Hybrid Machine Learning/Analytical Models for Performance Prediction

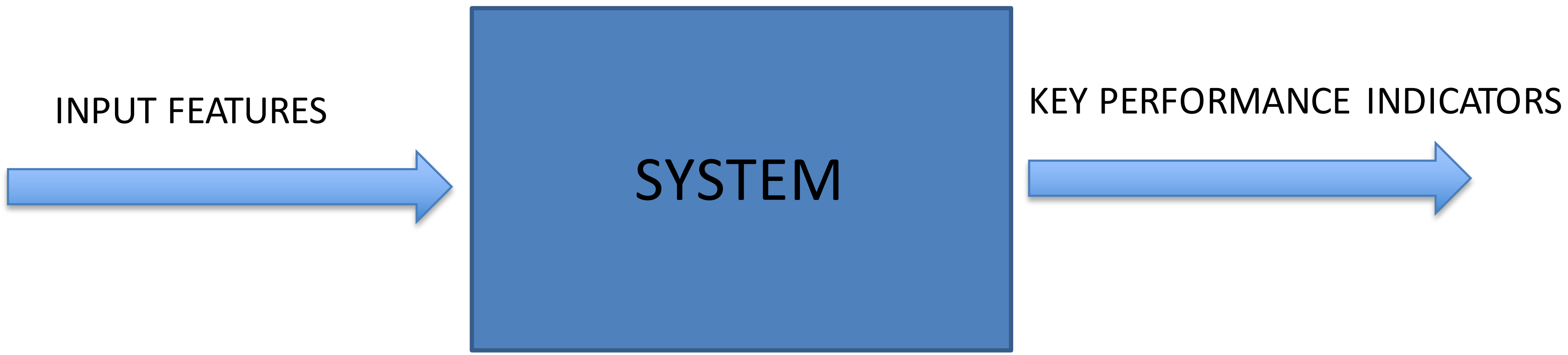
Diego Didona and Paolo Romano
INESC-ID / Instituto Superior Técnico

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

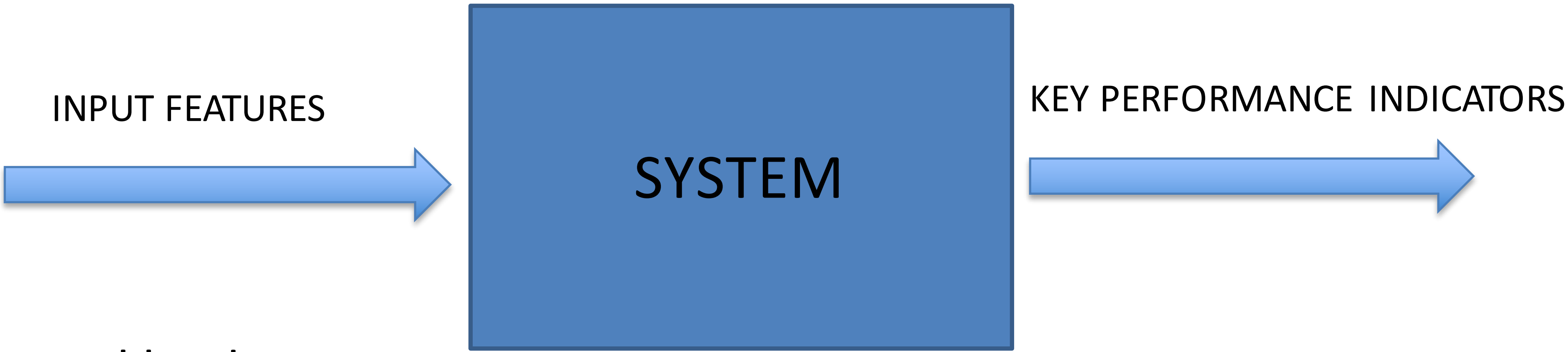
Outline

- 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



Modeling a system



Workload:
- Intensity, small vs large jobs

Infrastructure
- # servers, type of servers

Application-specific
- Replication

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)

$$\sum_{i=1}^{N} \frac{|real_i - pred_i|}{N\ real_i}$$

  - RMSE (Root Mean Square Error)

$$\sqrt{\sum_{i=1}^{N} \frac{(real_i - pred_i)^2}{N}}$$

White/Black Box Modeling 101

White box performance modeling

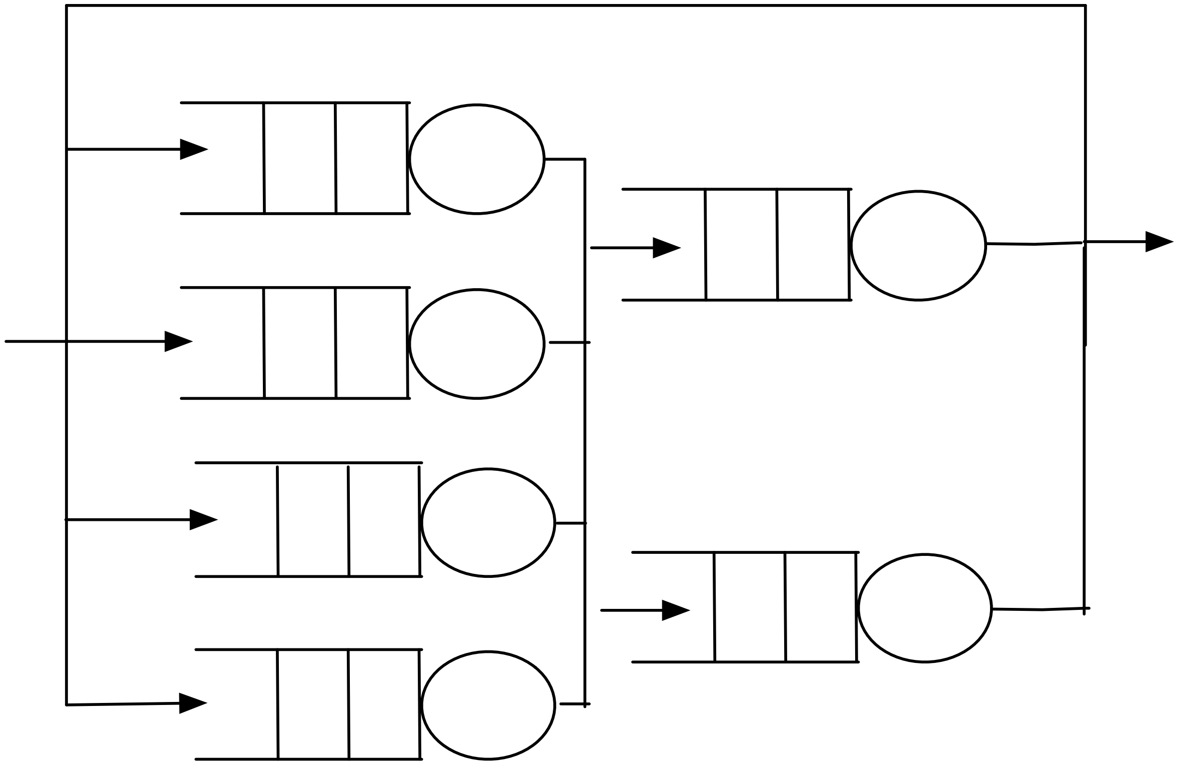
Leverage on knowledge about target app's internals

- 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

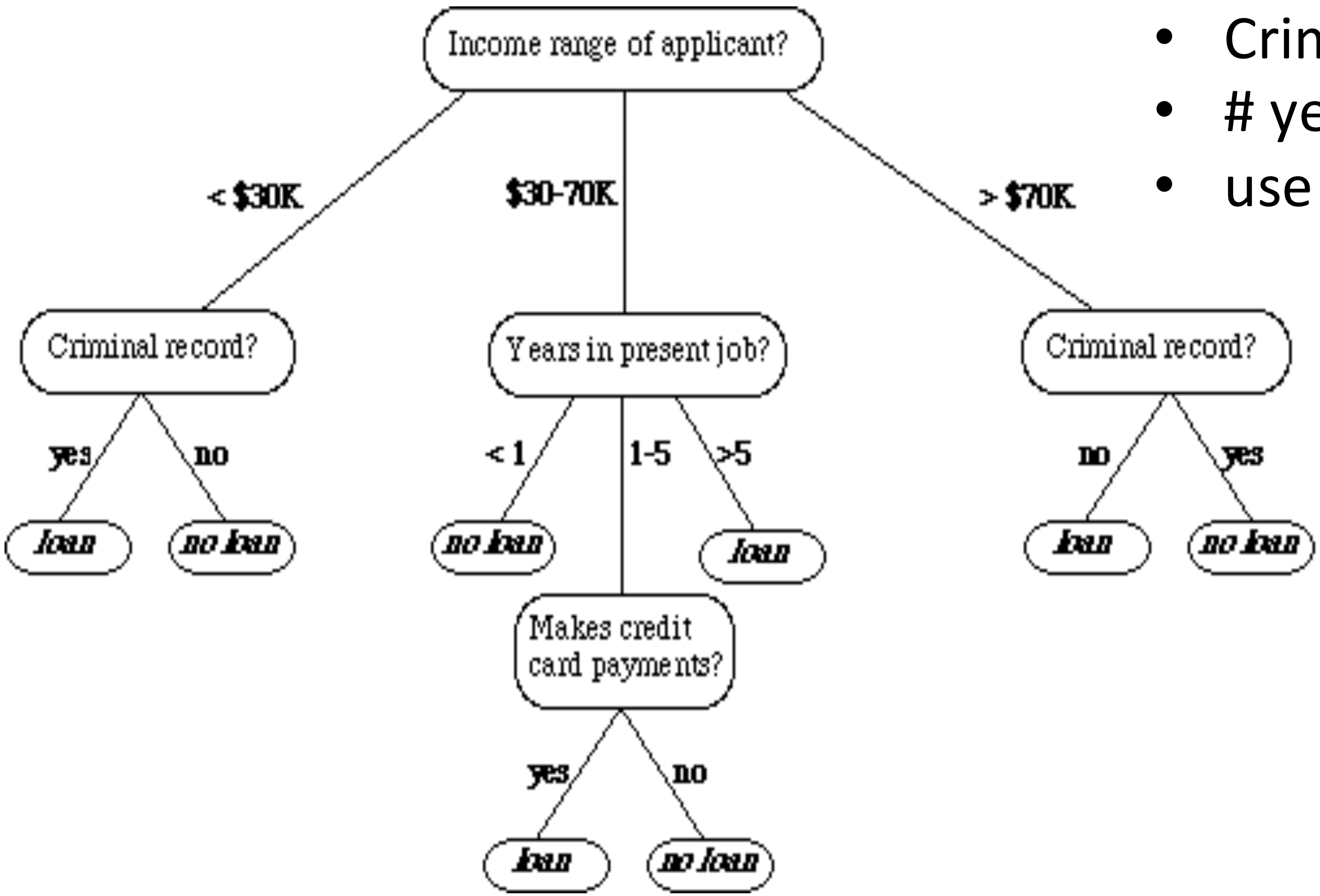
OFF-LINE SUPERVISED LEARNING

- 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

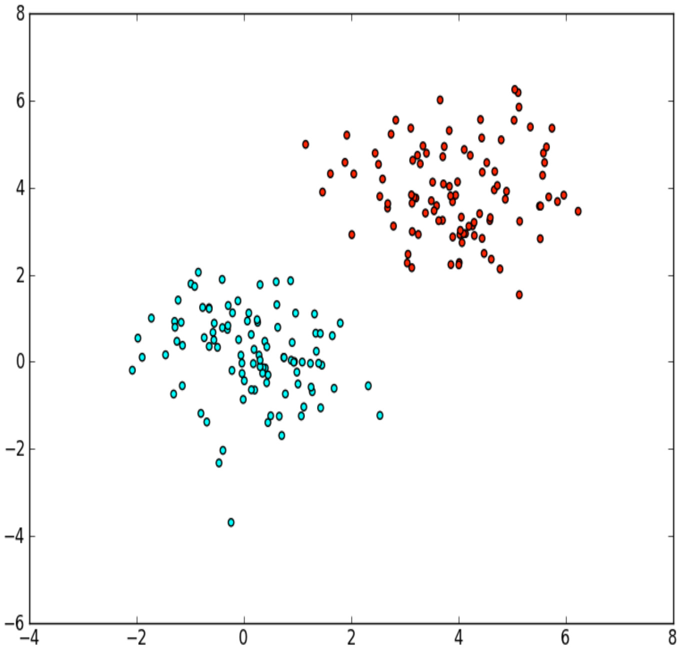


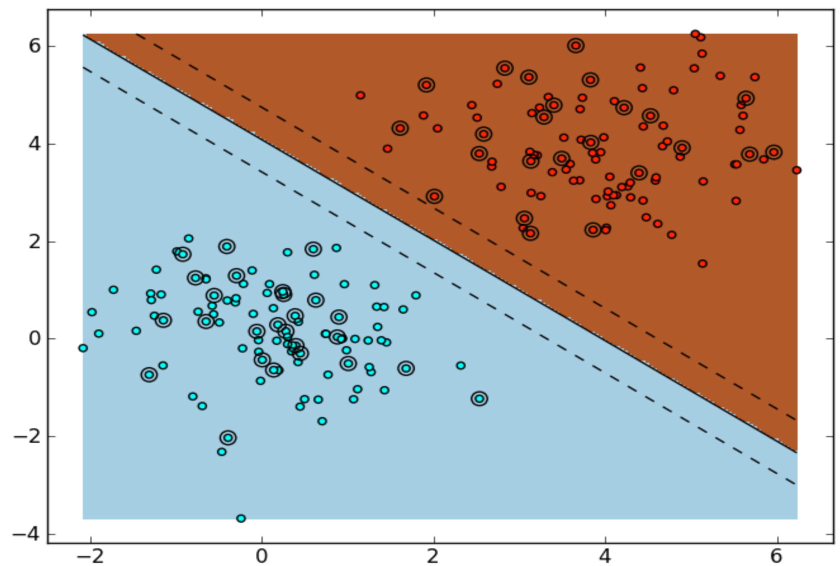
Input features

- Income range
- Criminal records
- # years in present job
- use credit card

Support Vector Machines [16]

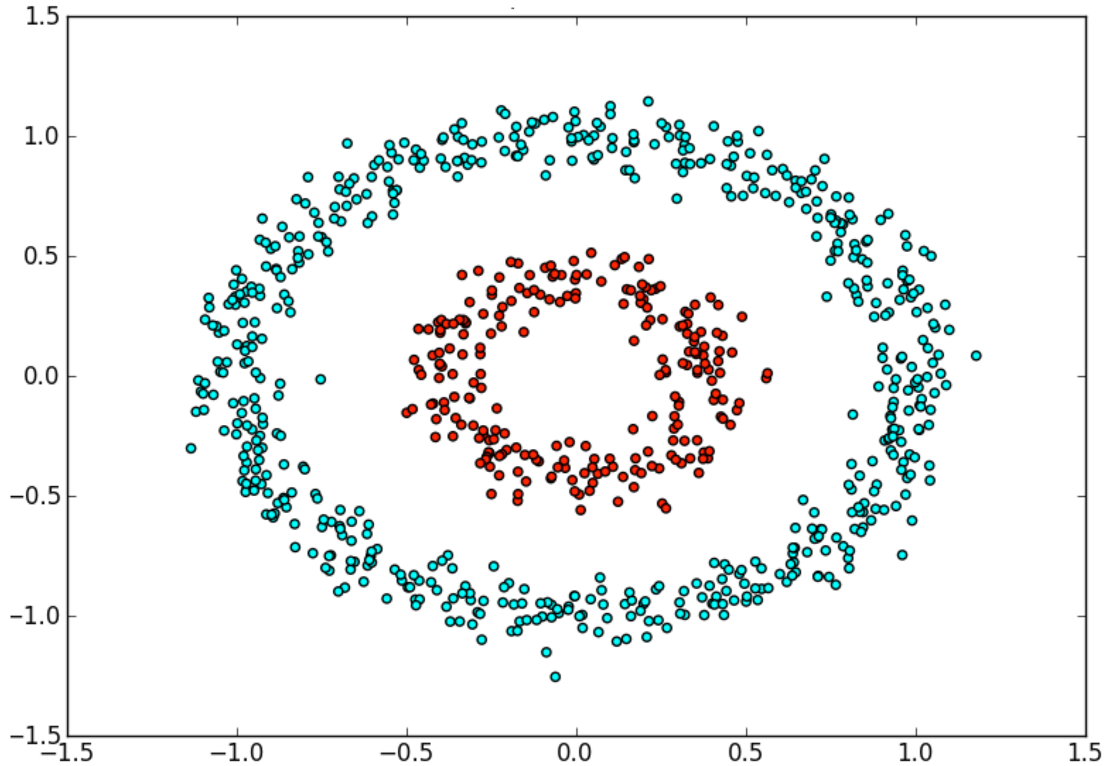
- A tuple is a point in a multidimensional space
- Find hyperplane s.t. different classes are as much distant as possible





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

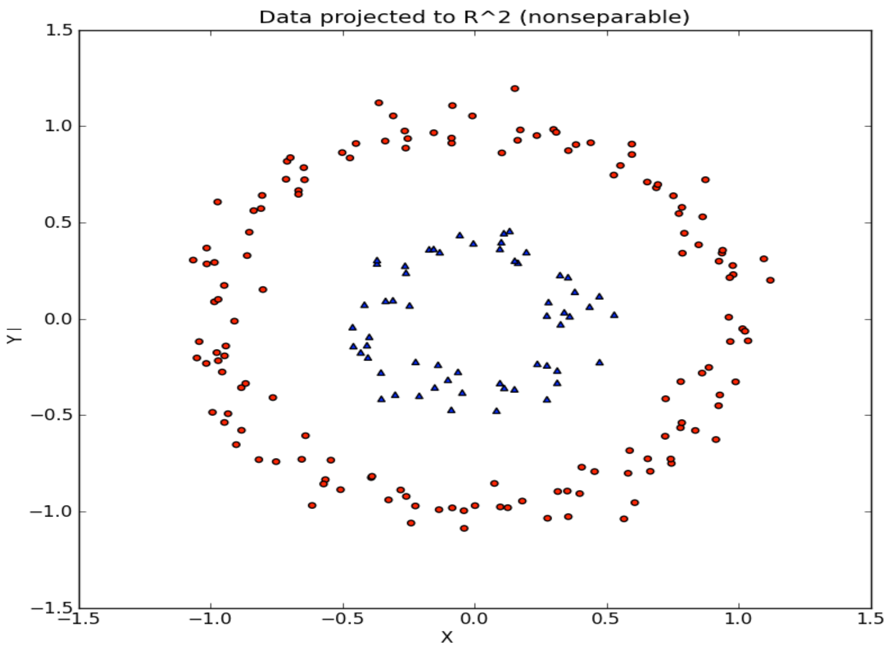
Support Vector Machines

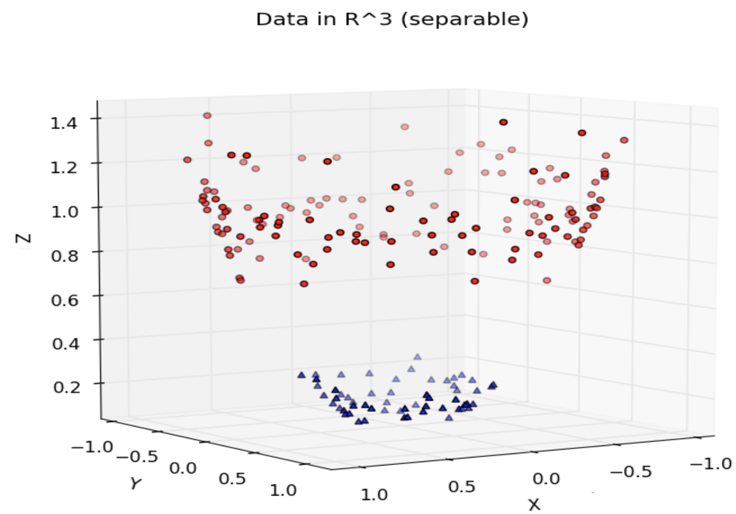


What if points are not linearly separable?

