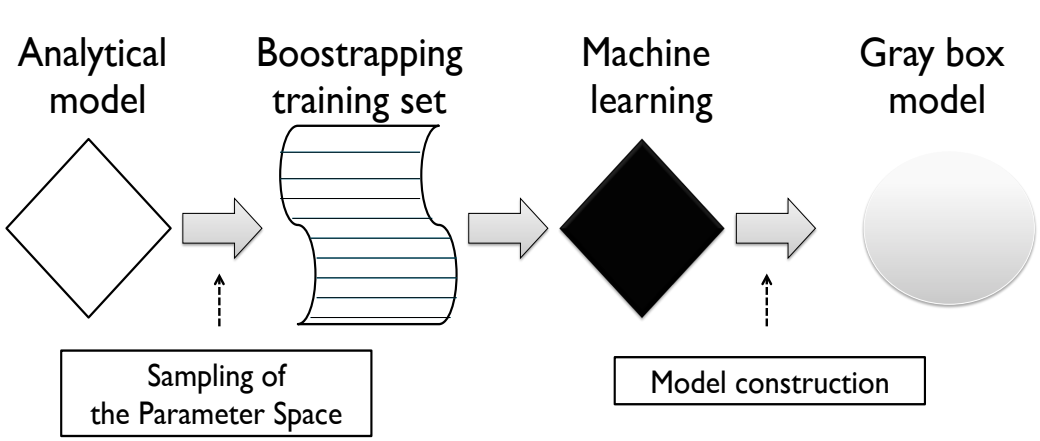
- I will present three methodologies:



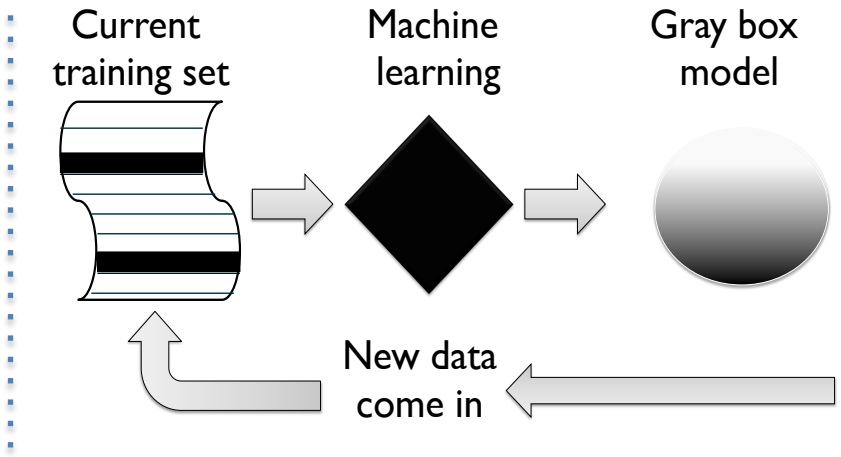
Bootstrapping [ICPADS15, SEAMS18]

Obtain zero-training-time ML via initial AM

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

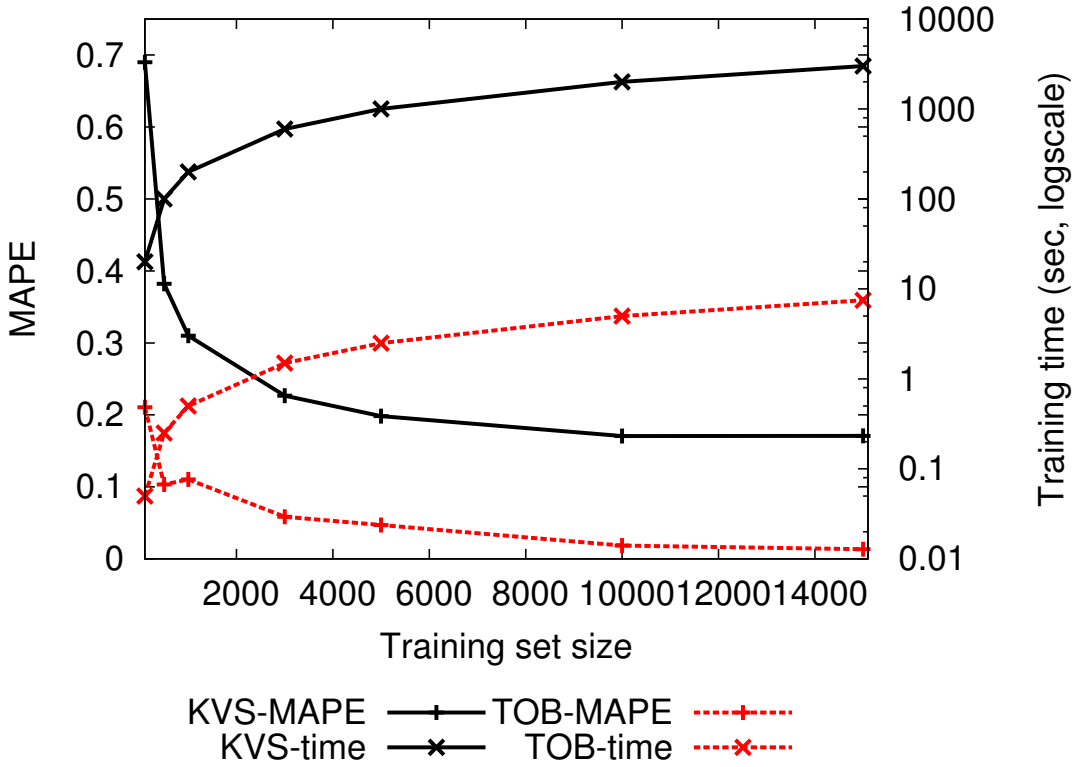


(1) Boostrapping phase



(2) Model update

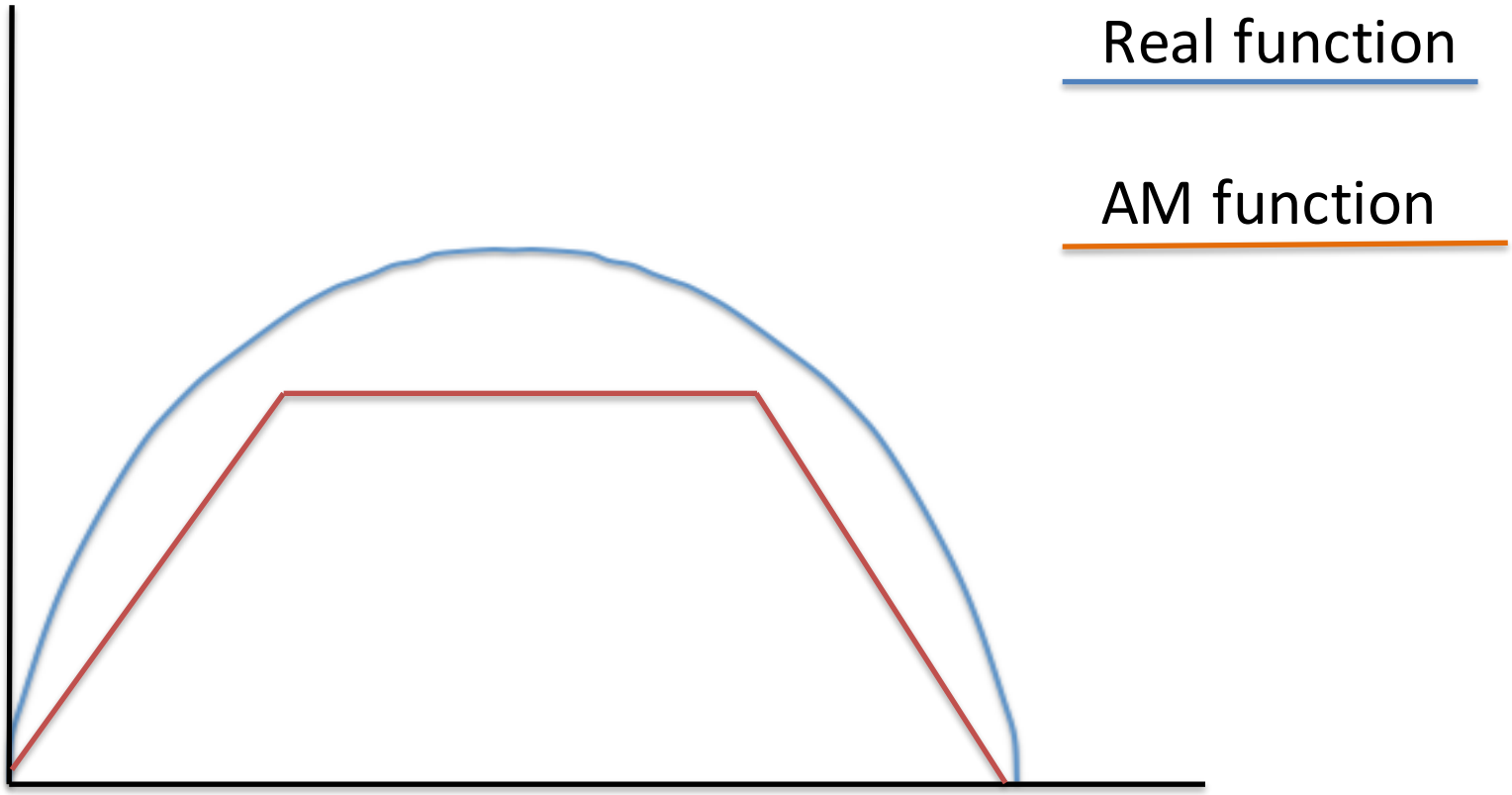
How many synthetic samples?



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

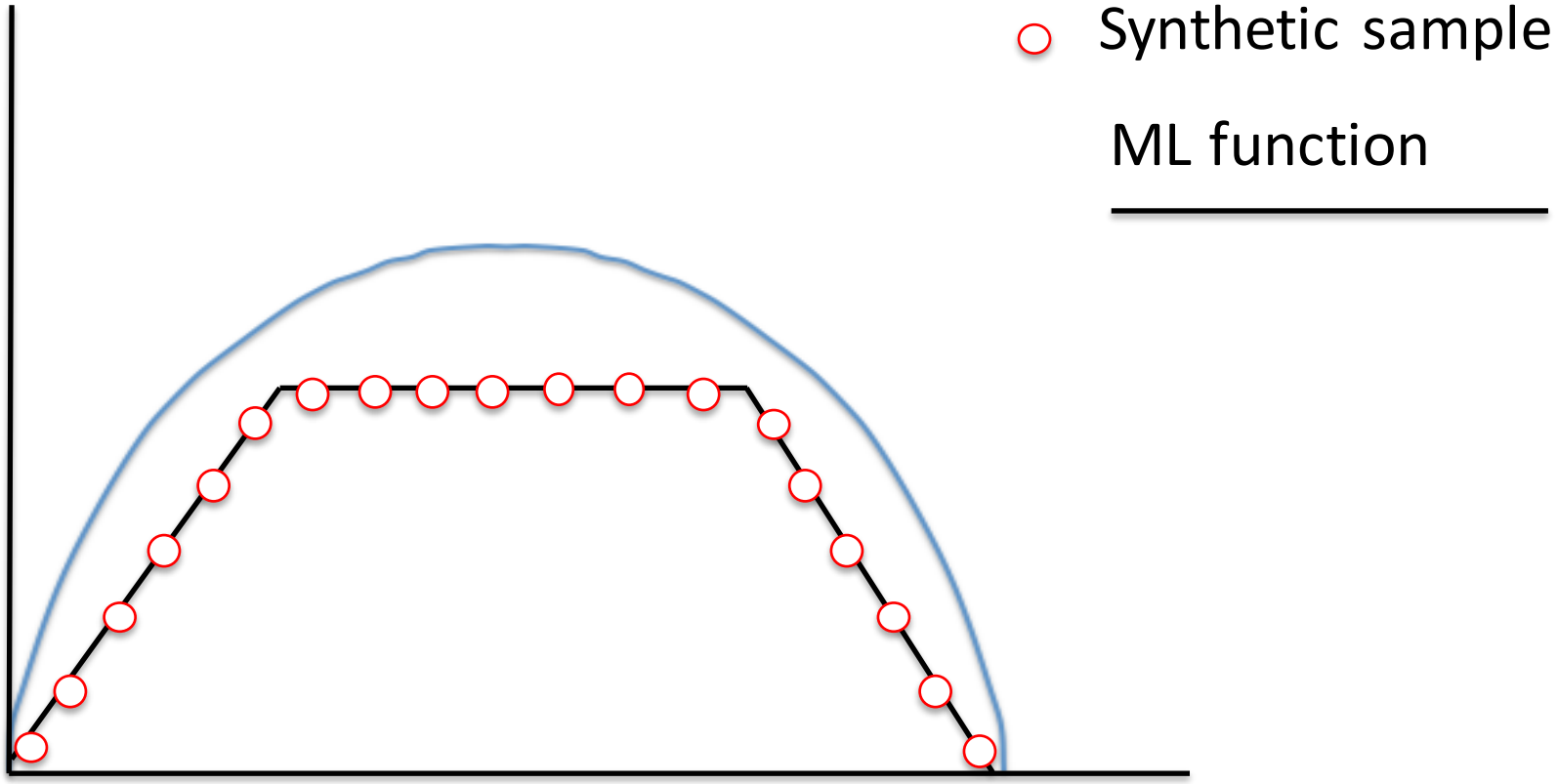
How to update the synthetic training set?