SVM: the kernel trick, I

- Map points to a higher dimensional space
- In that space, points are linearly separable

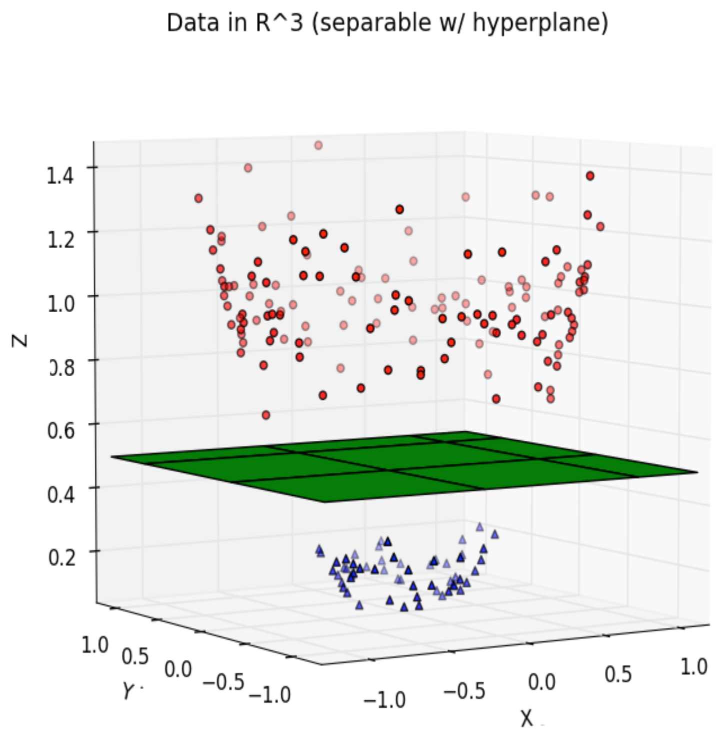


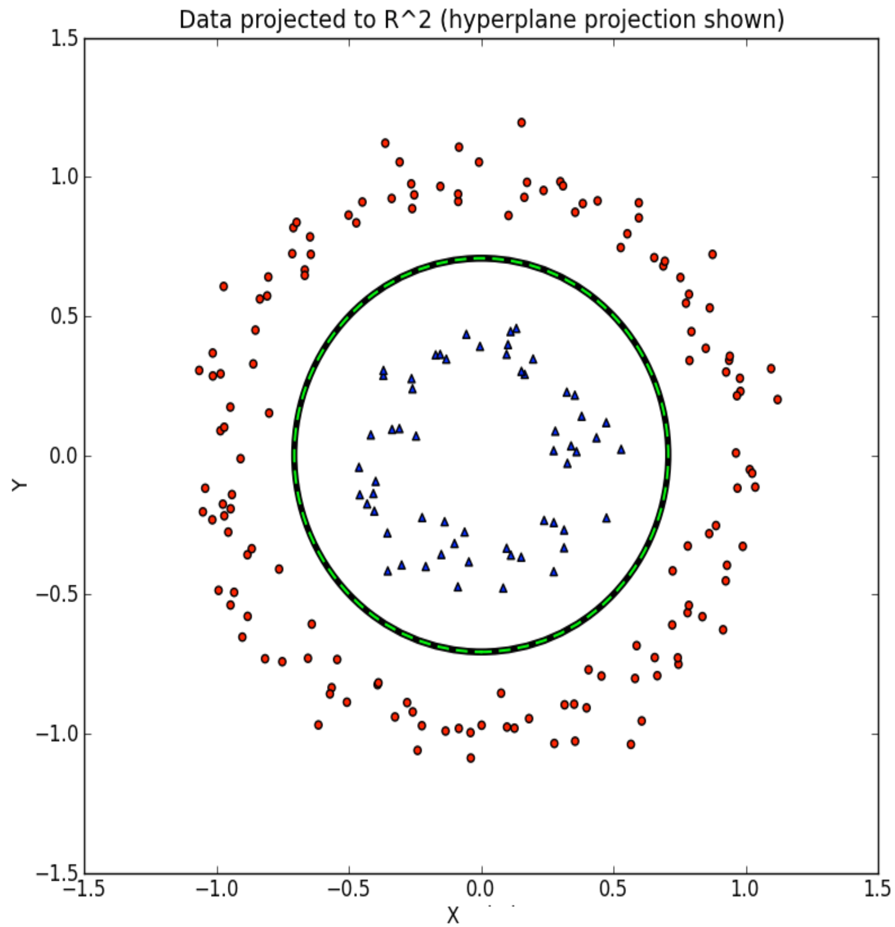


- Here, kernel is $f(x, y) = (x, y, x^2 + y^2)$

SVM: the kernel trick, II

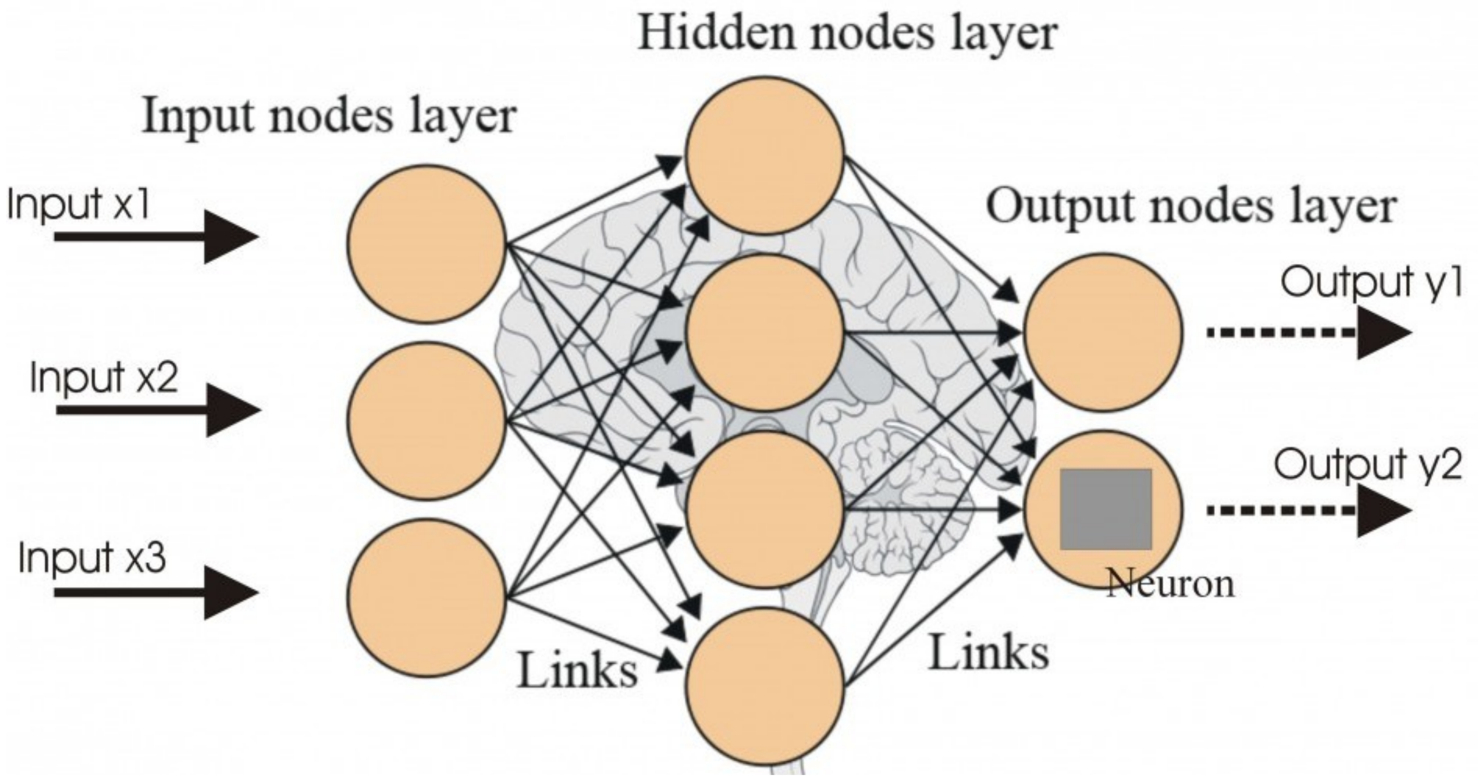
- Nonlinear separation in original domain





Artificial Neural Network [79]

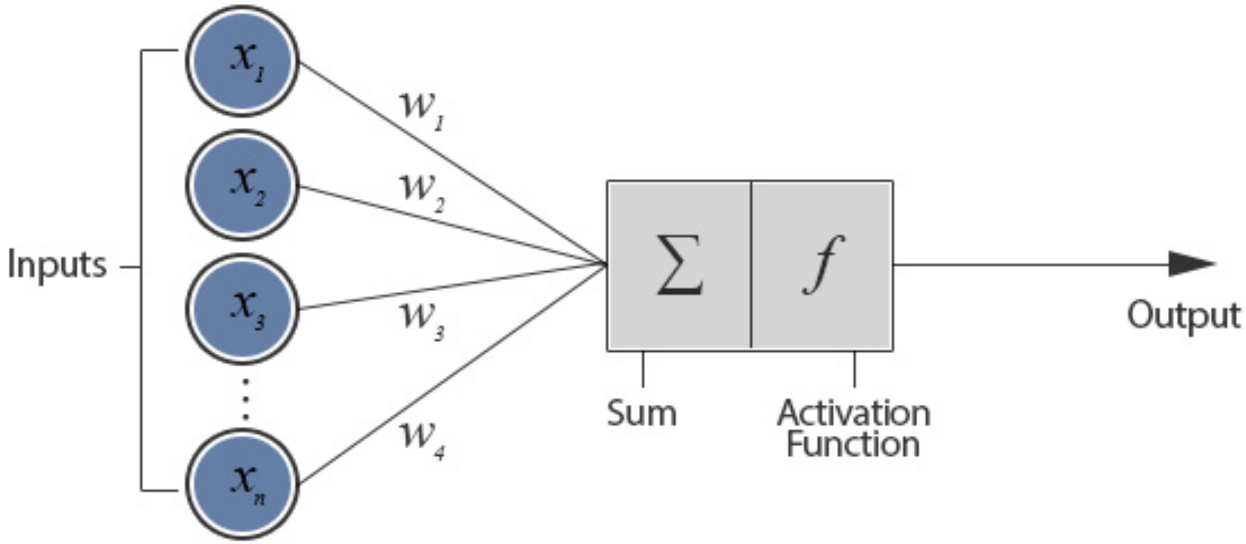
- Inner model is a graph
- Resembles neurons connections in brain



Credits to Koné Mamadou Tadiou for image
http://futurehumanevolution.com/artificial-intelligence-future-human-evolution/artificial-neural-networks

ANN internals

- Neuron structure

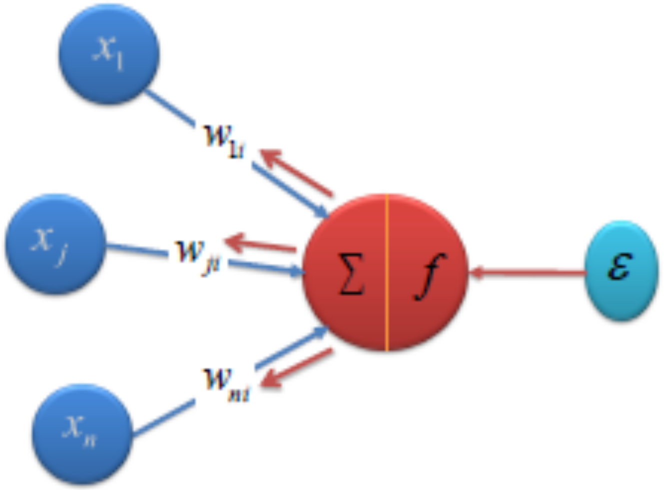


- Weighted sum of inputs
- Activation 0/1 function as output

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

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

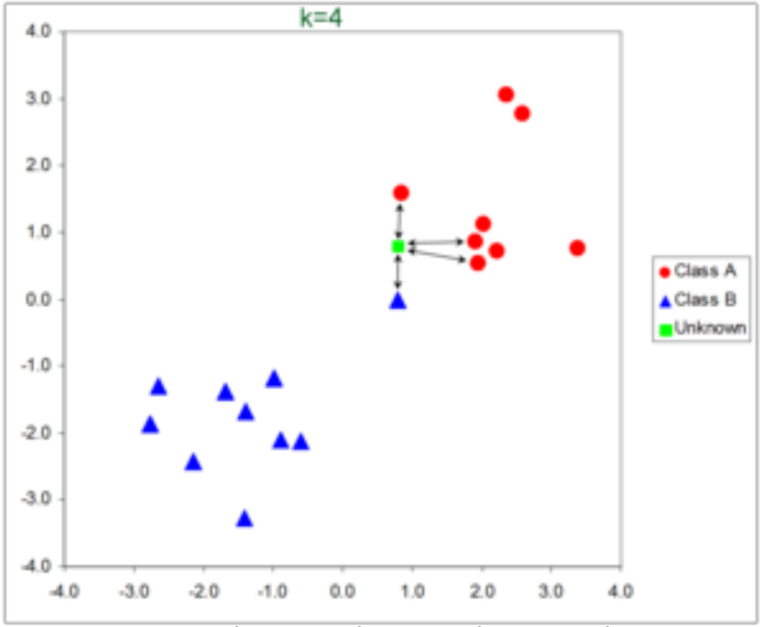
Manhattan $$\sum_{i=1}^{k}|x_i - y_i|$$

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

Pic from http://www.cs.bham.ac.uk/internal/courses/robotics/halloffame/2010/team12/knn.htm

K Nearest Neighbors

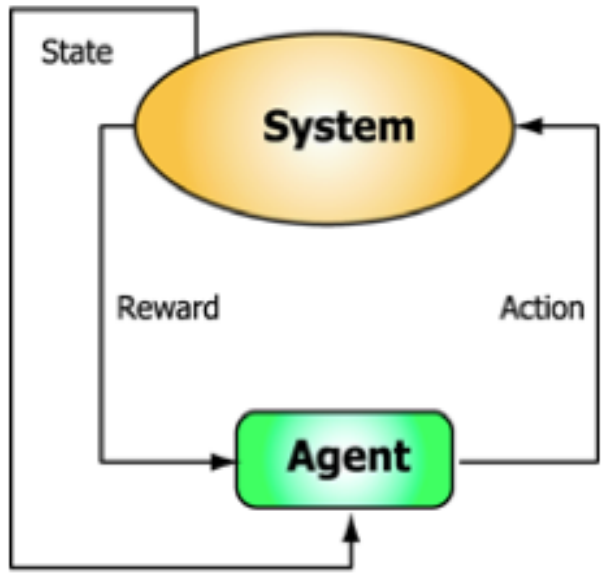
- Classification:
  - Class of X is the most common in neighborhood
- Regression
  - Value for X is a function of the values in the neighb.



Pic from http://www.cs.bham.ac.uk/internal/courses/robotics/halloffame/2010/team12/knn.htm

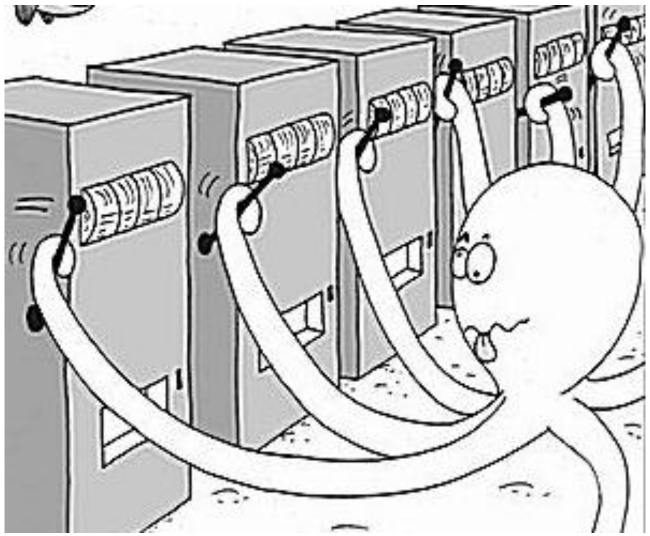
ONLINE LEARNING

- 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]

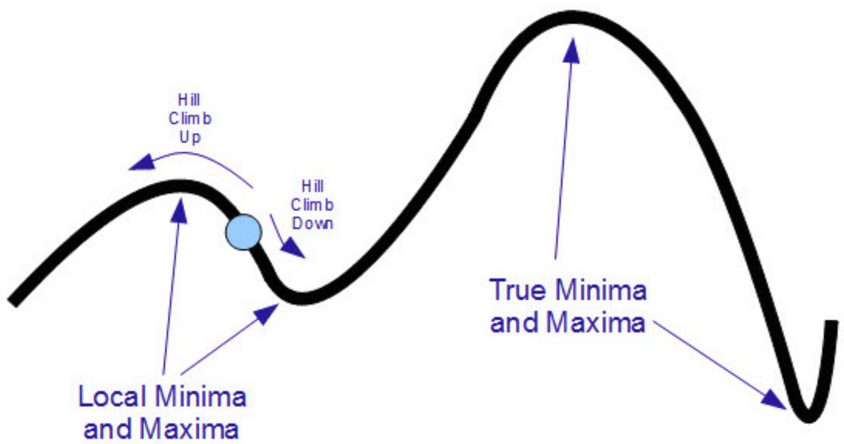
- Popular set of algorithms for MAB

- 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



Credit to Ben Etzkorn for pic. http://www.benetzkorn.com/2011/09/non-linear-model-optimization-using-palisades-evolver/

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

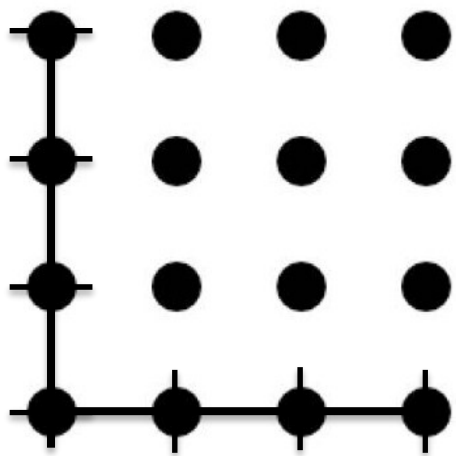
Hyperparameters optimization

- 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]