Real vs AM function



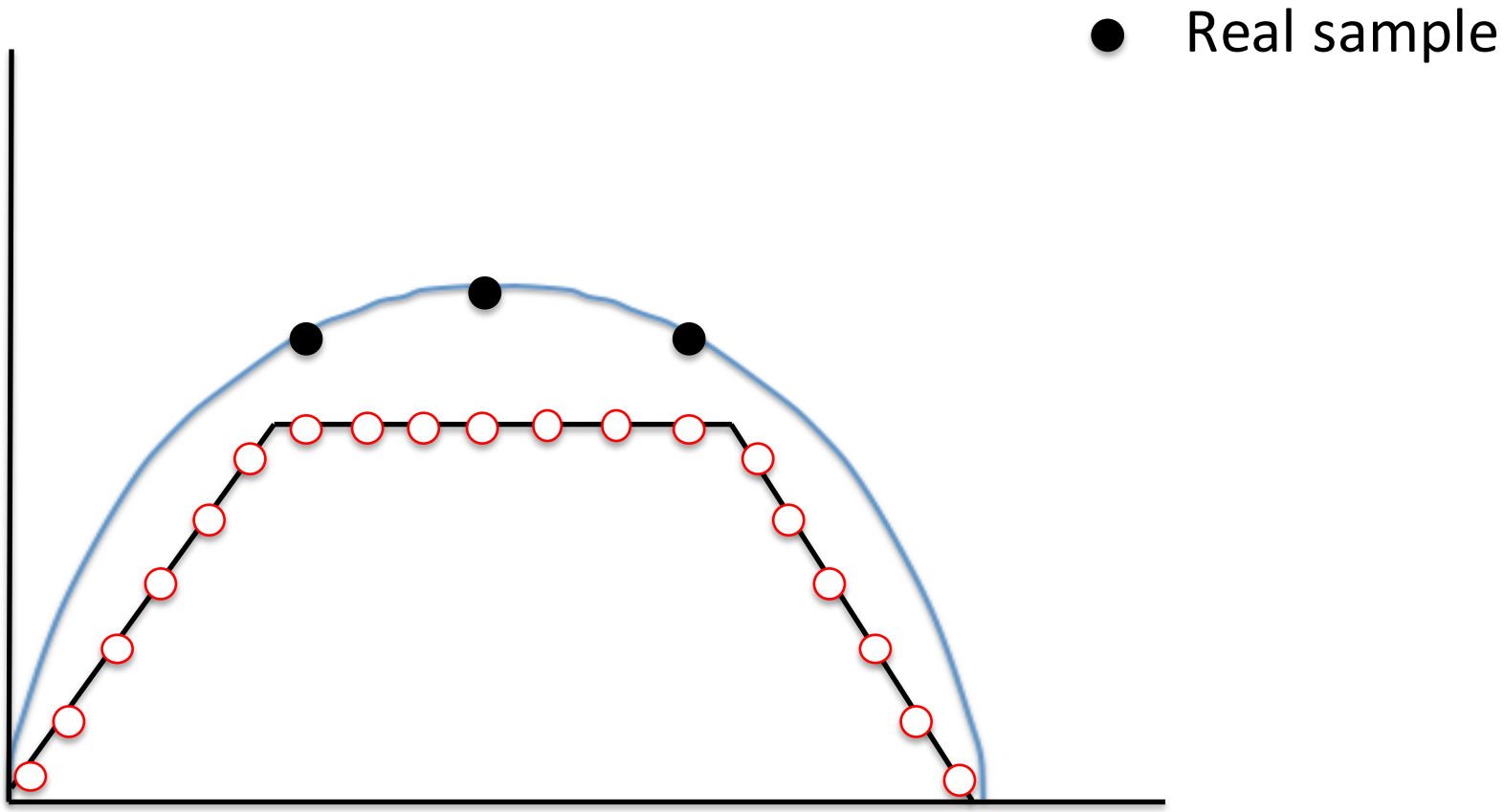
Real vs learnt

- Assuming enough point to perfectly learn AM



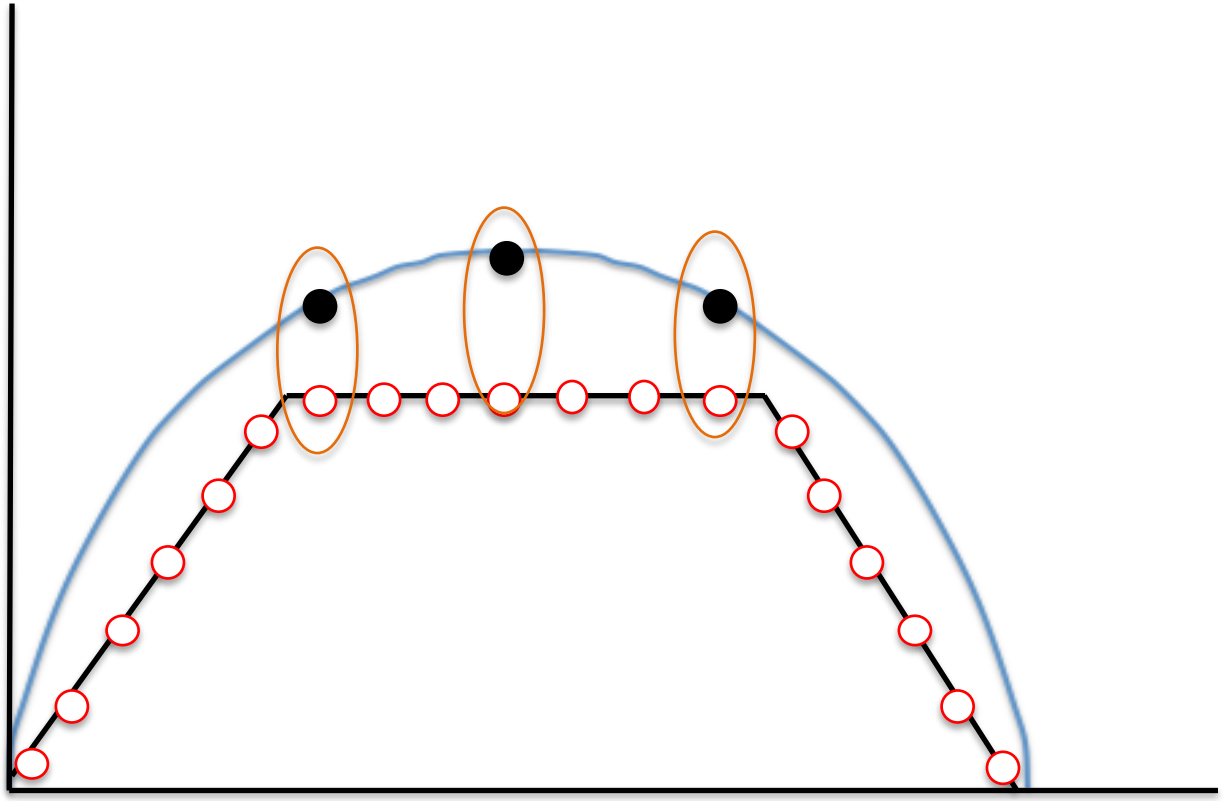
Merge

- Add real samples to synthetic



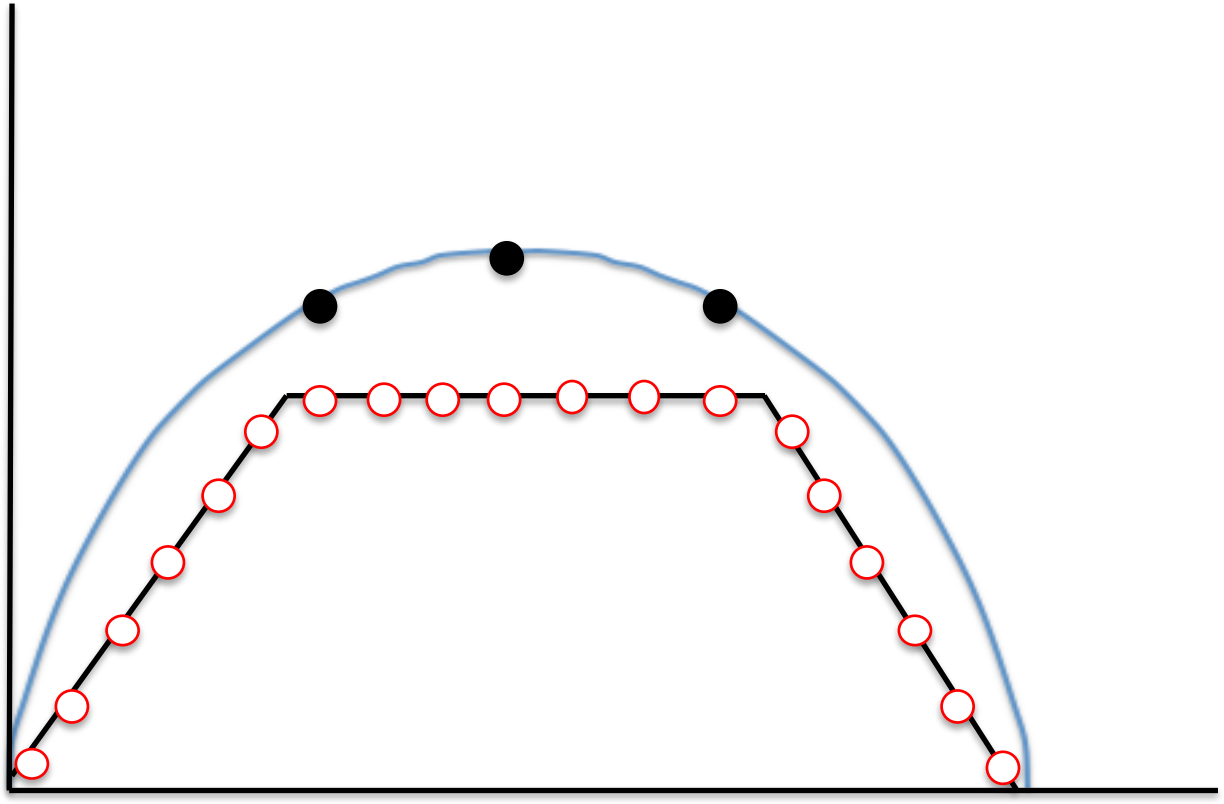
Merge

- Problem: same/near samples have diff. output



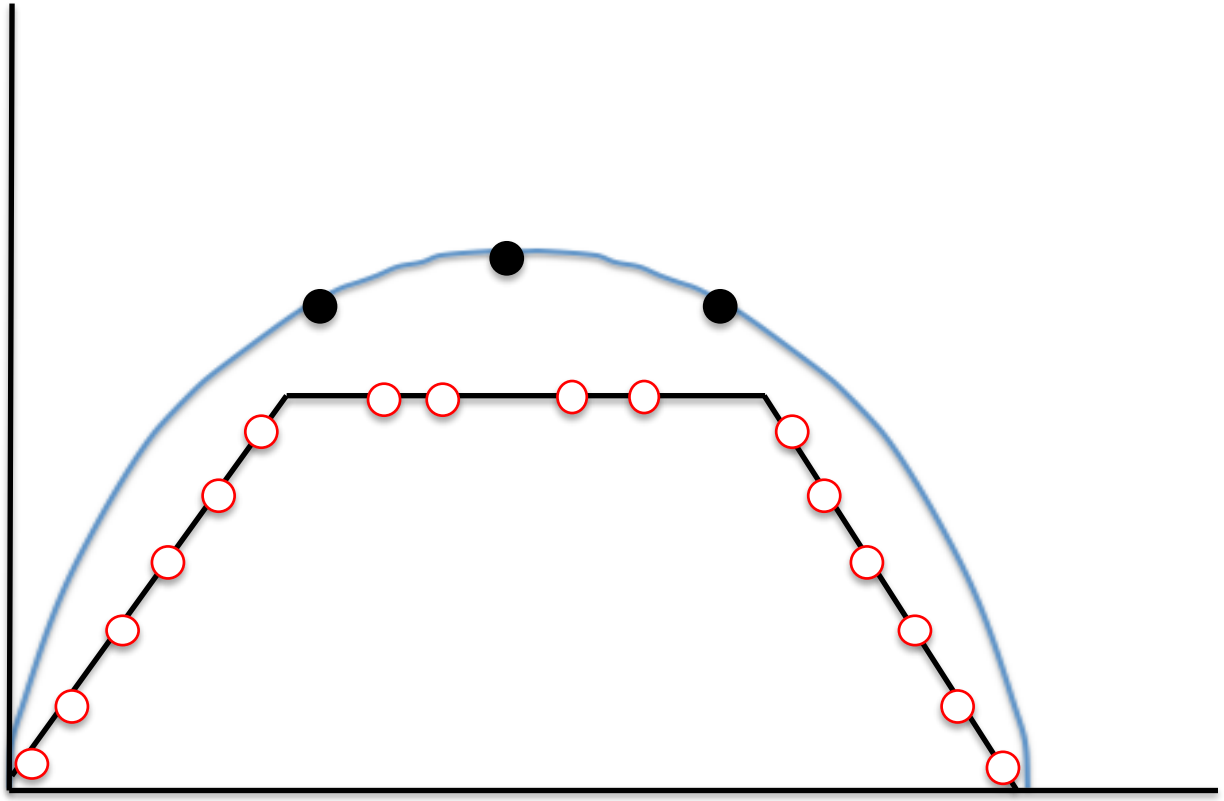
Replace Nearest Neighbor (RNN)

- Remove nearest neighbor



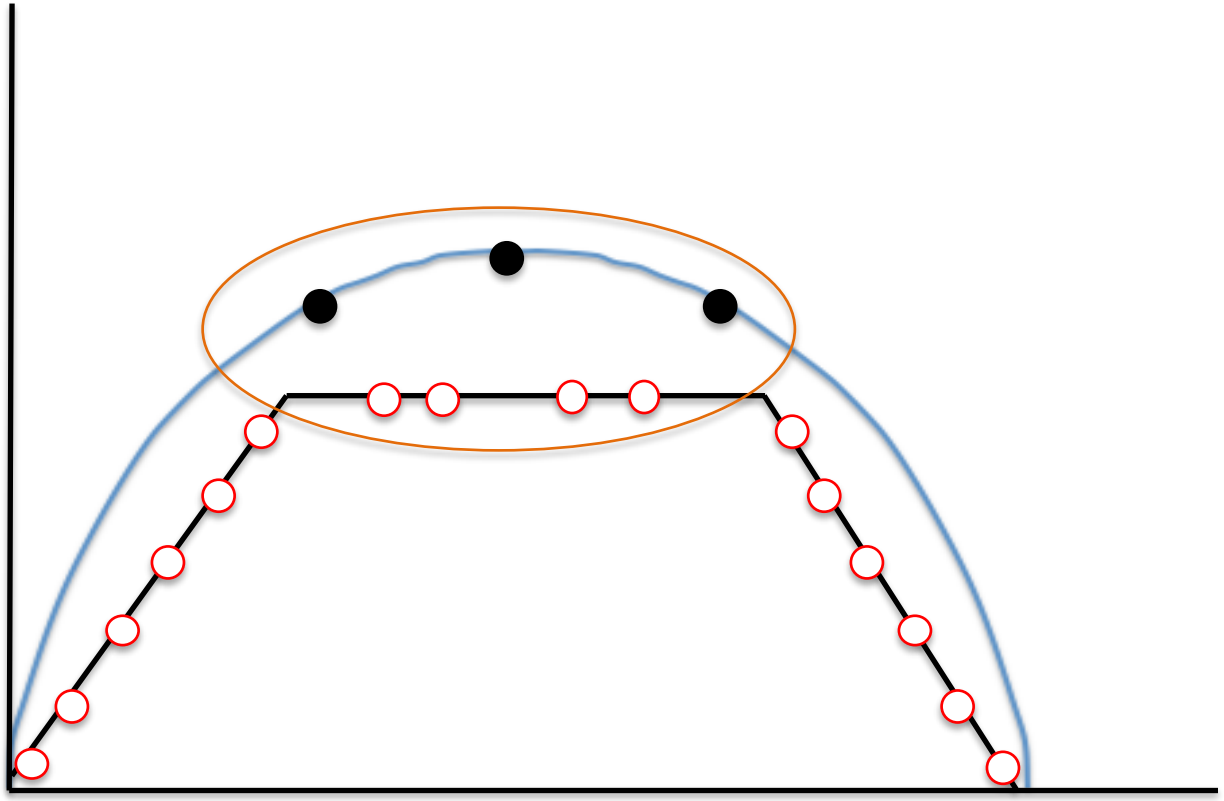
Replace Nearest Neighbor (RNN)