Grid Search

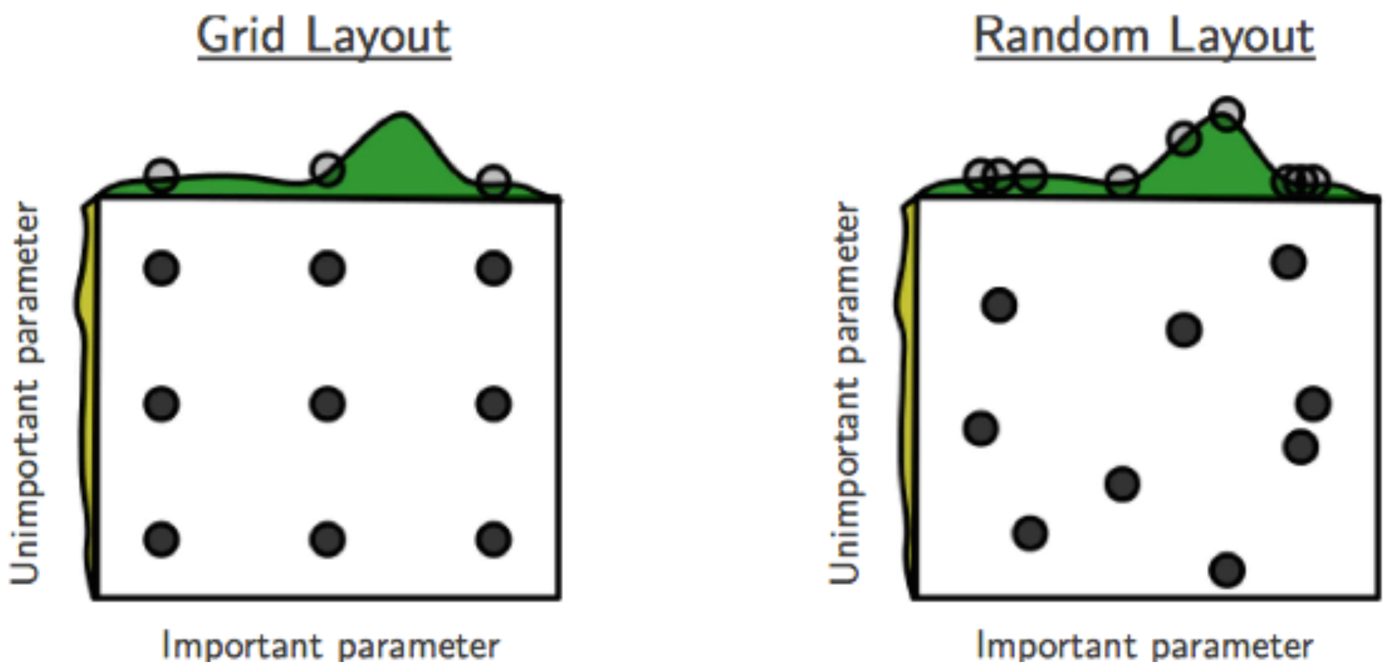
1. Uniformly discretize features' domain

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

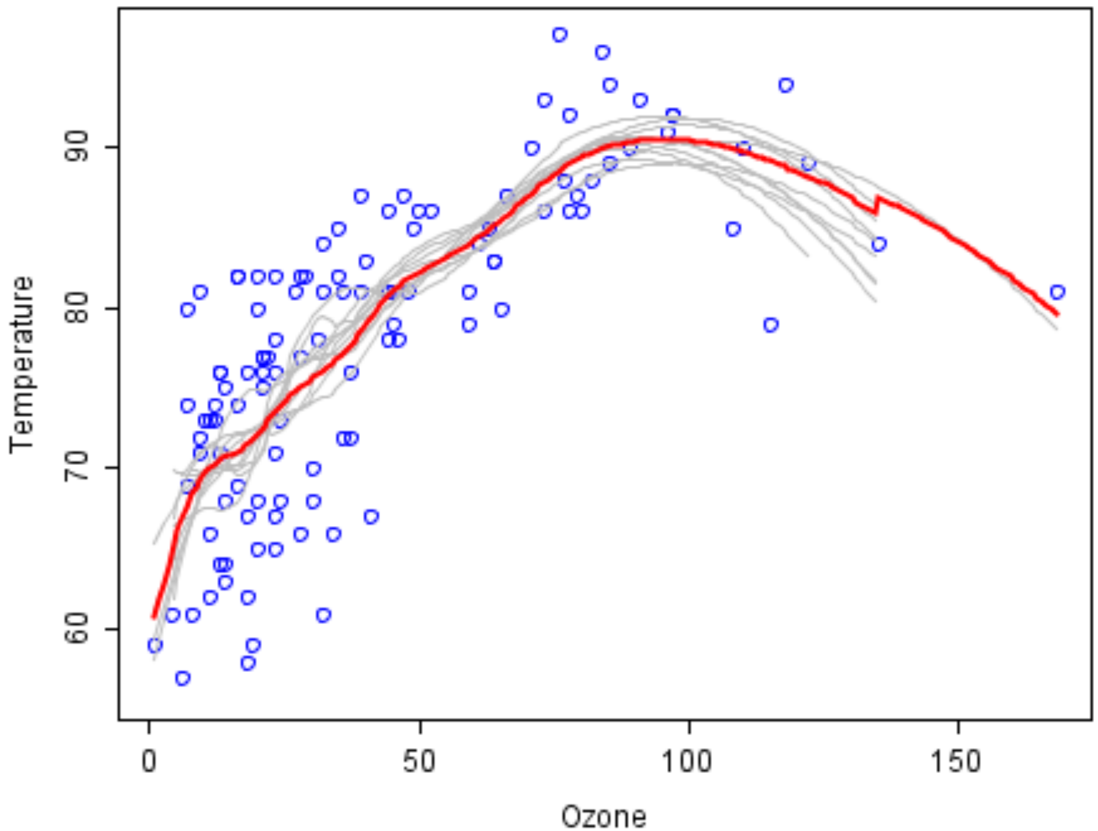
Bagging

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 extrap.) …
- Can reduce variance (i.e., better interp. accuracy)

Bagging example

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

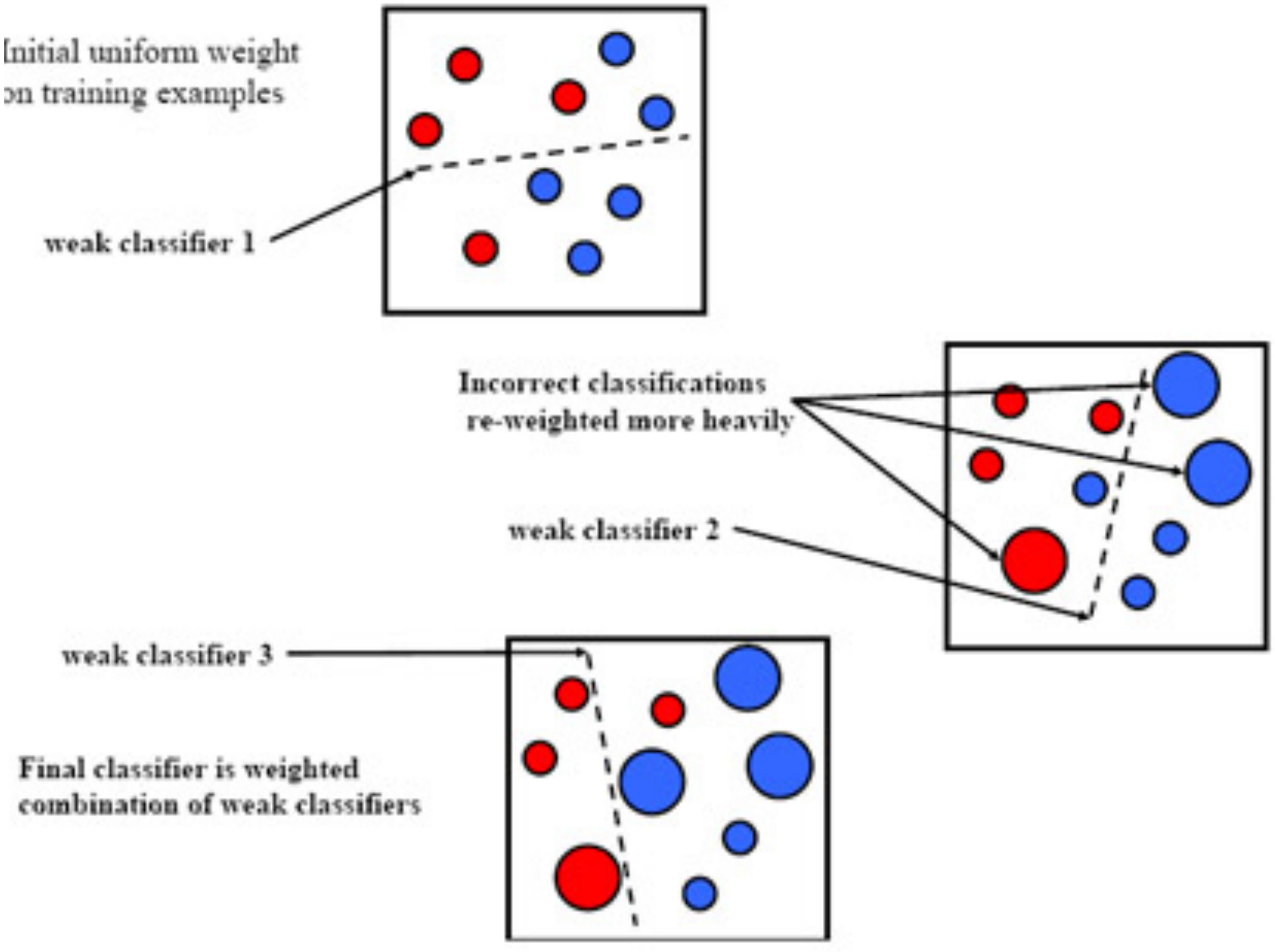


Boosting

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 $p > ½$
  - 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

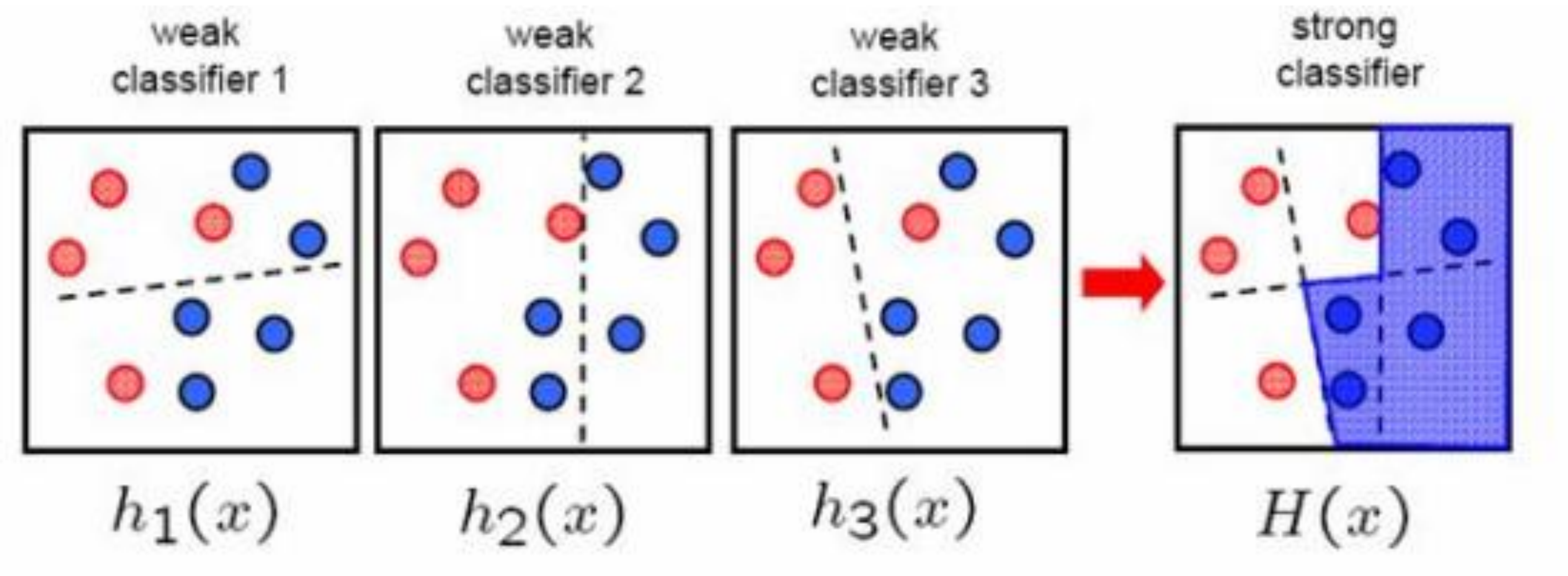
Adaboost, training



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

Adaboost, result

$$H(x) = sign(\alpha_1 h_1(x) + \alpha_2 h_2(x) + \alpha_3 h_3(x))$$

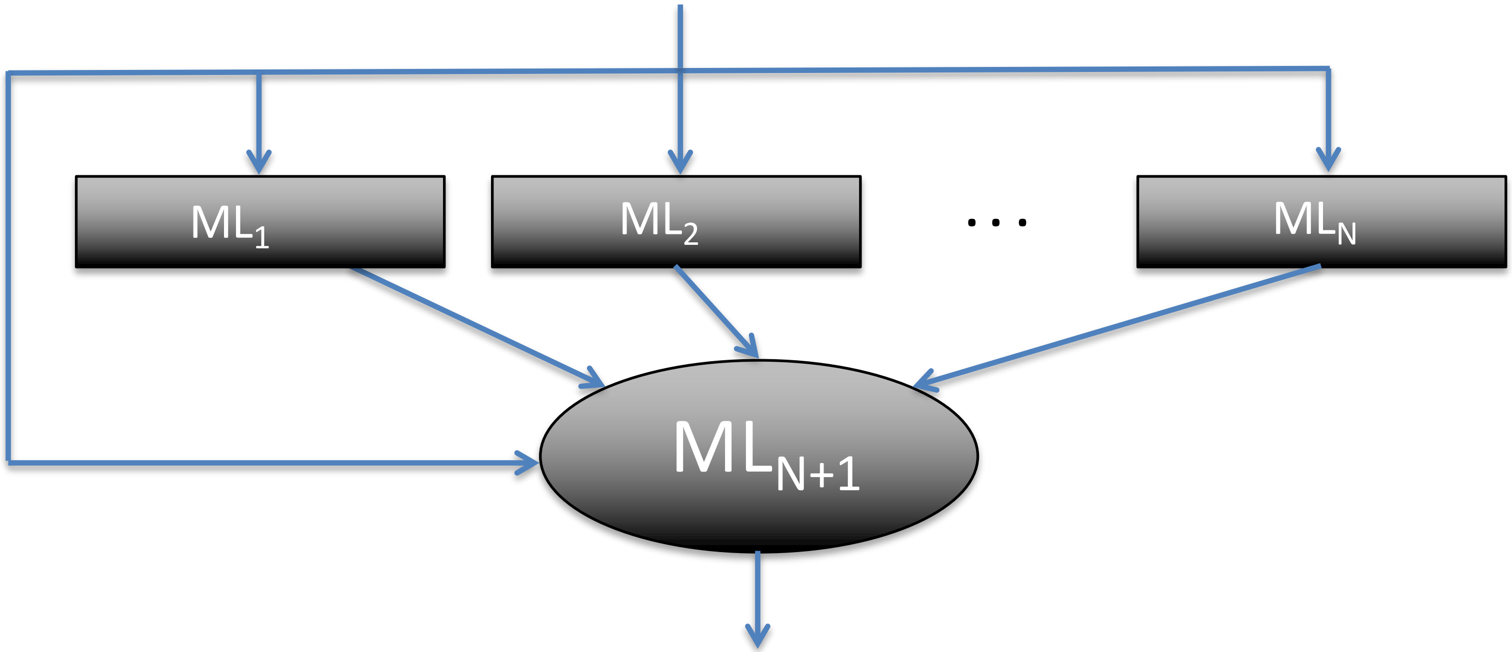


Pictures from http://www.ieev.org/2010/03/adaboost-haar-features-face-detection.html

Stacking

A meta-learner combine output of ML

- Partition D in D', D''
- Train 1…N learners on D'
- ML_{N+1} trained on $ML_1 \ldots ML_N$ predictions on D''



Introduction and Modeling of Main Case Studies

Background Case studies

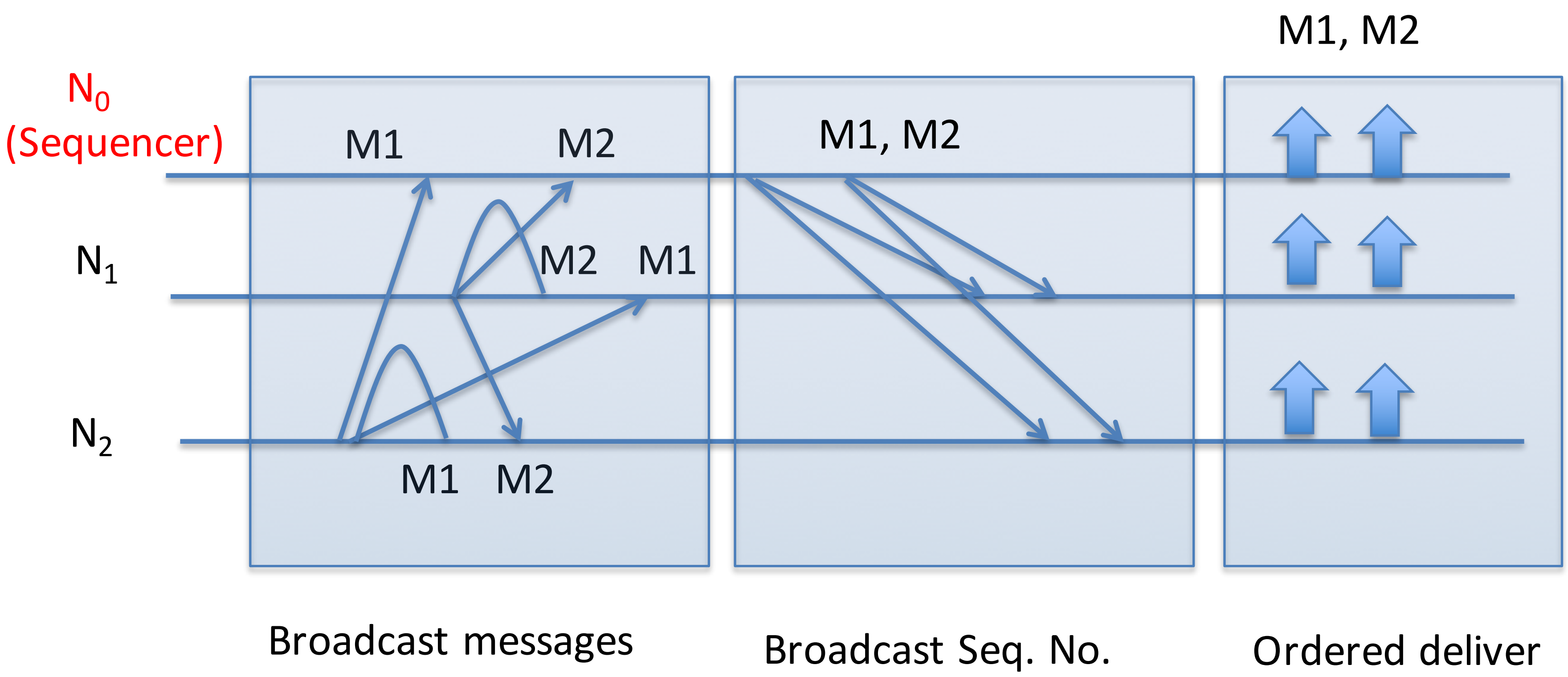
- Total Order Broadcast primitive
  - Analytical model
  - Black box online optimization