- Preserve distribution…



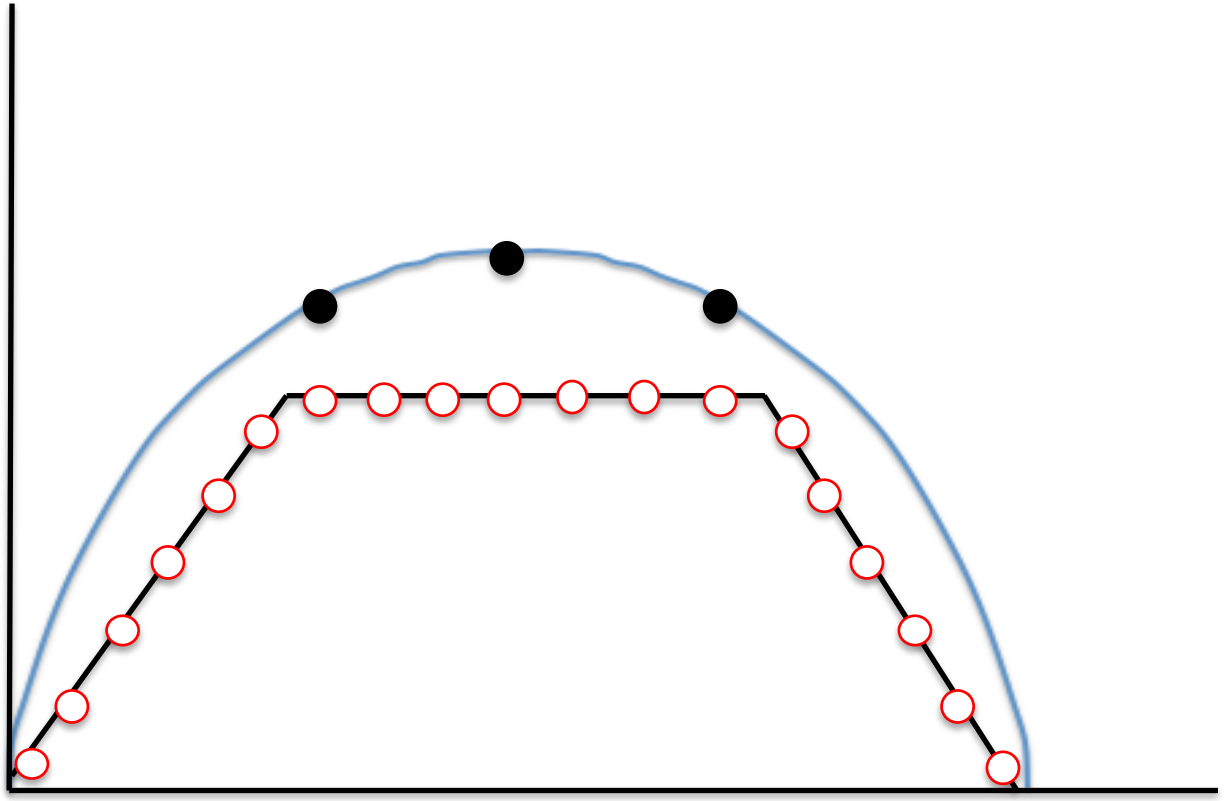
Replace Nearest Neighbor (RNN)

- … but may induce alternating outputs



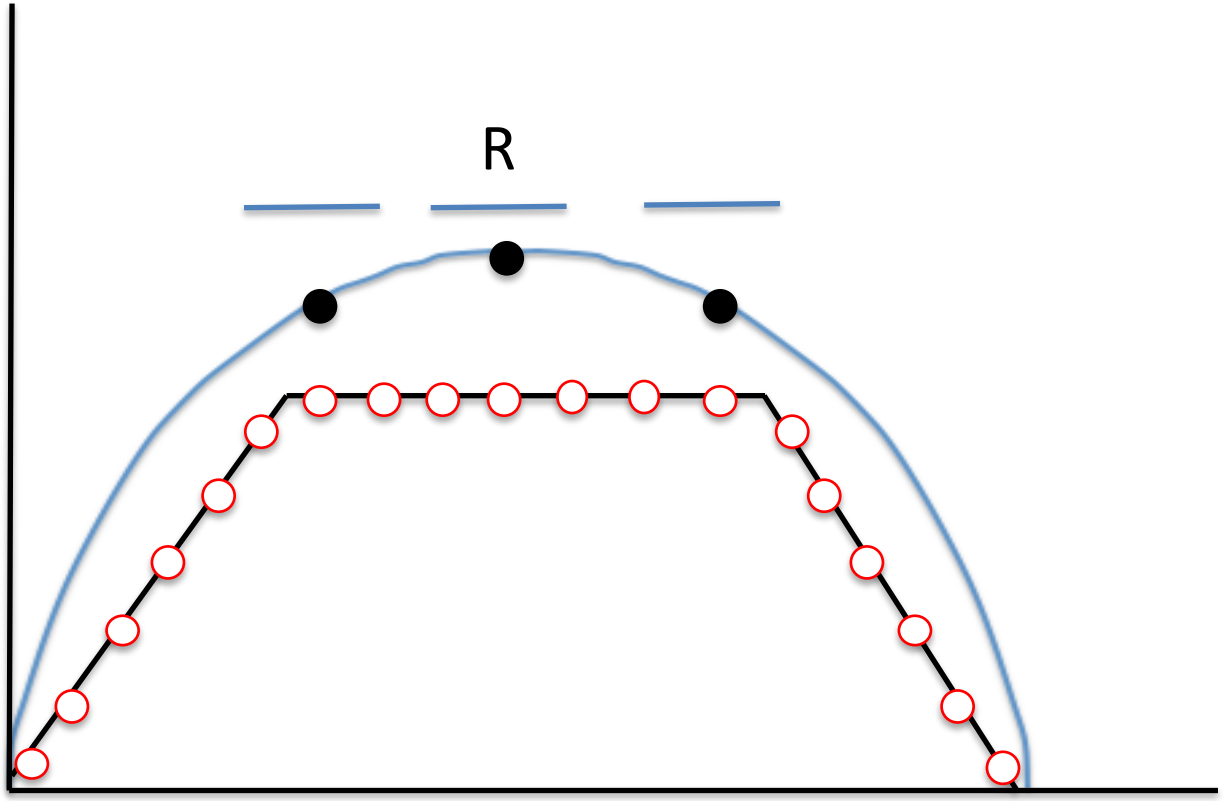
Replace Nearest Region (RNR)

- Add real and **remove** synth. samples in a radius



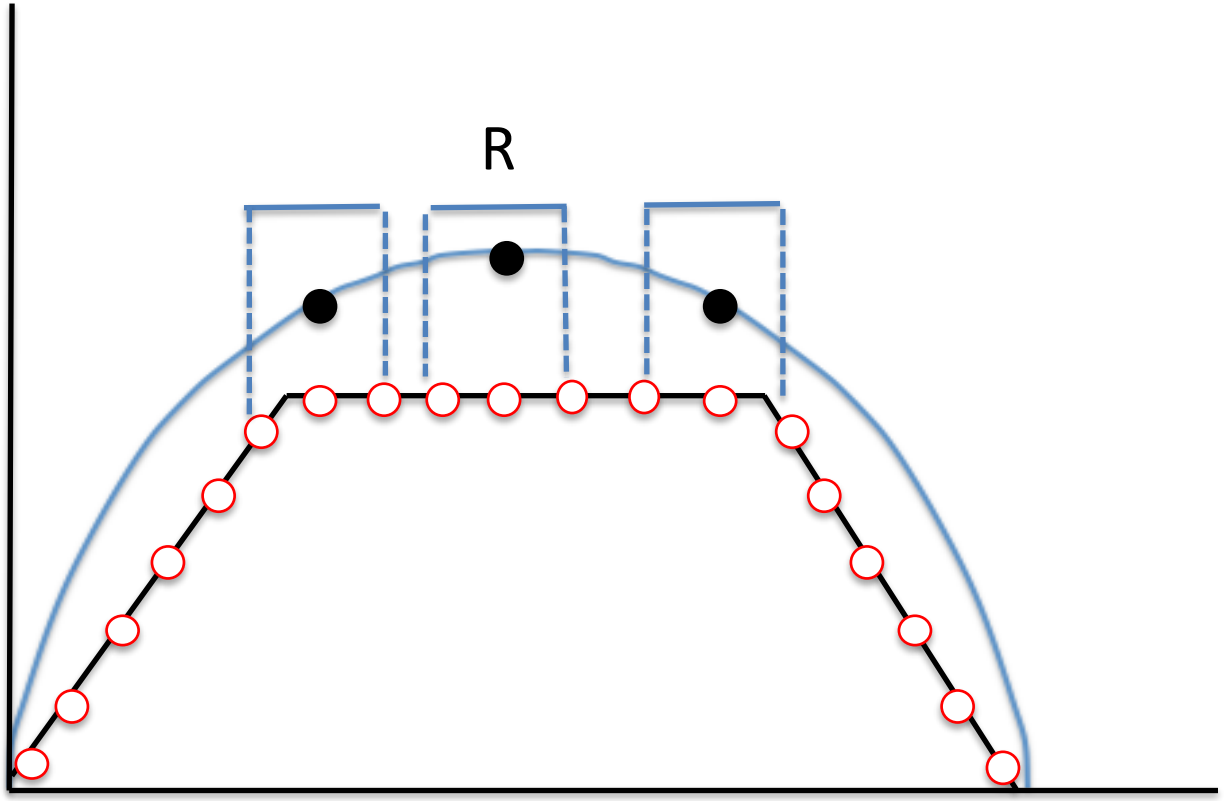
Replace Nearest Region (RNR)