- 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



Performance of STOB

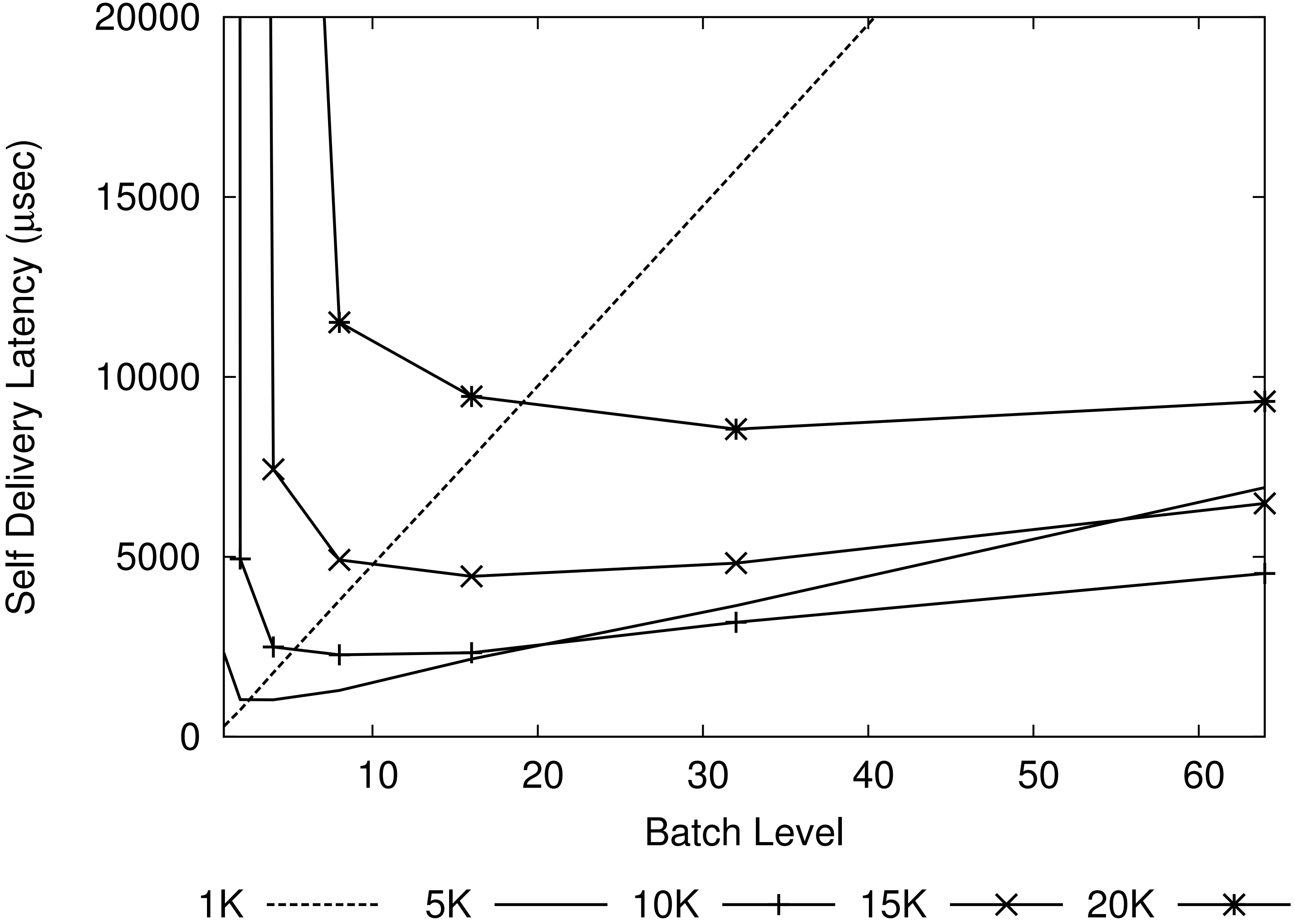
- STOB minimizes messages exchange, but…
- The sequencer may become the bottleneck
- Possible solution: batching
- The sequencer
  - Waits to receive N > 1 msgs
  - Send a single, bigger seq. msg for the N msgs instead of N smaller

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

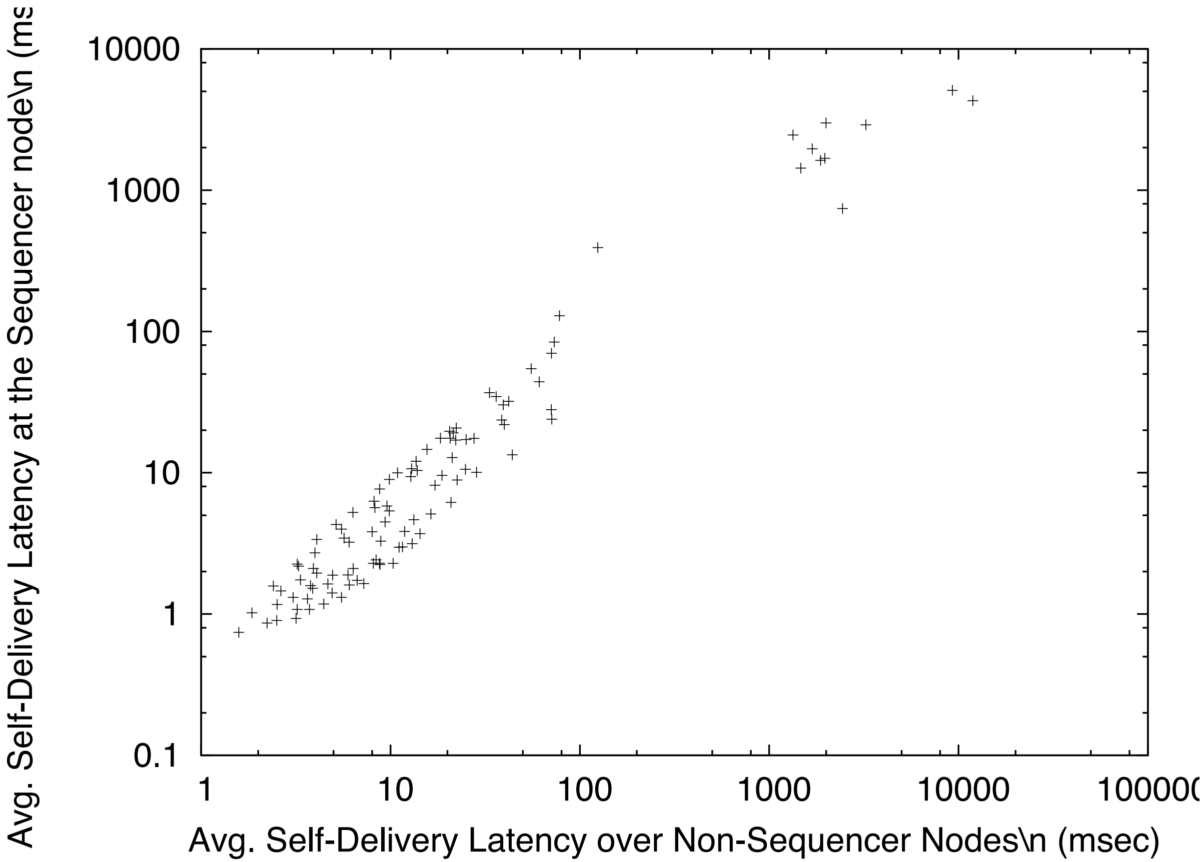


Tuning the batching level

- 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 1st message in batch
- T_Add = time to process additional msgs
  - Batching makes sense when T_1>T_Add

STOB analytical model [59]

- Sequencer = M/M/1 queue $T(b,m) = \frac{1}{\mu(b,m) - \lambda(b,m)}$

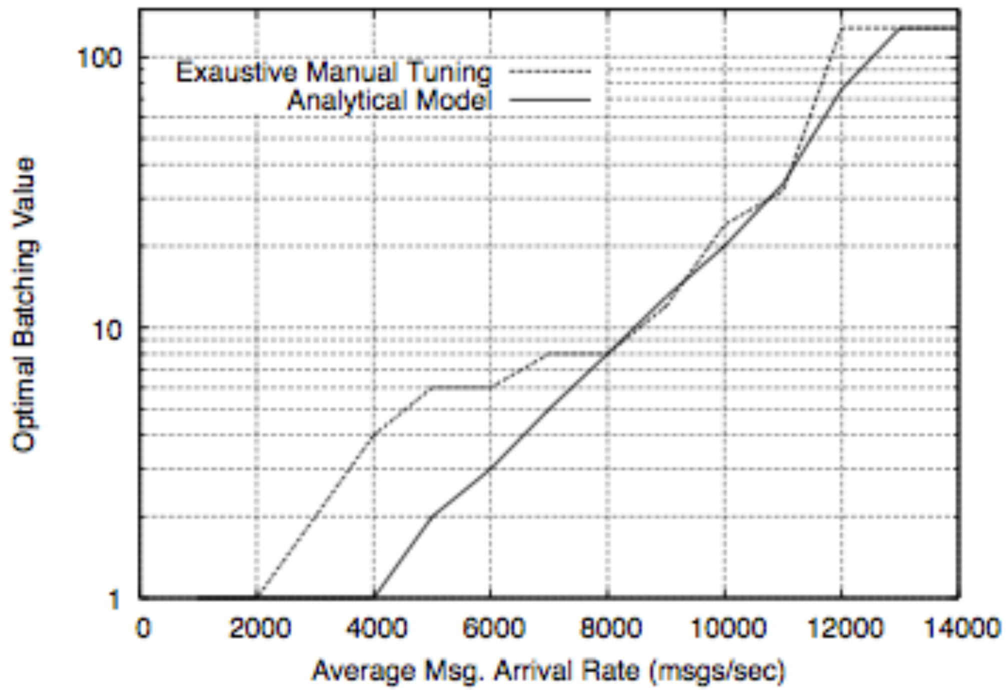
- Batch generation rate $\lambda(b,m) = \frac{m}{b}$

- 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



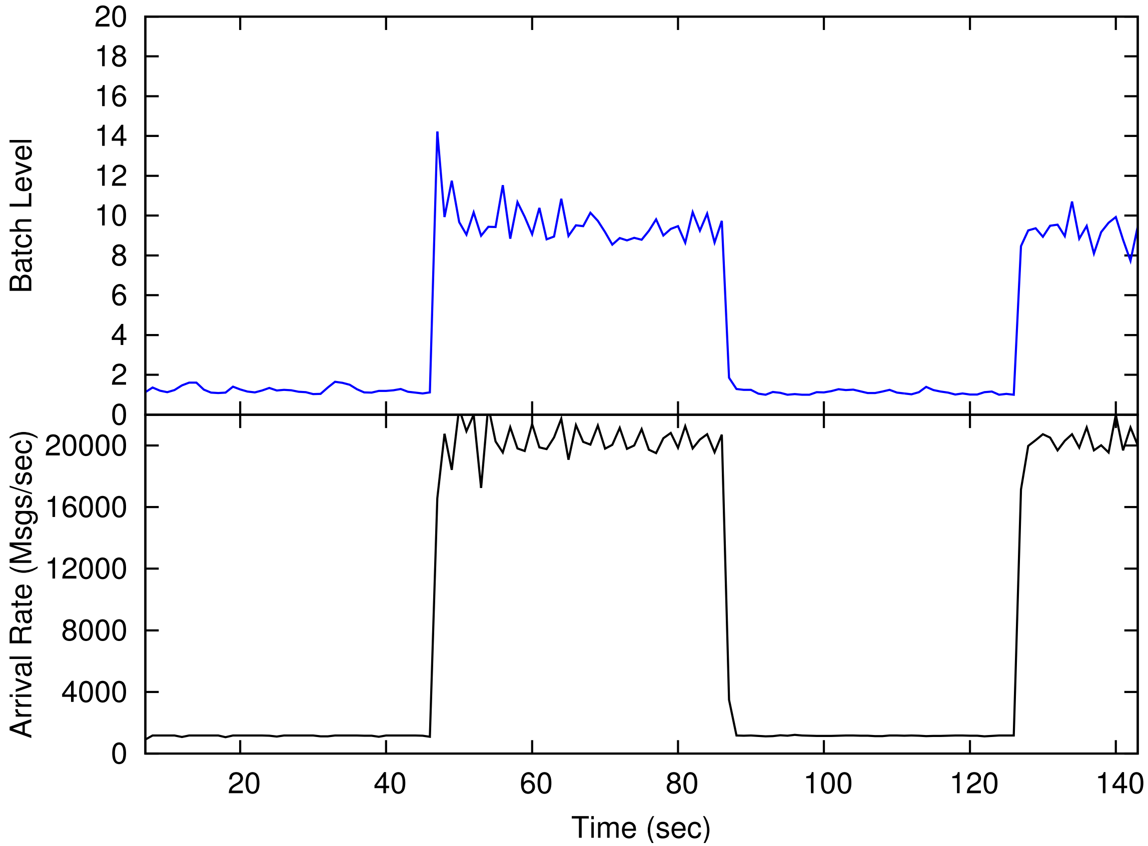
STOB black box optimization [24]

- 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

Hill Climbing in STOB

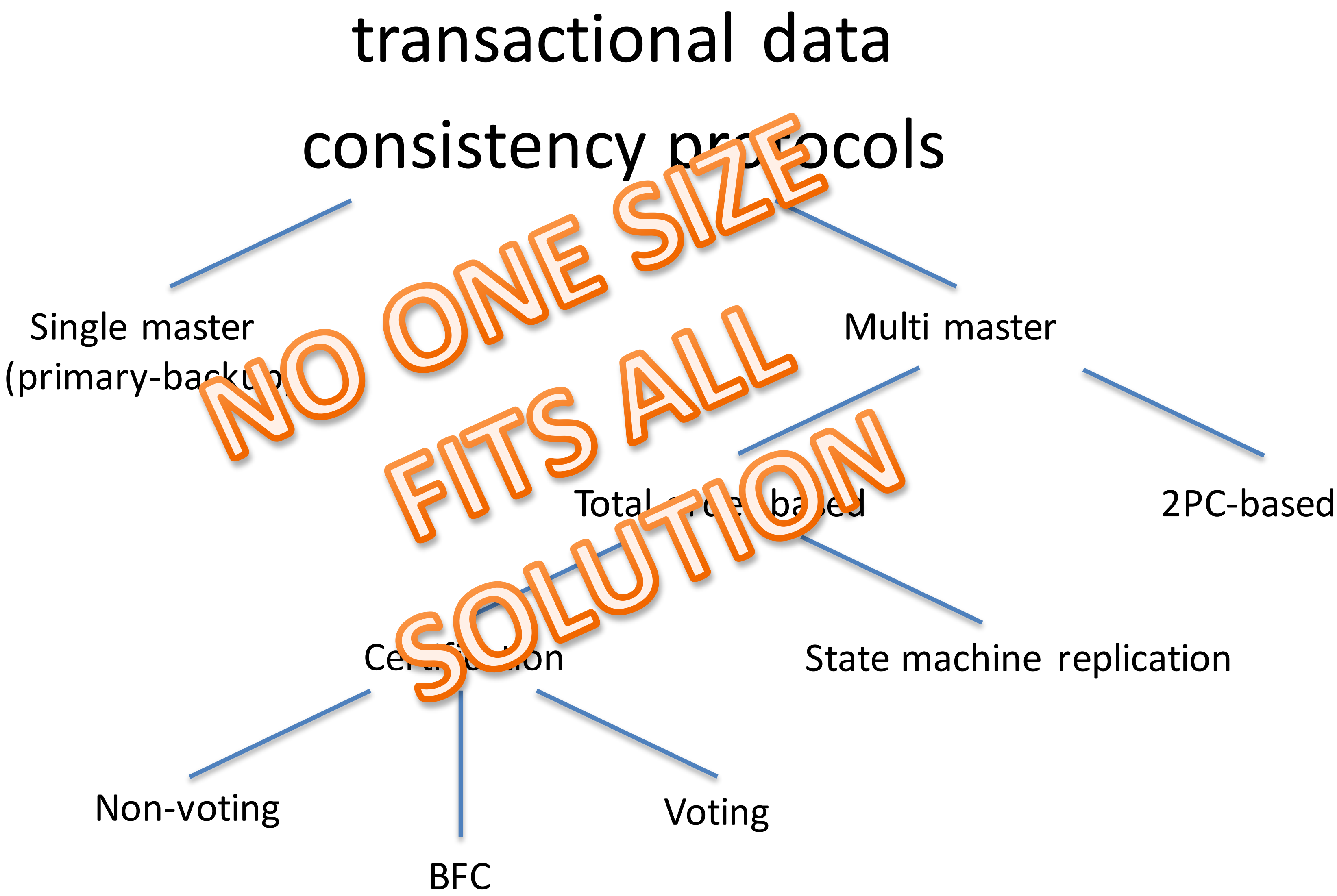


- But limited expressiveness:
  - Self-tuning at the cost of no predictability

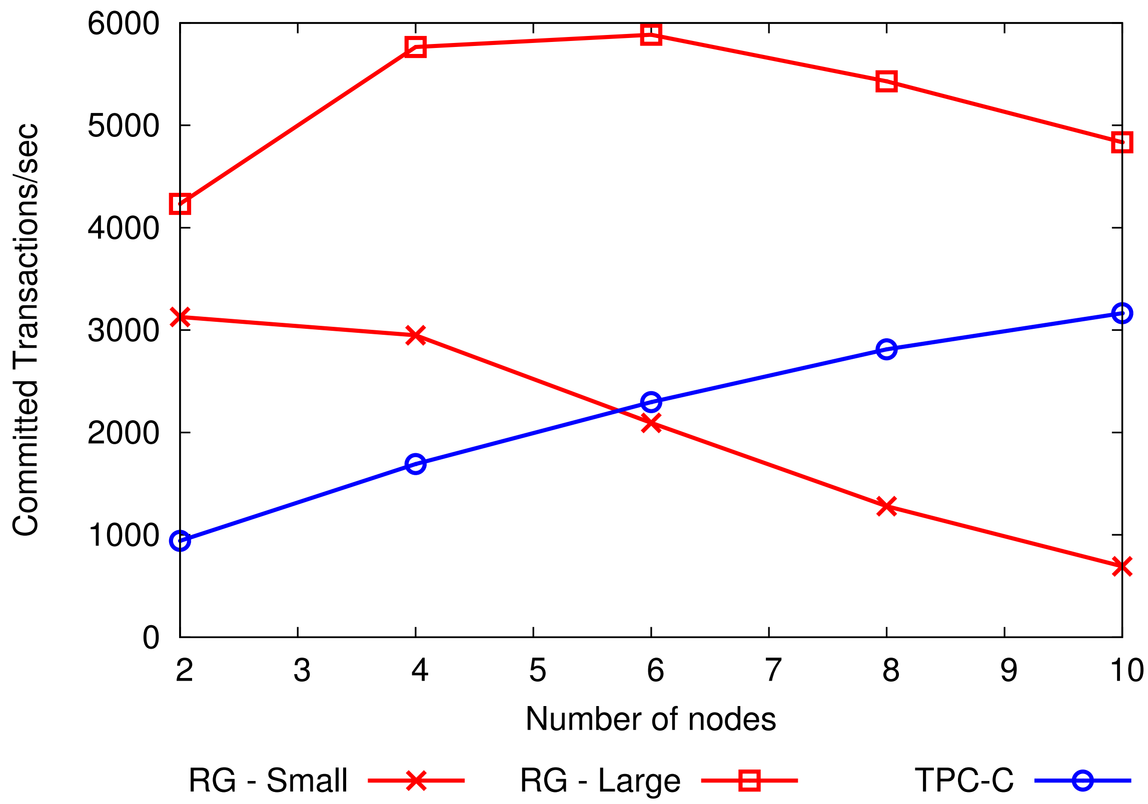
Transactional NoSQL store case study

- 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?