- R = radius defining neighborhood



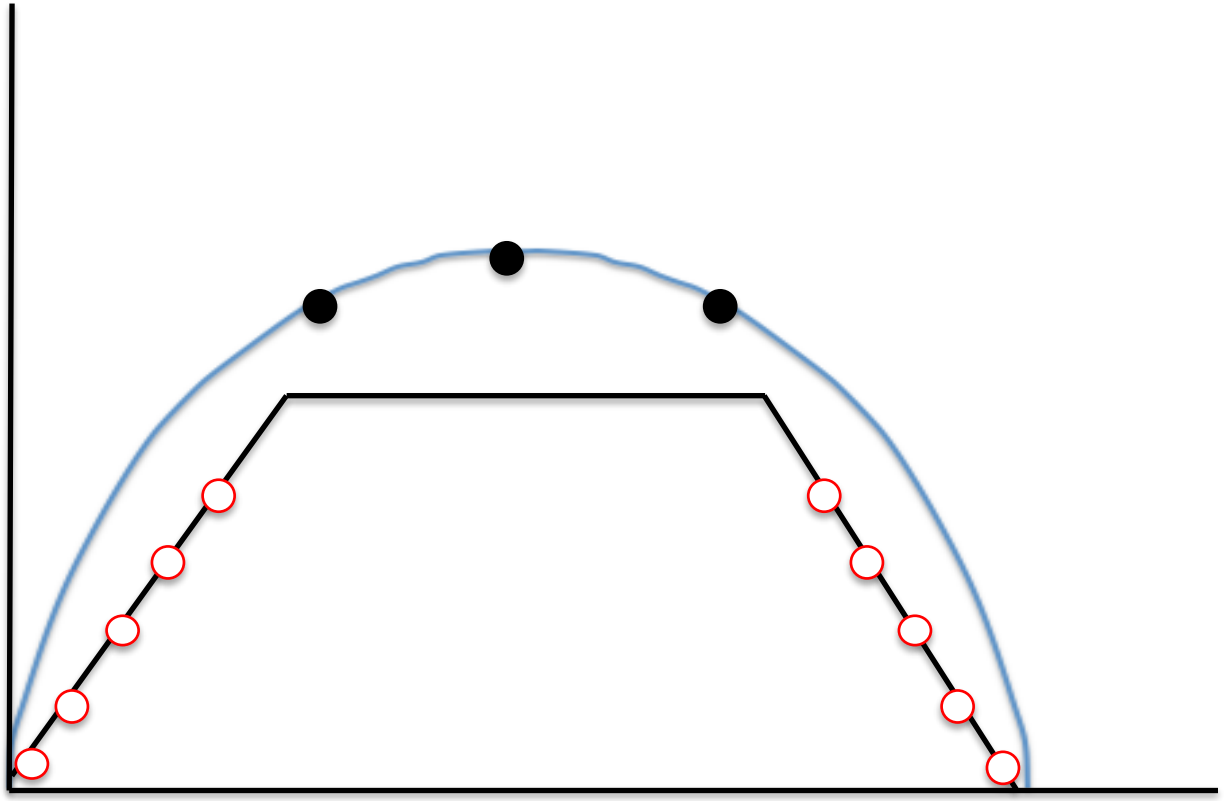
Replace Nearest Region (RNR)

- R = radius defining neighborhood



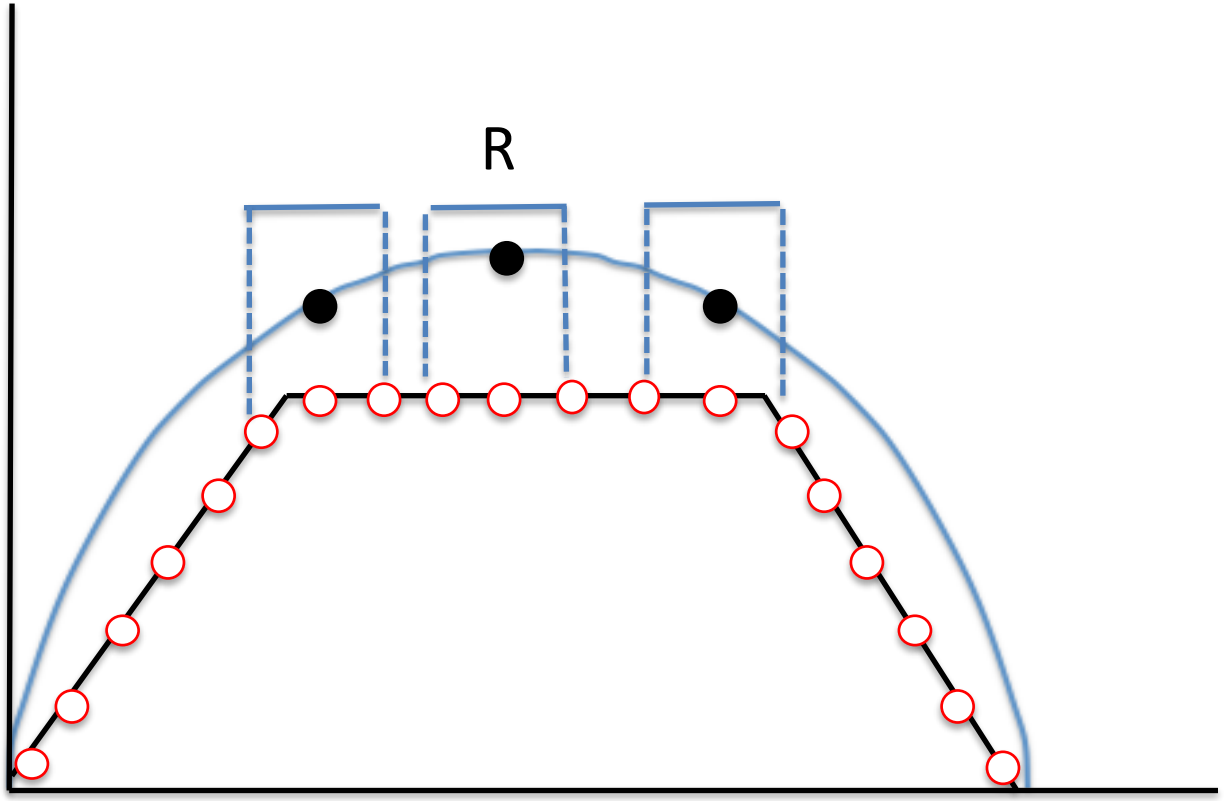
Replace Nearest Region (RNR)

- Skew samples' distribution



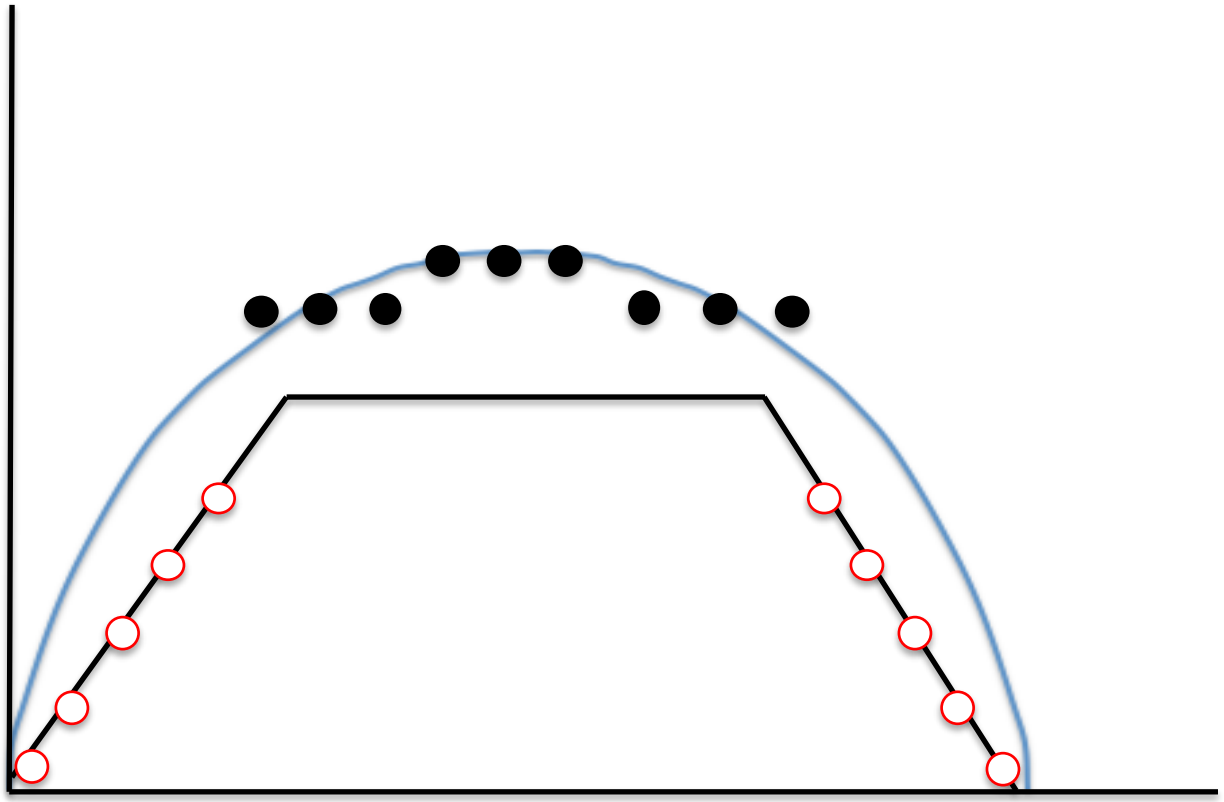
Replace Nearest Region 2 (RNR2)

- **Replace** all synthetic samples in a radius R



Replace Nearest Region 2 (RNR2)

- 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 with Merge-based updates

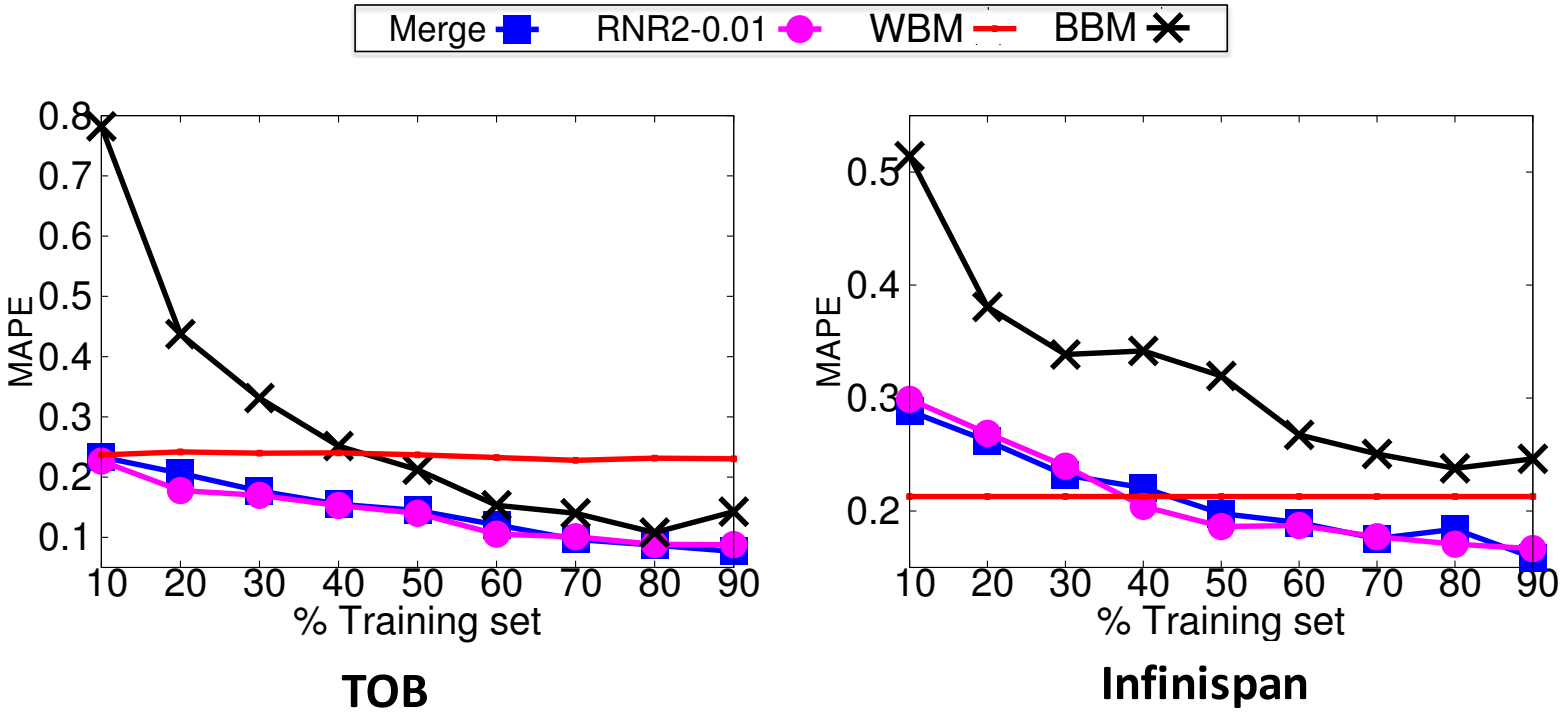
Too high can cause over-fitting!

- Learner fails to generalize

Evaluation

- Case studies
  - Response time in Total Order Broadcast (TOB)
    - building block at the basis of many data grids
    - 2-dimensional yet highly nonlinear perf. function
  - Throughput of Infinispan data grid
    - 7-dimensional performance function

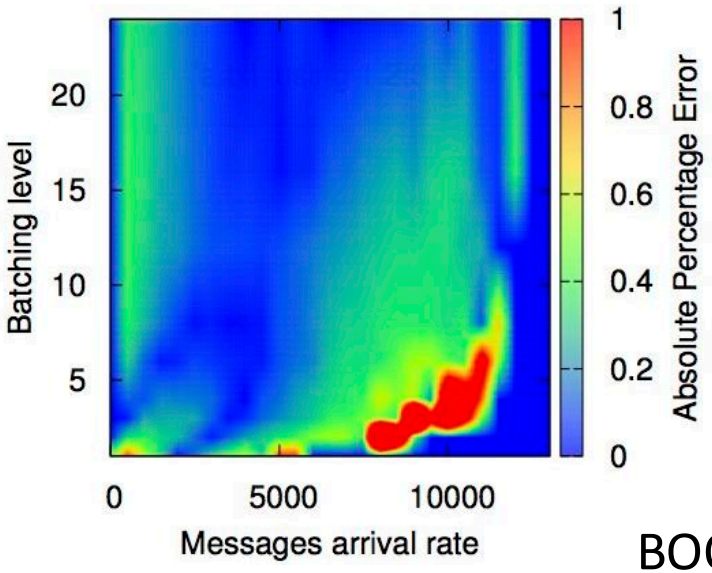
Accuracy



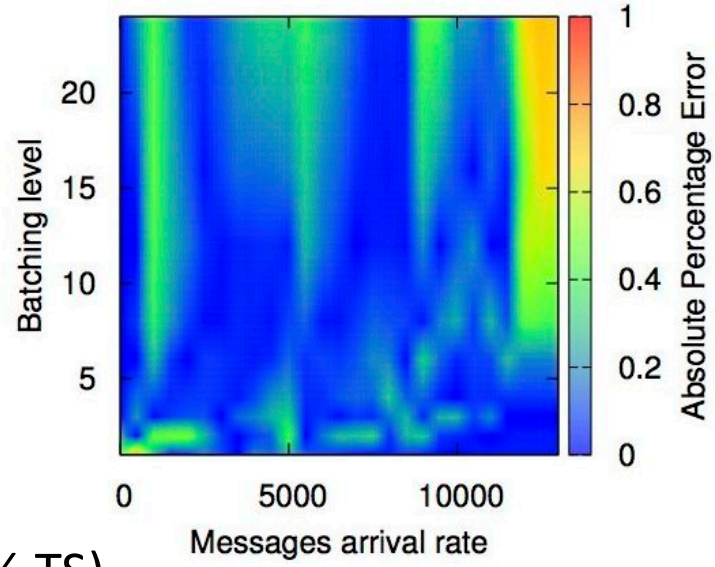
- Best accuracy than individual B&W-box models
  - AM prediction corrected as new data is acquired
  - Same accuracy of BB with far less training data (>5x)

Visualizing the correction

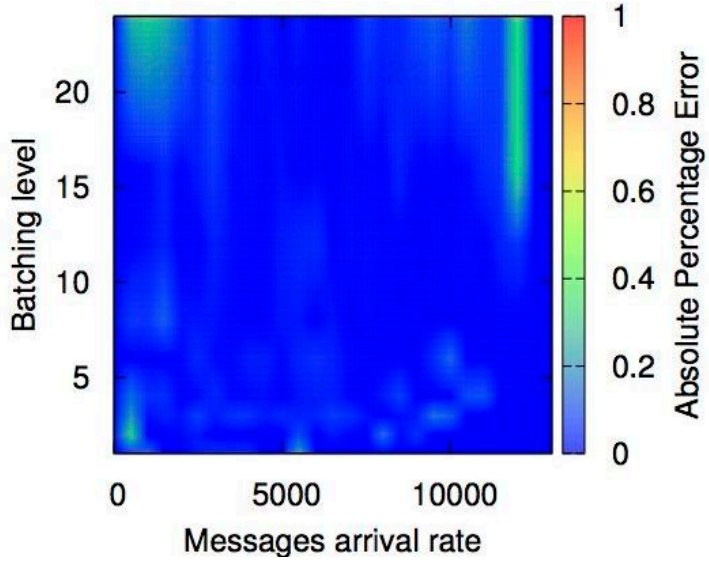
BASE AM



PURE ML (70% TS)