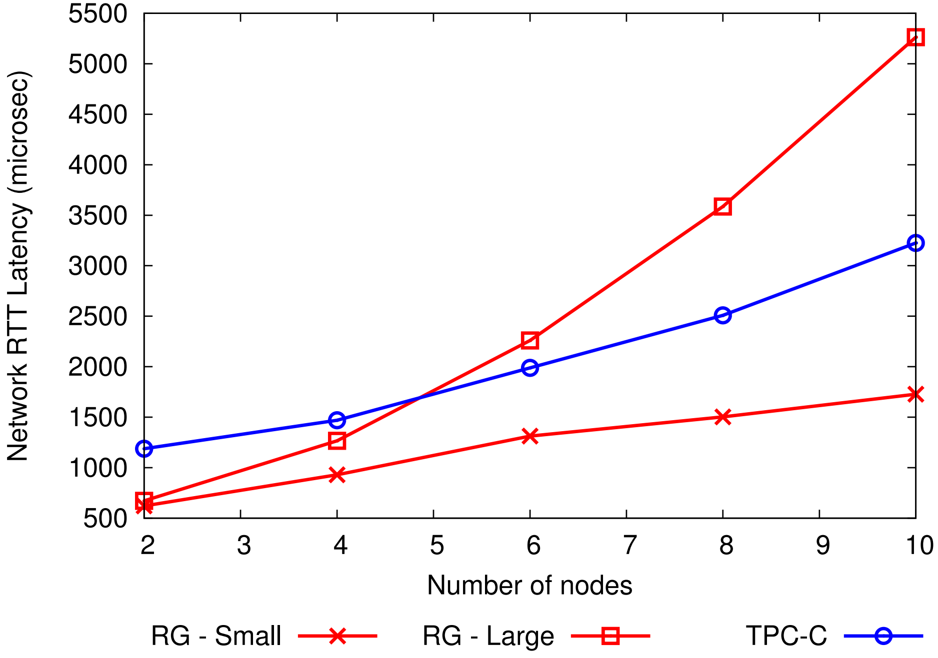


DSTM Performance

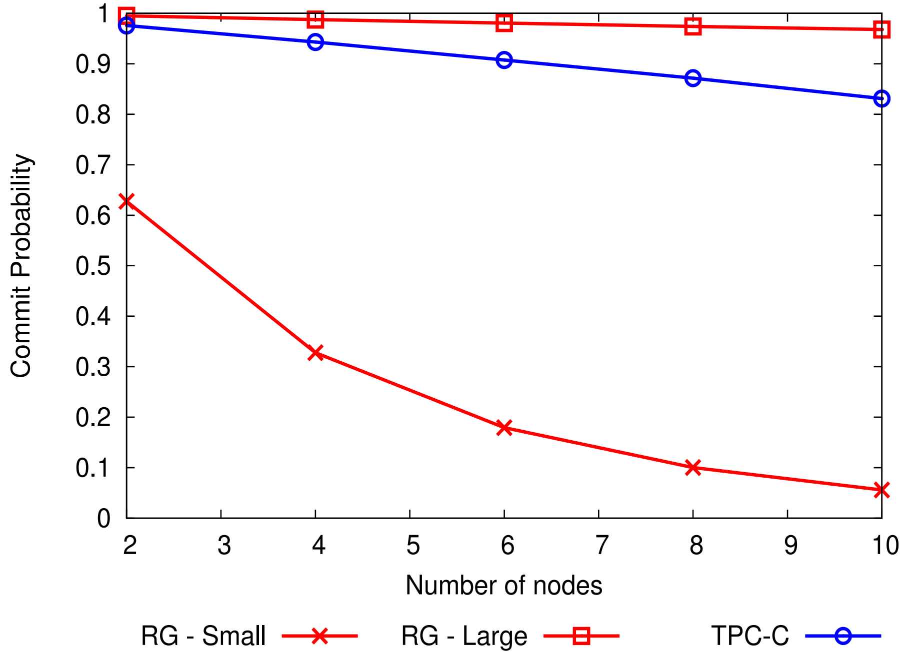


- Heterogeneous, nonlinear scalability trends!

Factors limiting scalability

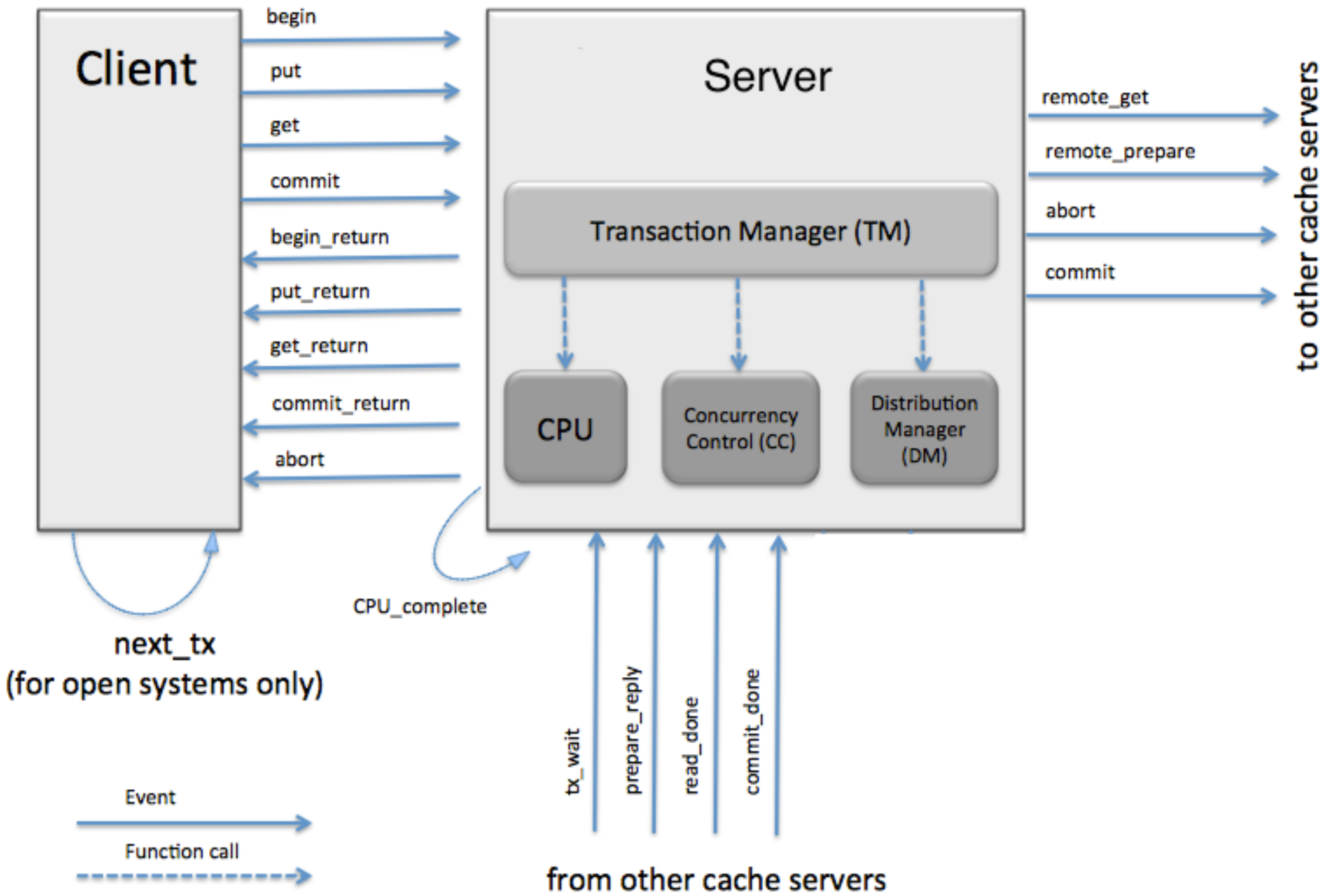


Network latency in commit phase



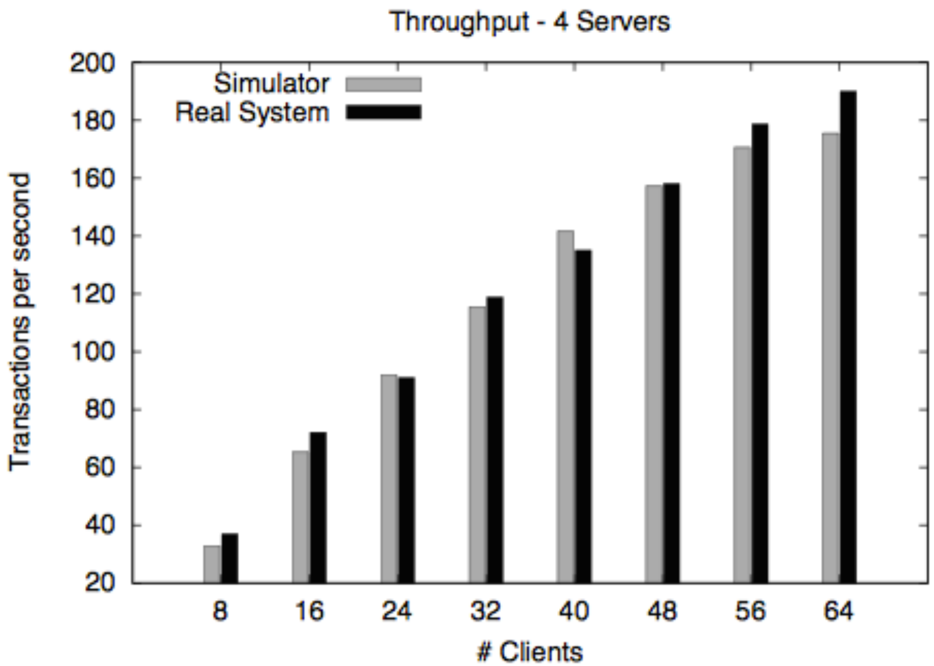
Aborted transactions because of conflicts

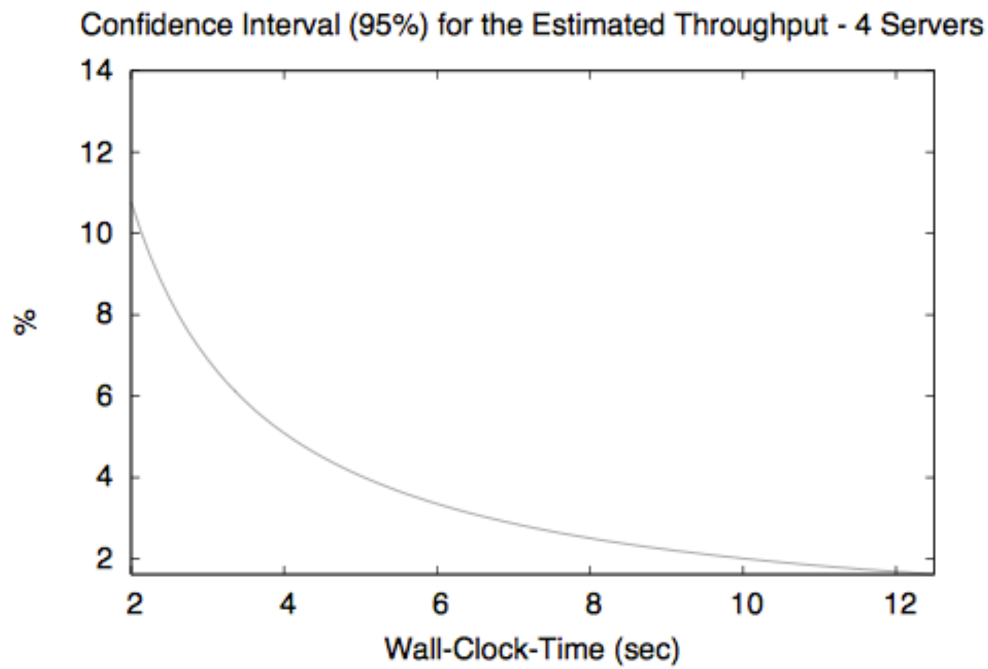
White box modeling



Simulator [21]

- Assumptions and approximations
  - CPU = G/**M**/K
  - Fixed point to point network latency
- Accuracy / resolution time trade-off

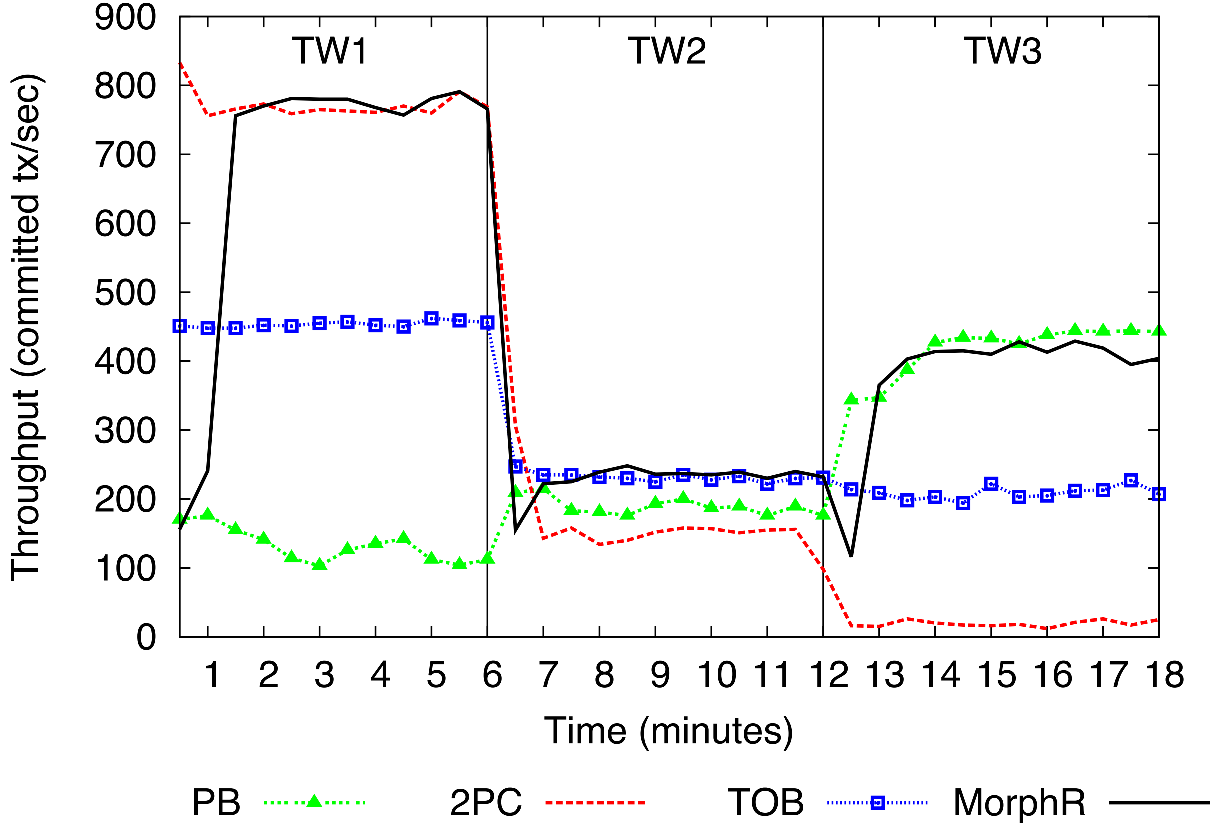




Black box modeling

- 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

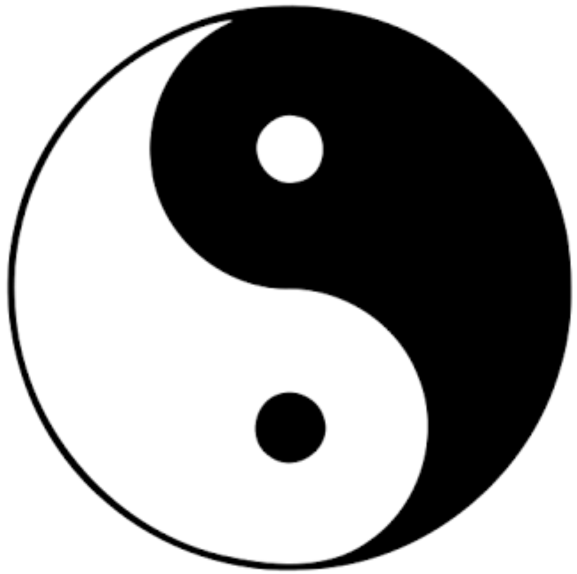
MorphR in action



Gray Box Modeling

Gray box modeling

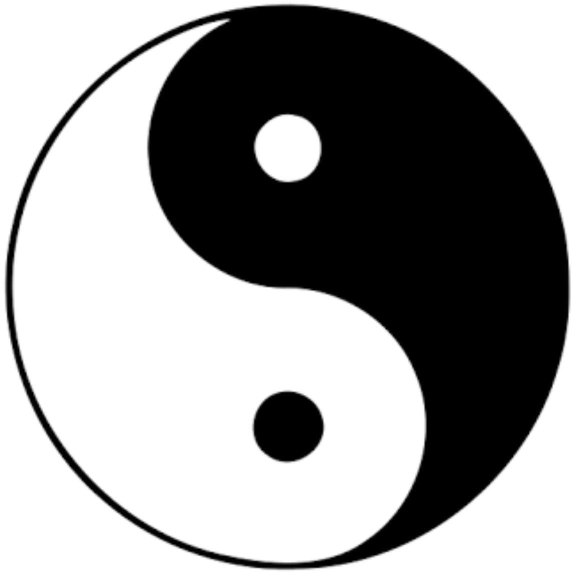
- Combine WB and BB modeling
  - Lower training time thx to WBM
  - Incremental learning thx to BBM



- 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…
- 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

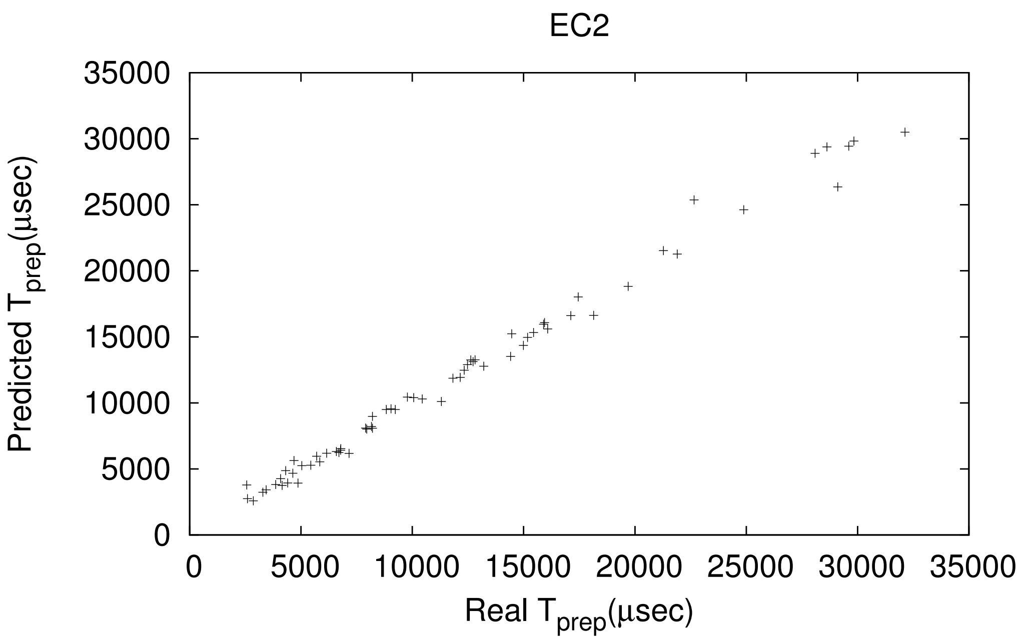
- Concurrency control scheme (lock-based)
  - A lock is a M/G/1 server
  - Conflict prob = utilization of the server
- Replication protocol
  - 2PC: all nodes are similar → one model
  - PR: primary vs backups → two models
- Replication scheme
  - Probability of accessing remote data
  - # nodes involved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

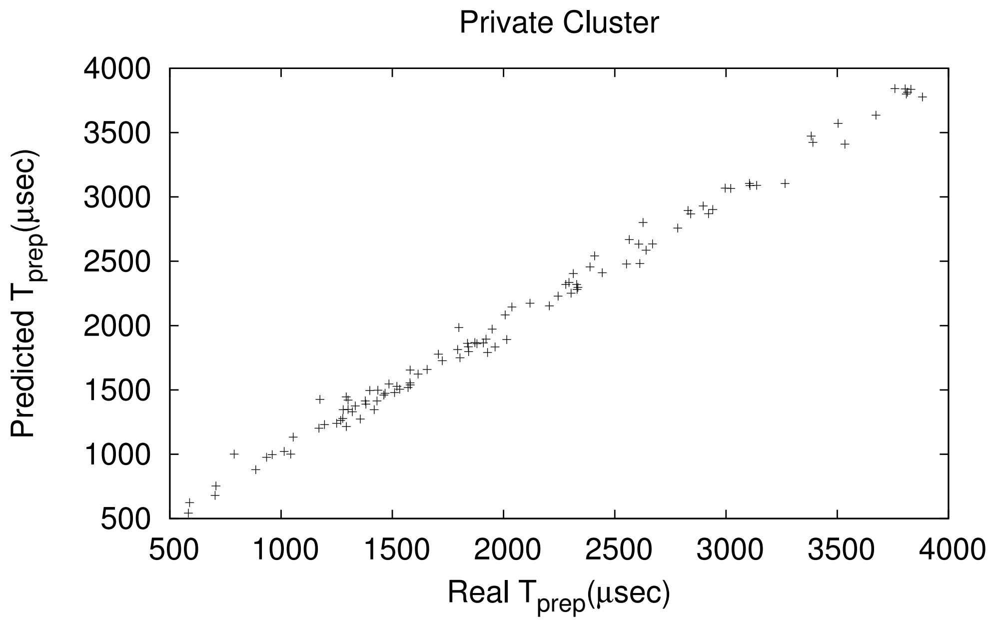
Machine Learning in TAS/PROMPT

- 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!

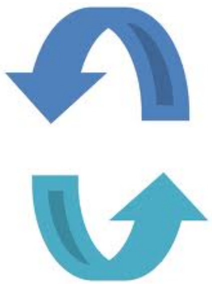
EXAMPLE

- 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

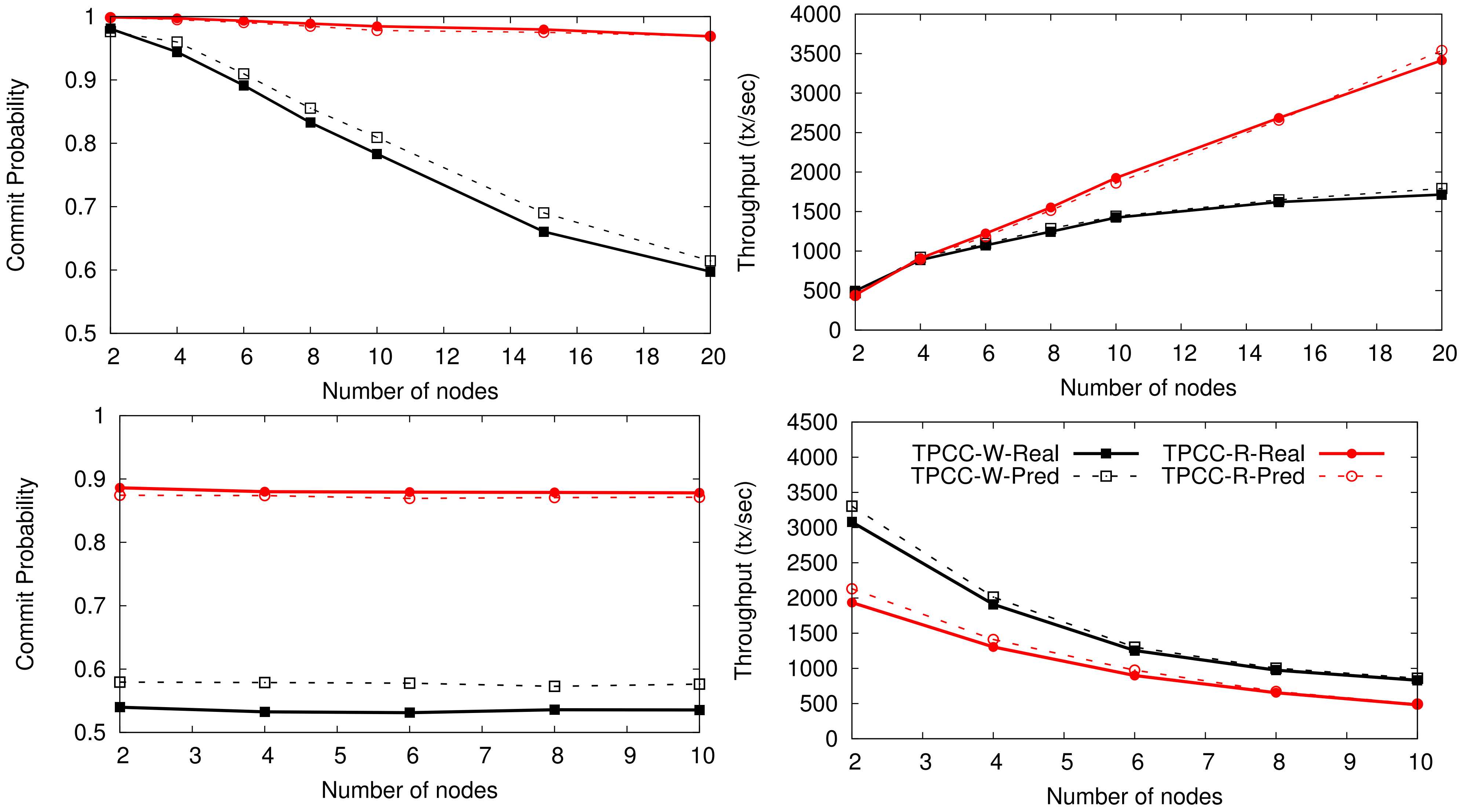
- AM can provide (estimates of) missing input
- Iterative coupling scheme

ML takes some input parameters from AM



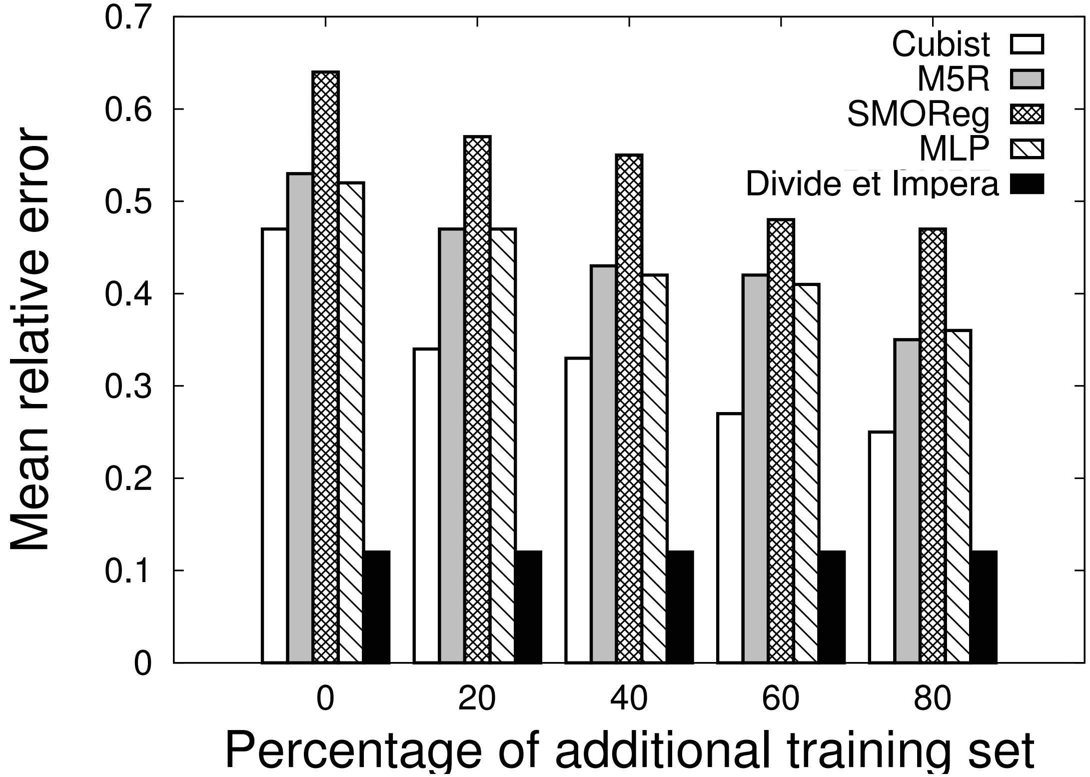
AM takes latencies forecast by ML as input parameter

Model's accuracy



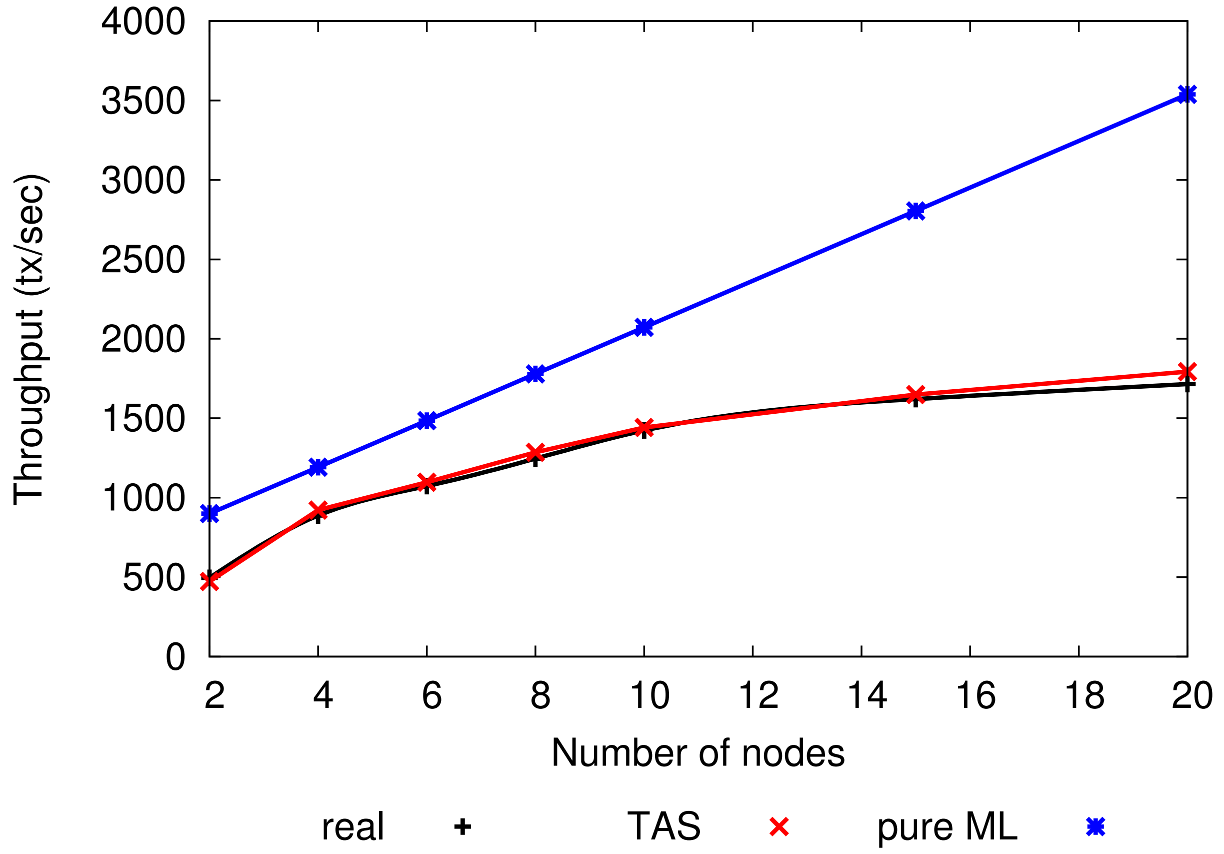
TOP: PB, only master node. BOTTOM: 2PC. FULL REPL.

COMPARISON WITH PURE BLACK, I



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

COMPARISON WITH PURE BLACK, II



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

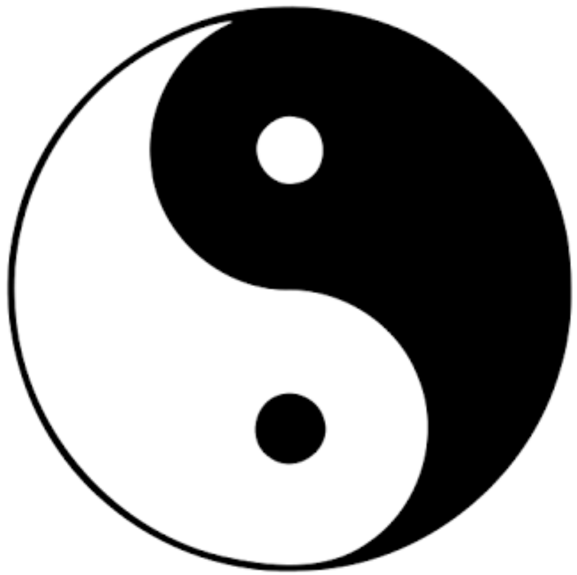
TAS/PROMPT integration

- 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

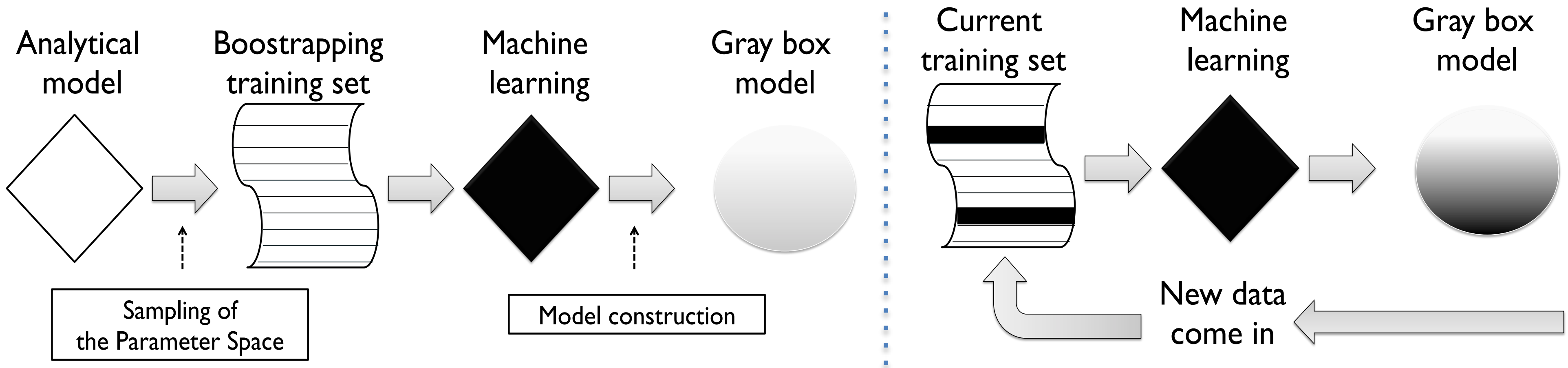
- Techniques in this tutorial
  - Divide et impera
  - Bootstrapping
  - Hybrid ensembling



BOOTSTRAPPING [27]

Obtain zero-training-time ML via initial AM

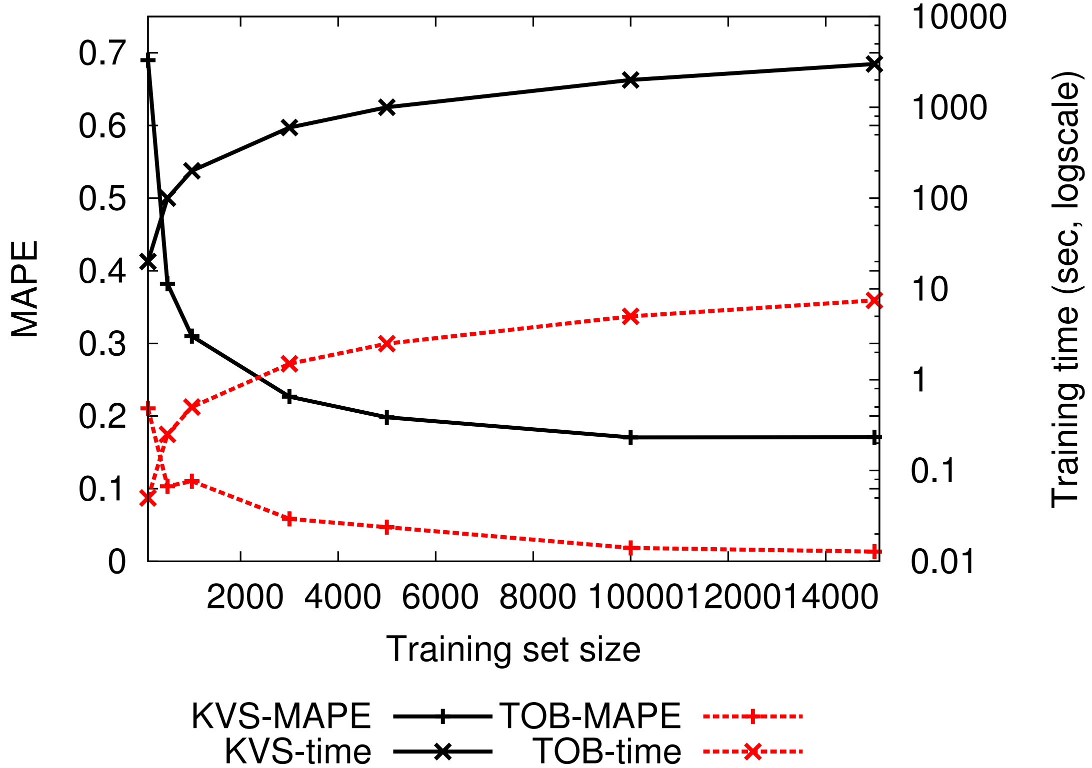
1. Initial (synthetic) training set of ML from AM
2. Retrain periodically with “real” samples



(1
)

(2)

How many synthetic samples?