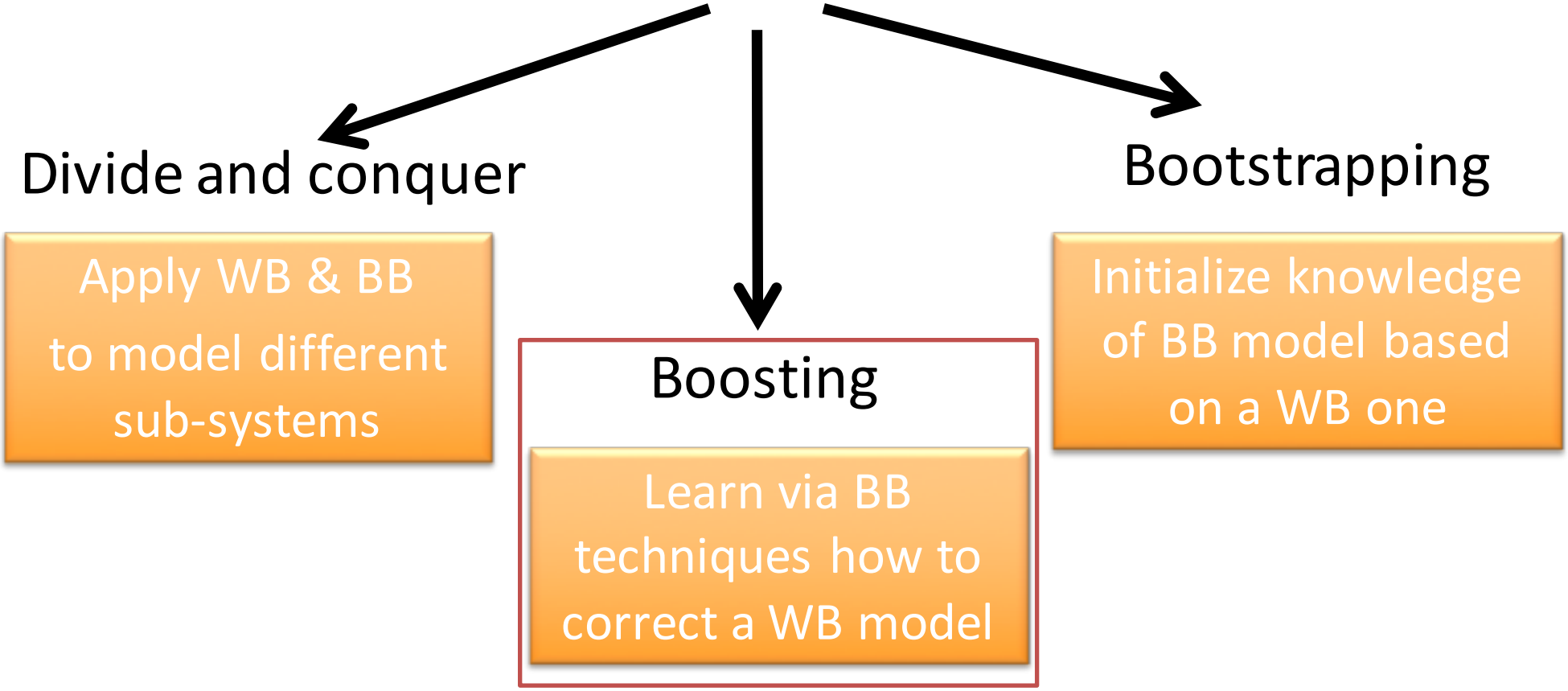


BOOTSTRAPPED ML (70% TS)



Gray box modeling

- I will present three methodologies:



Boosting [ICPE15, Netys13]

Learning the error of a model on a function may be simpler than learning the function itself

- 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}

Training and Querying

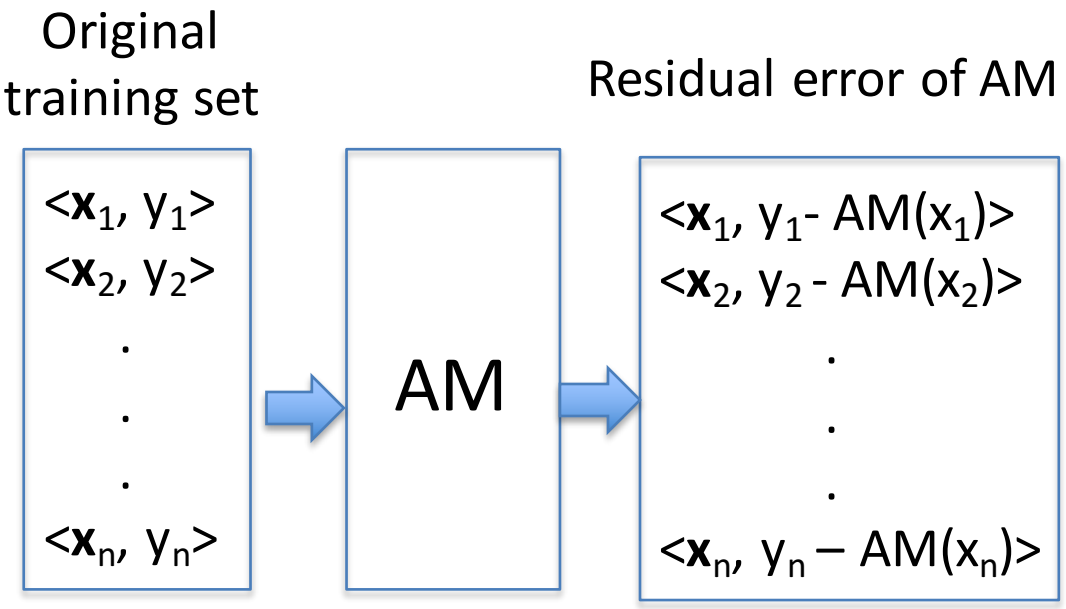
Training

Original training set

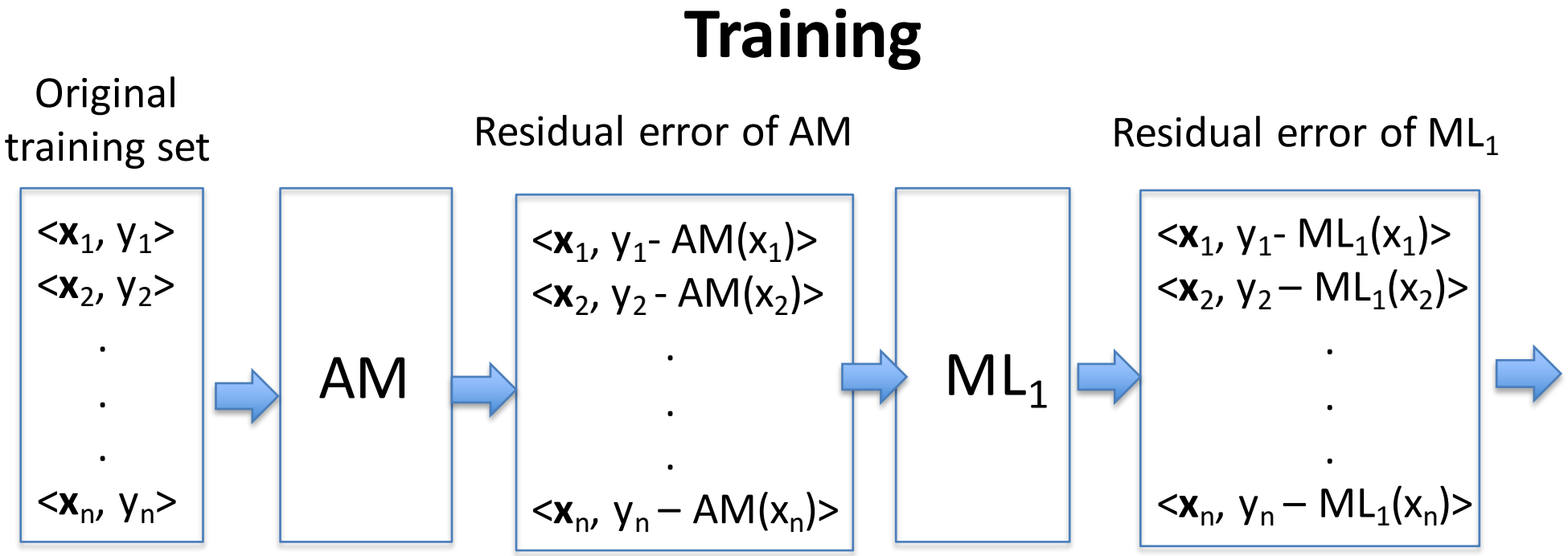
$\langle \mathbf{x}_1, y_1 \rangle$
$\langle \mathbf{x}_2, y_2 \rangle$
.
.
.
$\langle \mathbf{x}_n, y_n \rangle$