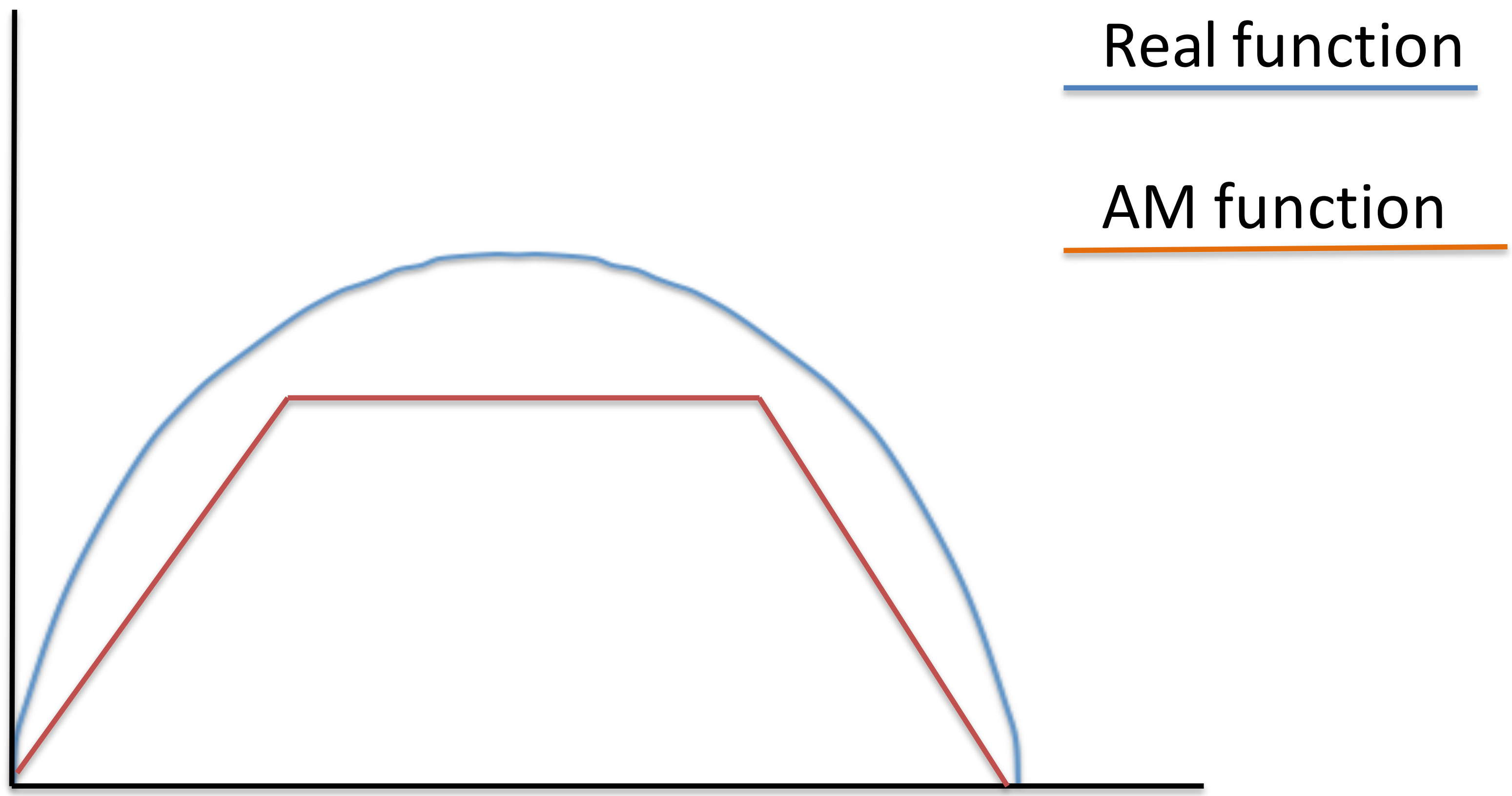


- Important tradeoff
  - Higher # → lower fitting error over the AM output
  - Lower # → higher density of real samples in dataset

How to update

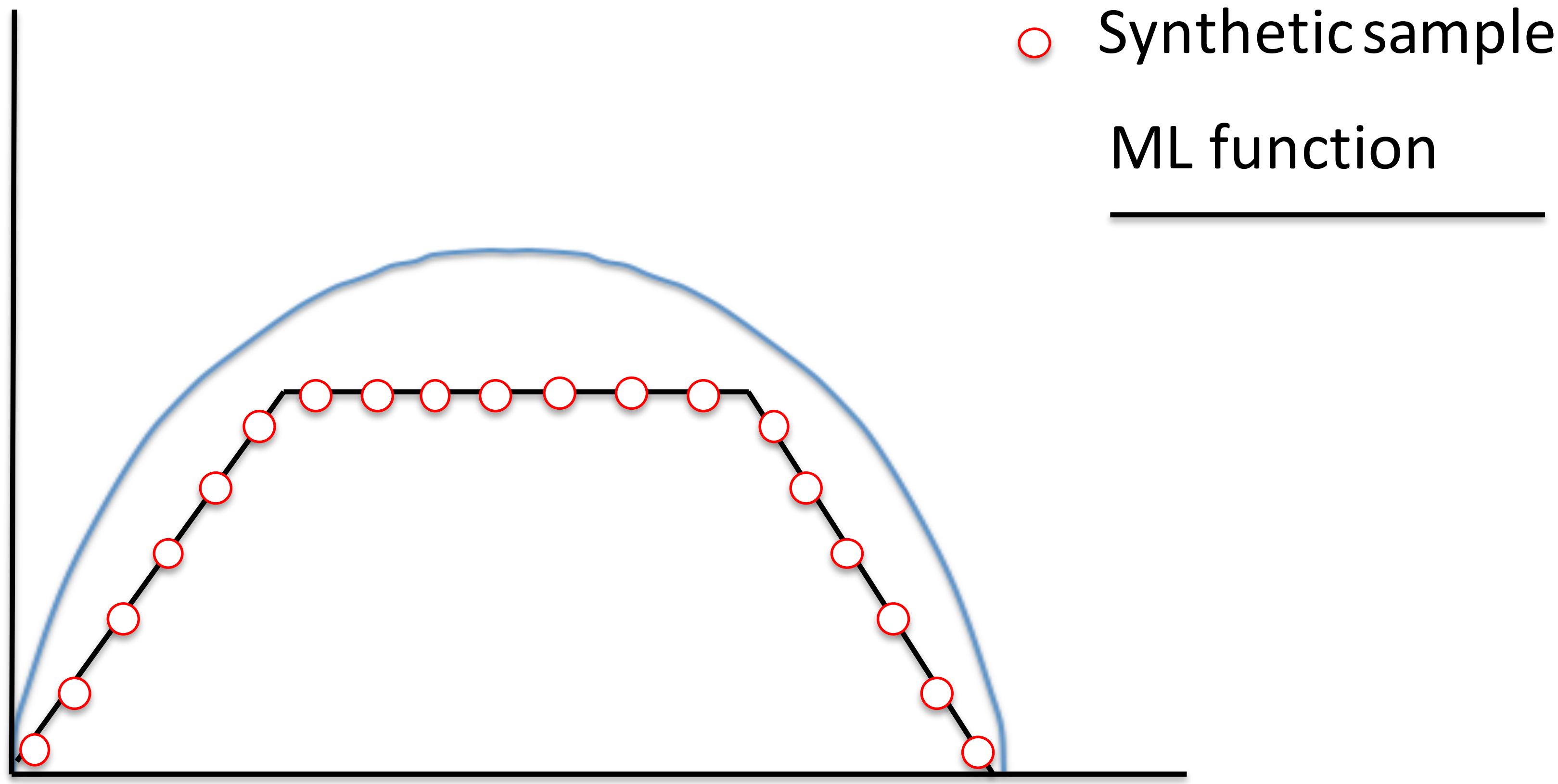
- Merge: simply add real samples to synthetic set
- Replace only the nearest neighbor (RNN)
- Replace neighbors in a given region (RNR)
  - Two variants

Real vs AM function



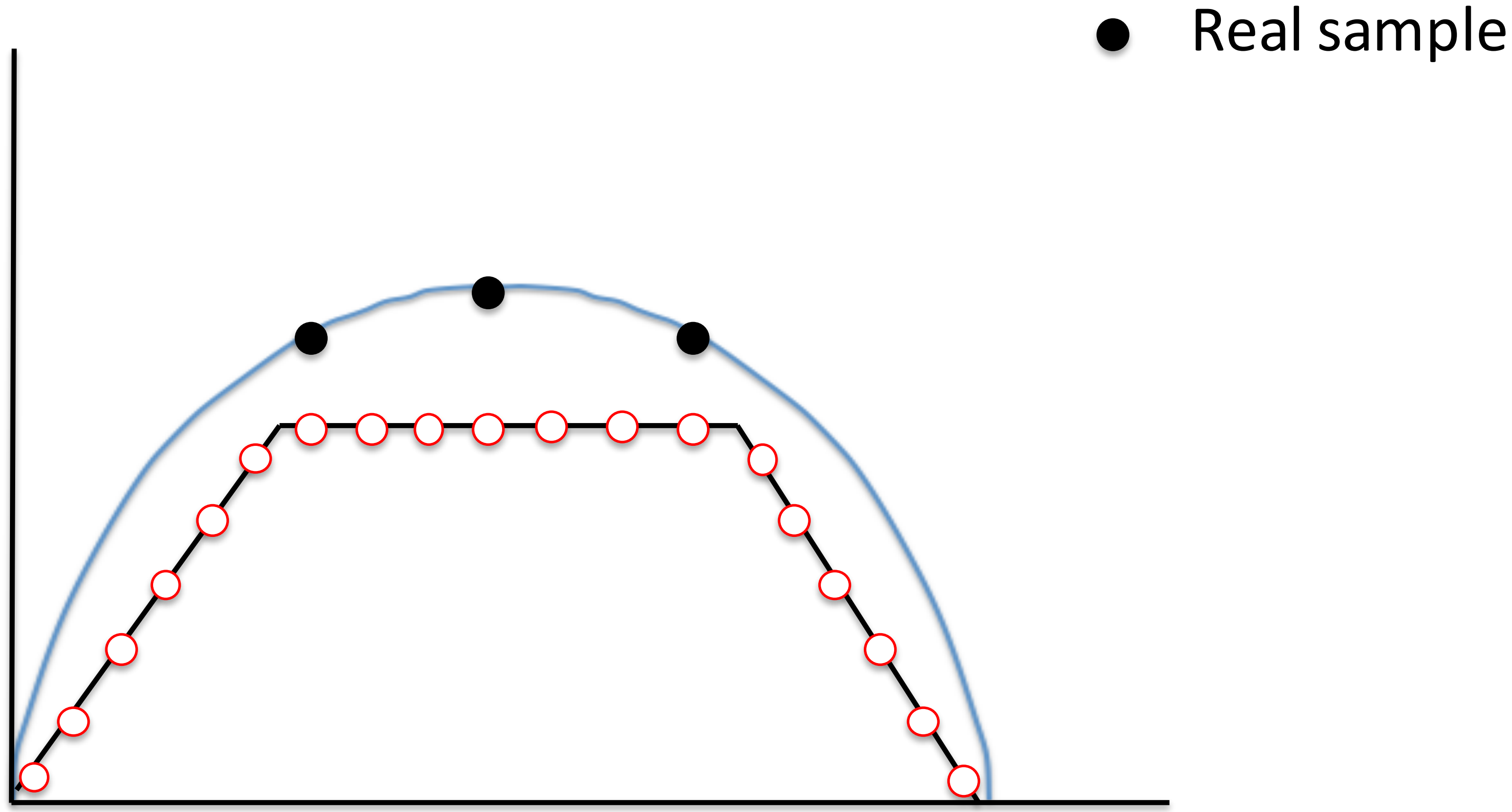
Real vs learnt

- Assuming enough point to perfectly learn AM



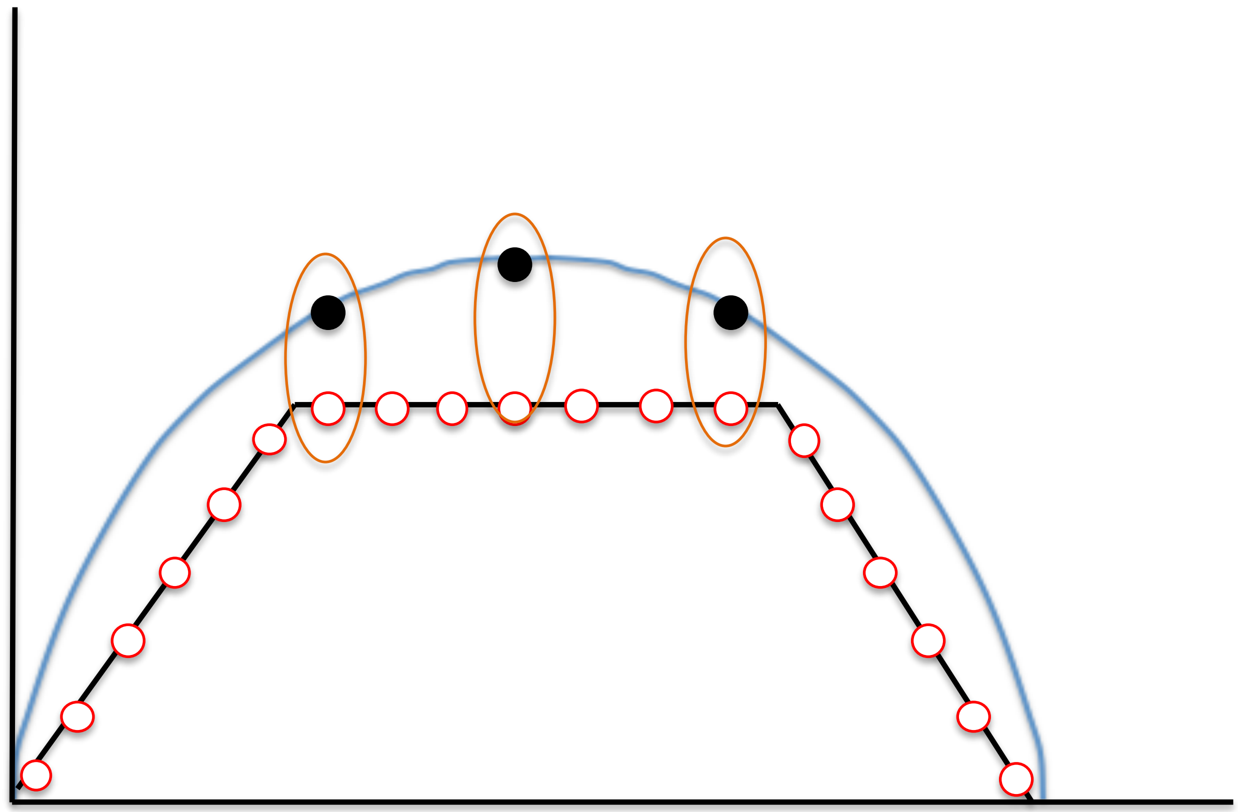
Merge

- Add real samples to synthetic



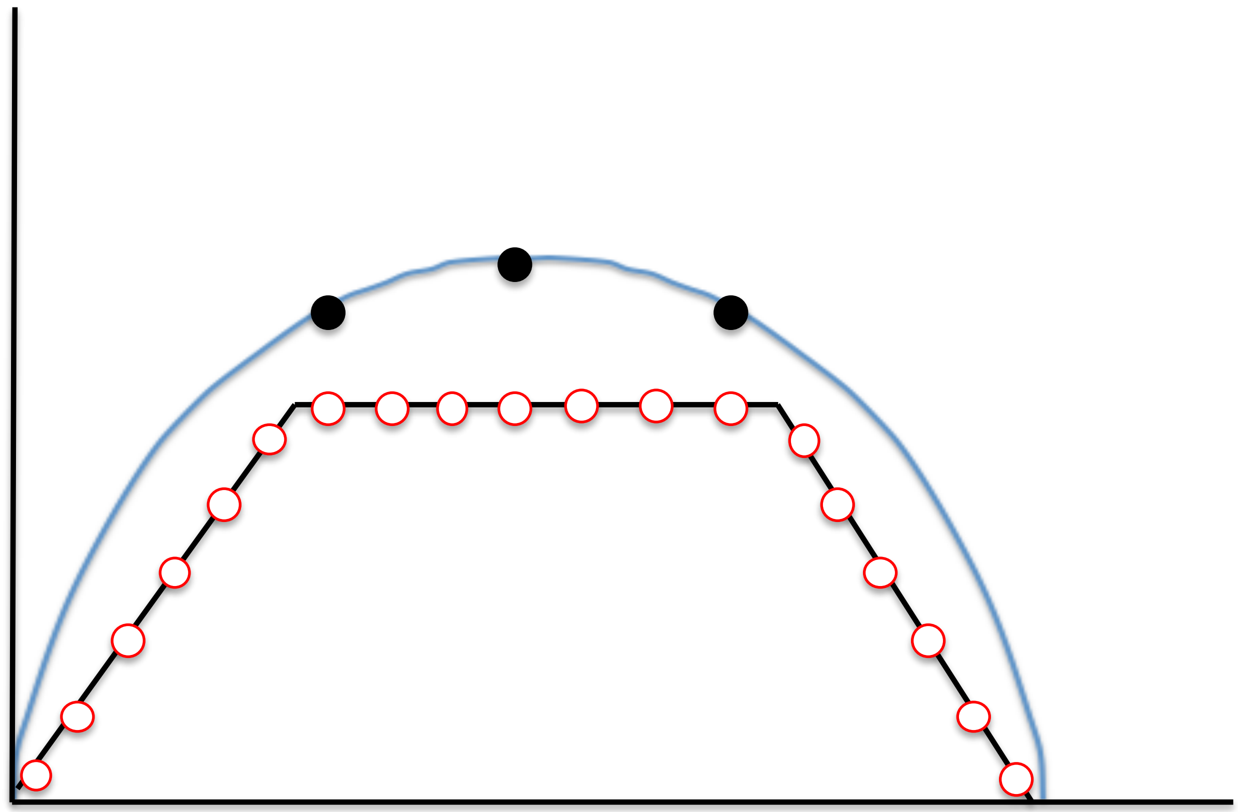
Merge

- Problem: same/near samples have diff. output



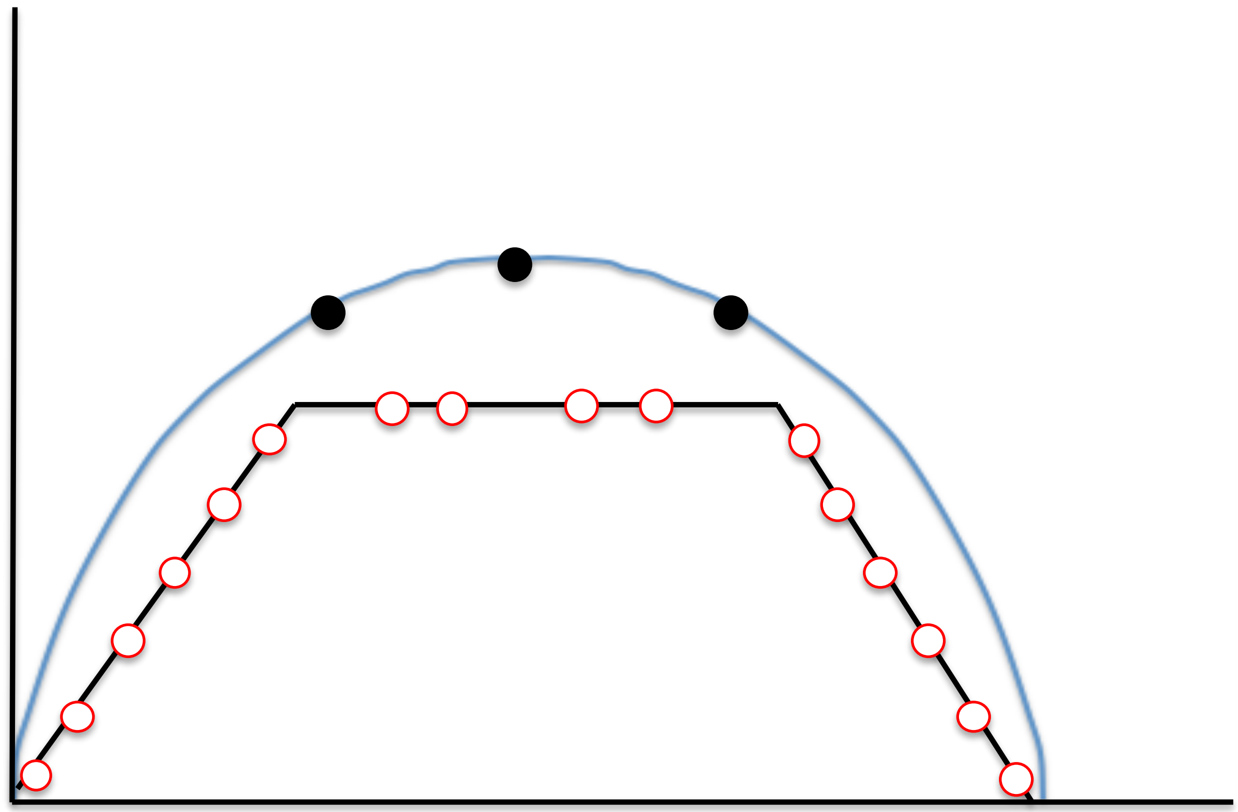
Replace Nearest Neighbor (RNN)