Training and Querying

Training

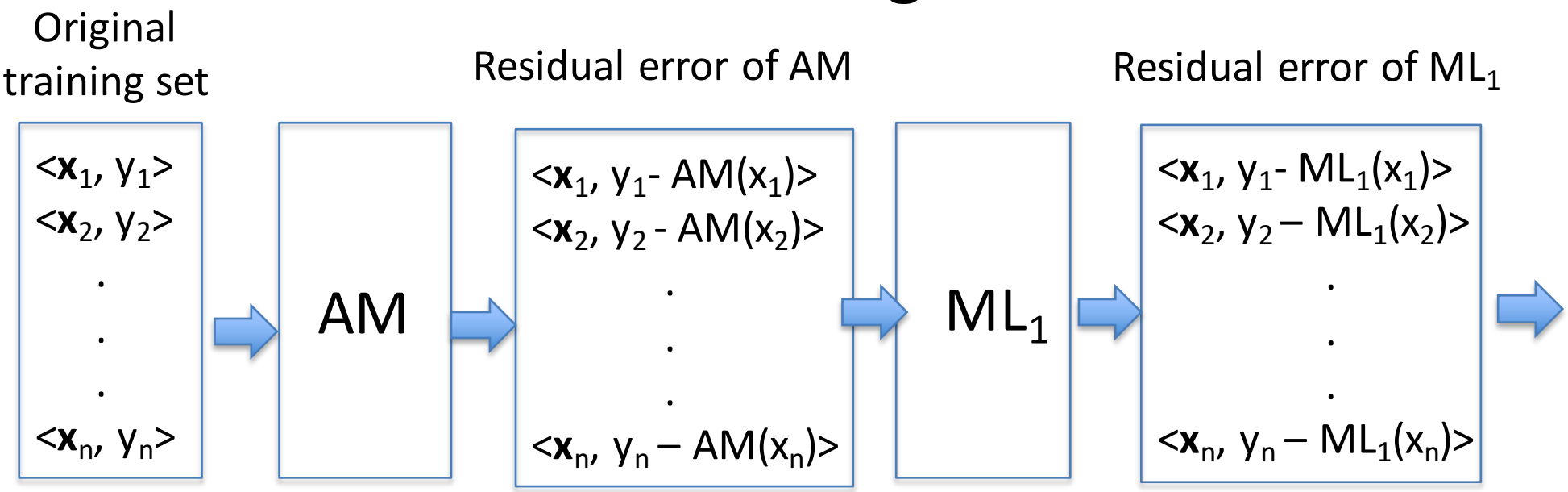


Training and Querying



Training and Querying

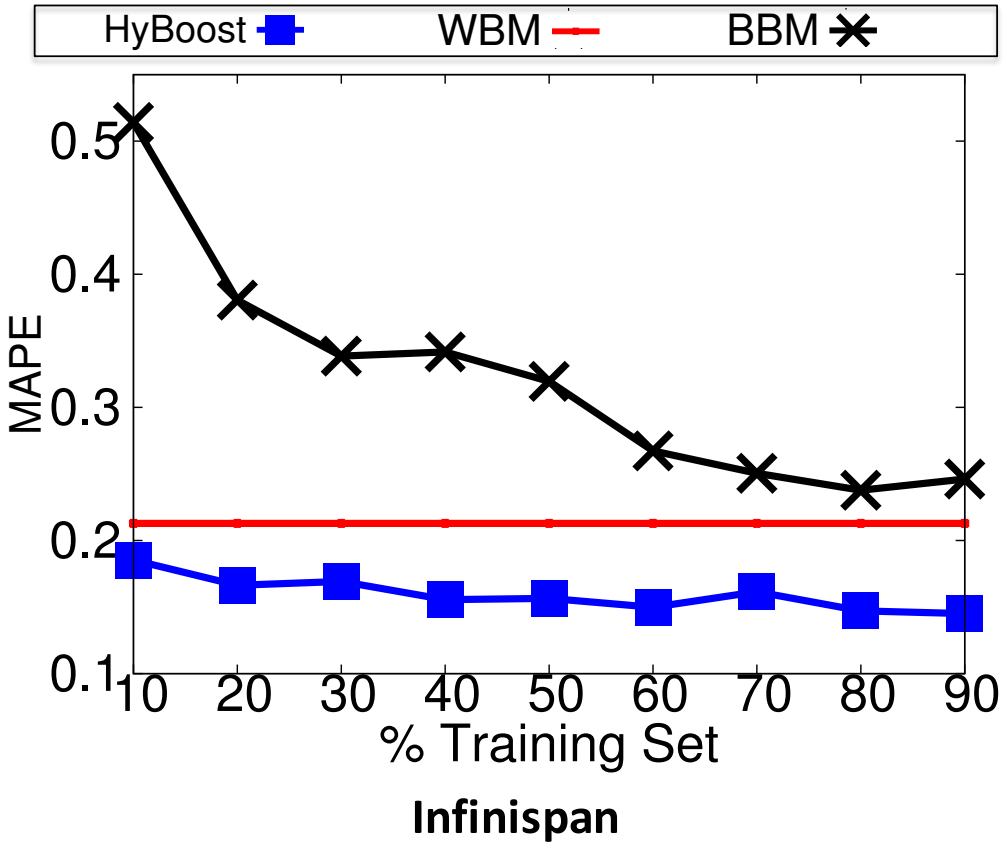
Training



Query

$$F(\mathbf{x}) = AM(\mathbf{x}) + ML_1(\mathbf{x}) + \ldots + ML_m(\mathbf{x})$$

Evaluation



- Chain composed by AM + Decision Tree

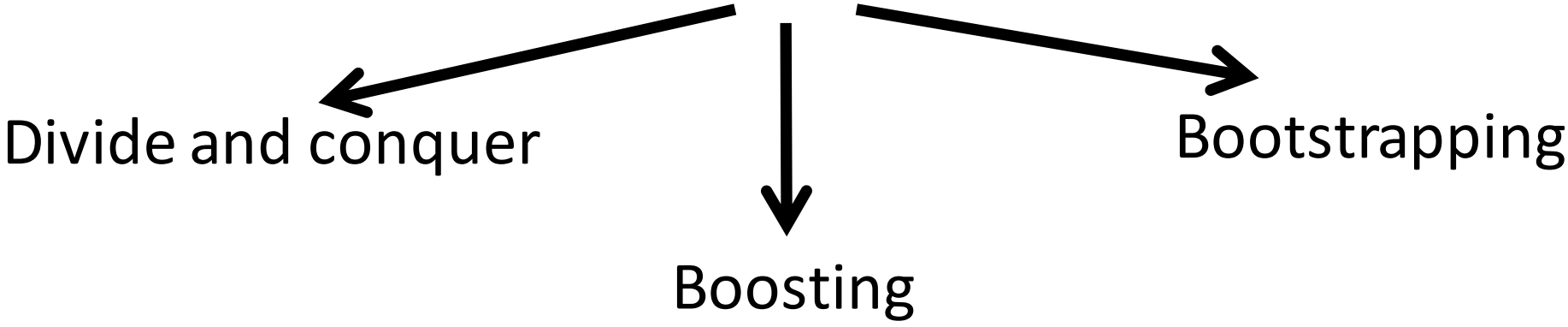
Concluding remarks

Time to reconcile black-box & white-box modeling

- White and black box modelling are not antithetic techniques!
- They can be effectively used in synergy
  - Increased predictive power via analytical models
  - Incremental learning via black box models
- Presented three gray box methodologies:
  - Divide and conquer, Bootstrapping, Boosting
  - Use case: transactional data-grids in the cloud

Open questions (1/4)

- The 3 presented methodologies are only some possible approaches

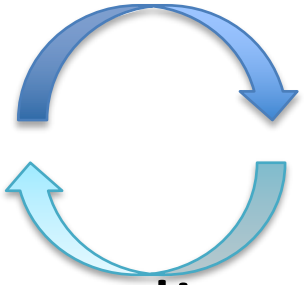


- Design space is largely unexplored
- Any other way of using white-box models and machine learning in synergy?

Open questions (2/4)

- Convergence of model coupling in *divide and conquer* schemes
- Fixed point recursion vs iterative schemes
- Sufficient/necessary conditions for convergence?

AM predicts input parameters for ML



ML predicts input parameters for AM

Open questions (3/4)

- Which gray box modelling methodology to choose?

- Can we infer the best gray box technique by analyzing the error distribution of the AM model?

Open questions (4/4)

- White box models are normally understandable by humans

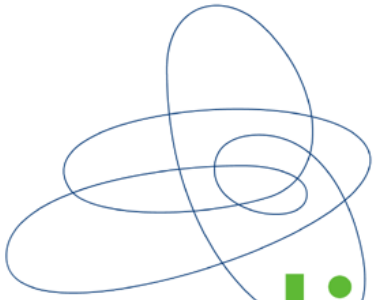
- Not necessarily true for gray-box models, e.g.:
  - bootstrapping a Neural Network with an AM
  - boosting an AM with a decision tree

- Can we distill a “corrected” white-box model that preserves human-readability?

THANK YOU

Questions?

TÉCNICO
LISBOA



inesc id
lisboa

References

- [Netys13] Diego Didona, Pascal Felber, Derin Harmanci, Paolo Romano and Joerg Schenker, Identifying the Optimal Level of Parallelism in Transactional Memory Systems, The International Conference on Networked Systems 2013, **BEST PAPER AWARD**
- [DSN13] M. Couceiro, P. Ruivo, Paolo Romano, L. Rodrigues, Chasing the Optimum in Replicated In-memory Transactional Platforms via Protocol Adaptation, The 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2013)
- [ICAC 13] Joao Paiva, Pedro Ruivo, Paolo Romano and Luis Rodrigues, AutoPlacer: scalable self-tuning data placement in distributed key-value stores, The 10th International Conference on Autonomic Computing (ICAC 2013), San Jose, CA, USA, 26-28 June 2013 - BEST PAPER AWARD FINALIST
- [TAAS14] D. Didona, Paolo Romano, S. Peluso, F. Quaglia, Transactional Auto Scaler: Elastic Scaling of In-Memory Transactional Data Grids, ACM Transactions on Autonomous and Adaptive Systems (TAAS), 9, 2, 2014
- [ICPE15] D. Didona, Paolo Romano, F. Quaglia, E. Torre, Combining Analytical Modeling and Machine-Learning to Enhance Robustness of Performance Prediction Models, 6th ACM/SPEC International Conference on Performance Engineering (ICPE), Feb 2015
- [SEAMS18] F. Duarte, R. Gil, Paolo Romano, A. Lopes and L. Rodrigues. Learning Non-Deterministic Impact Models for Adaptation. In Proceedings of the 13th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), Gothenburg, Sweden, May 2018.