- Remove nearest neighbor



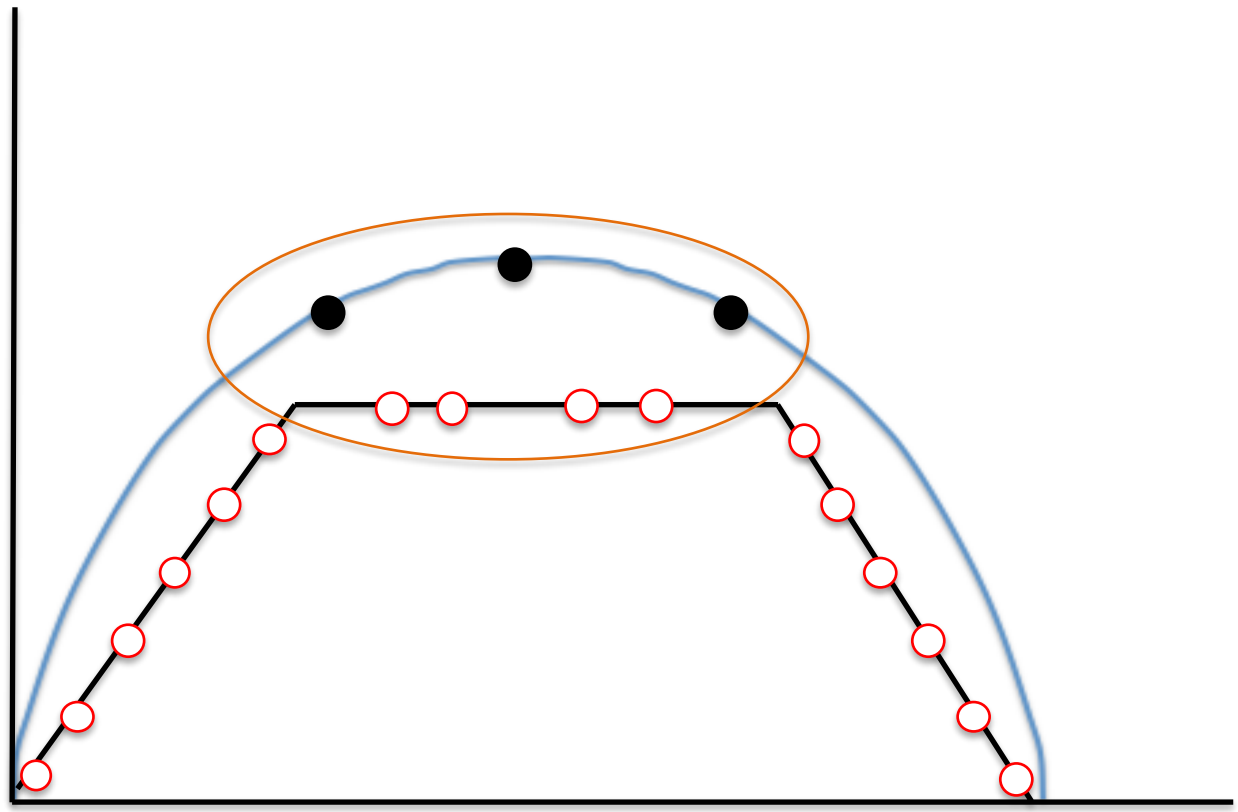
Replace Nearest Neighbor (RNN)

- Preserve distribution…



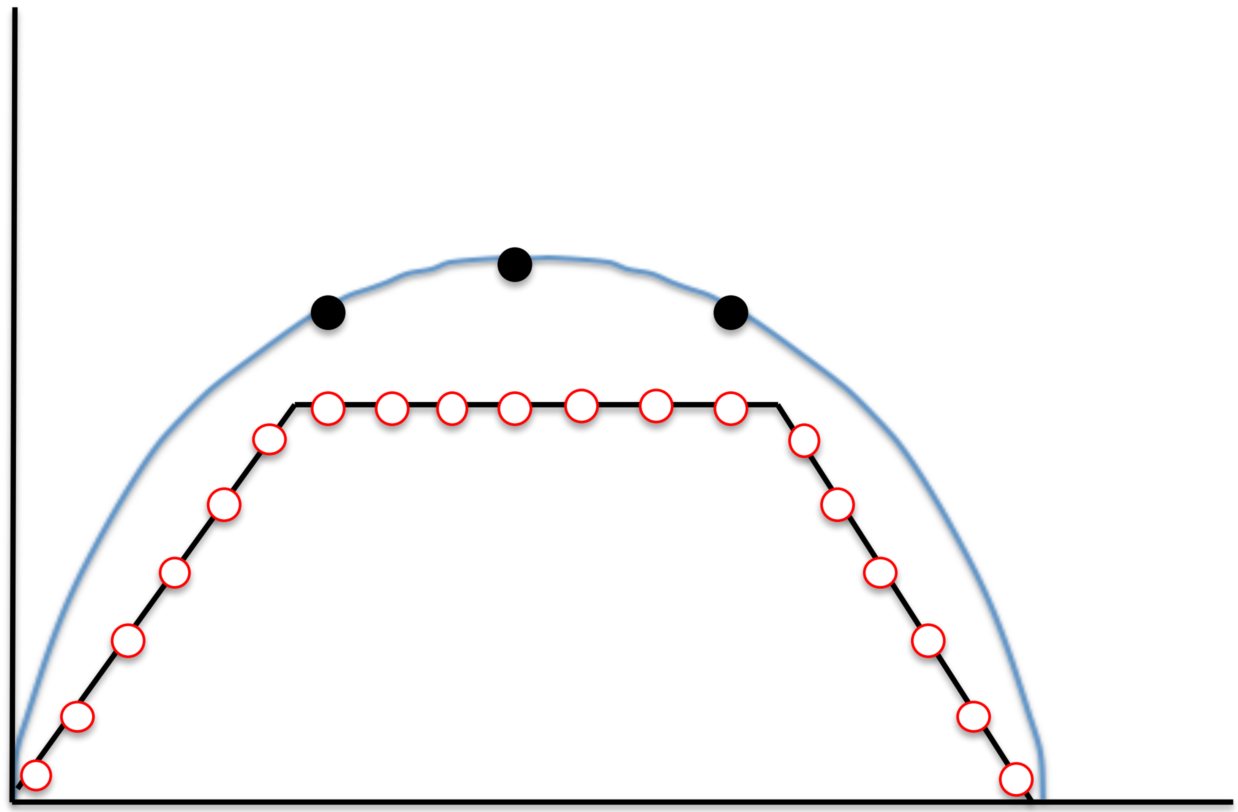
Replace Nearest Neighbor (RNN)

- … but may induce alternating outputs



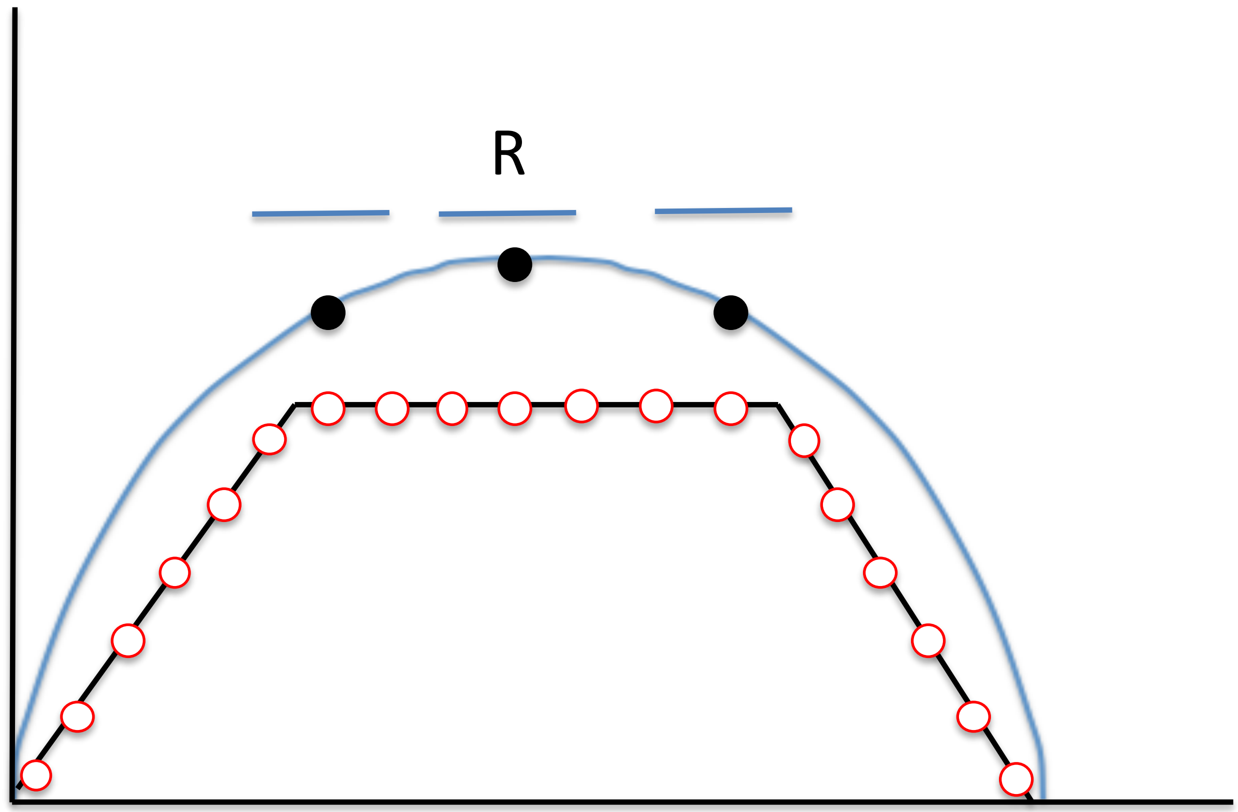
Replace Nearest Region (RNR)

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



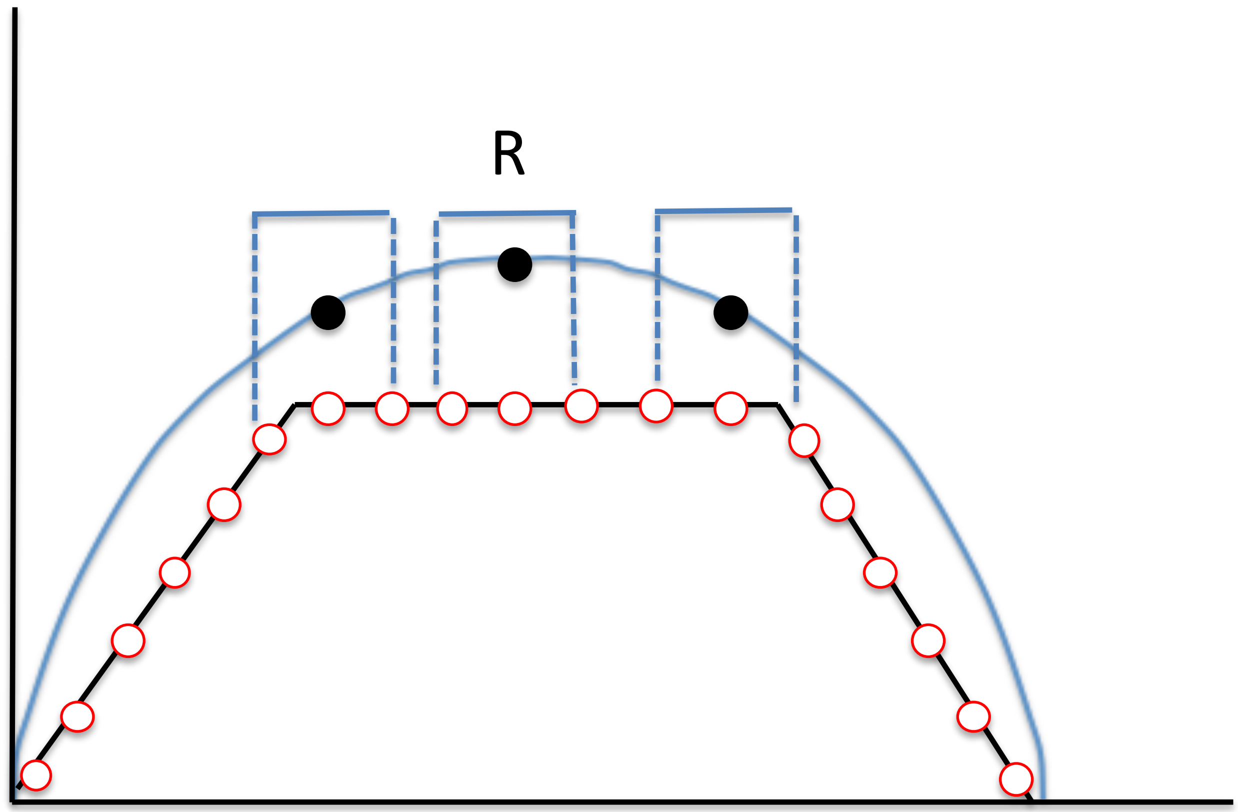
Replace Nearest Region (RNR)

- R = radius defining neighborhood



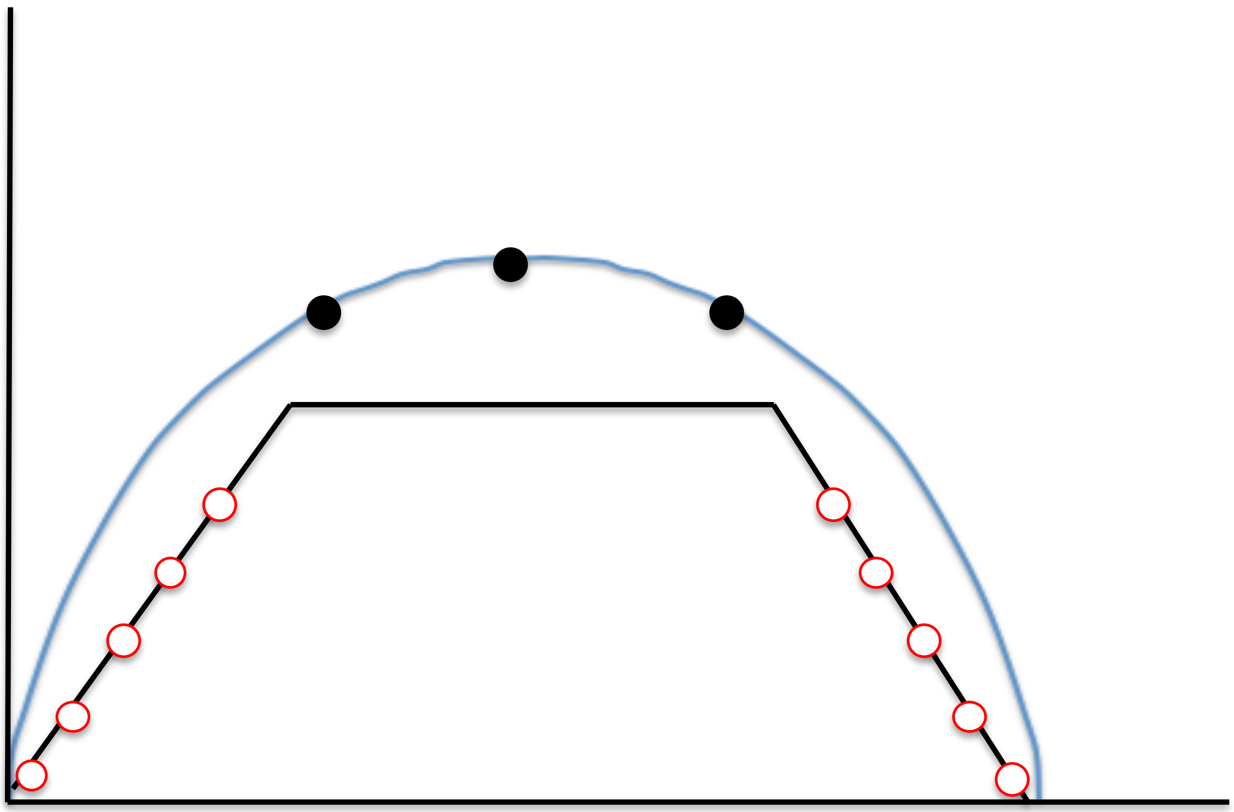
Replace Nearest Region (RNR)

- R = radius defining neighborhood



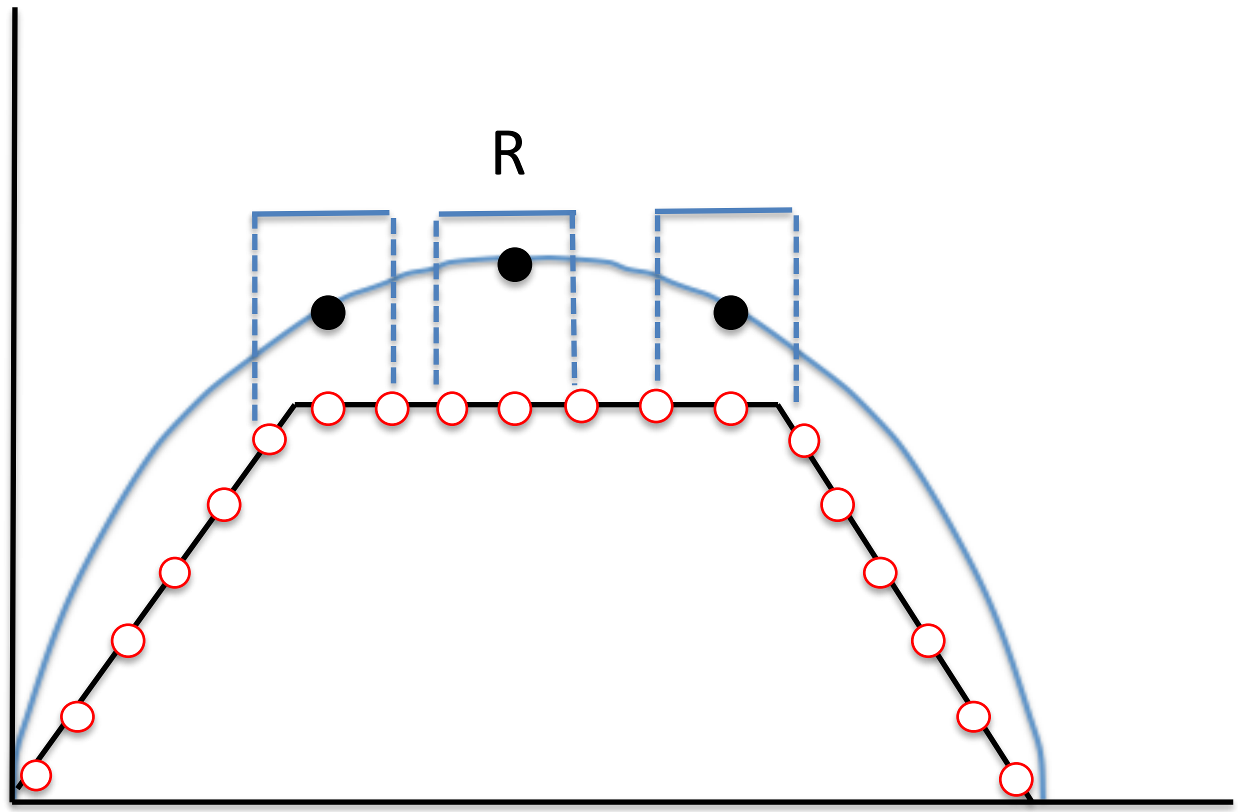
Replace Nearest Region (RNR)

- Skew samples' distribution



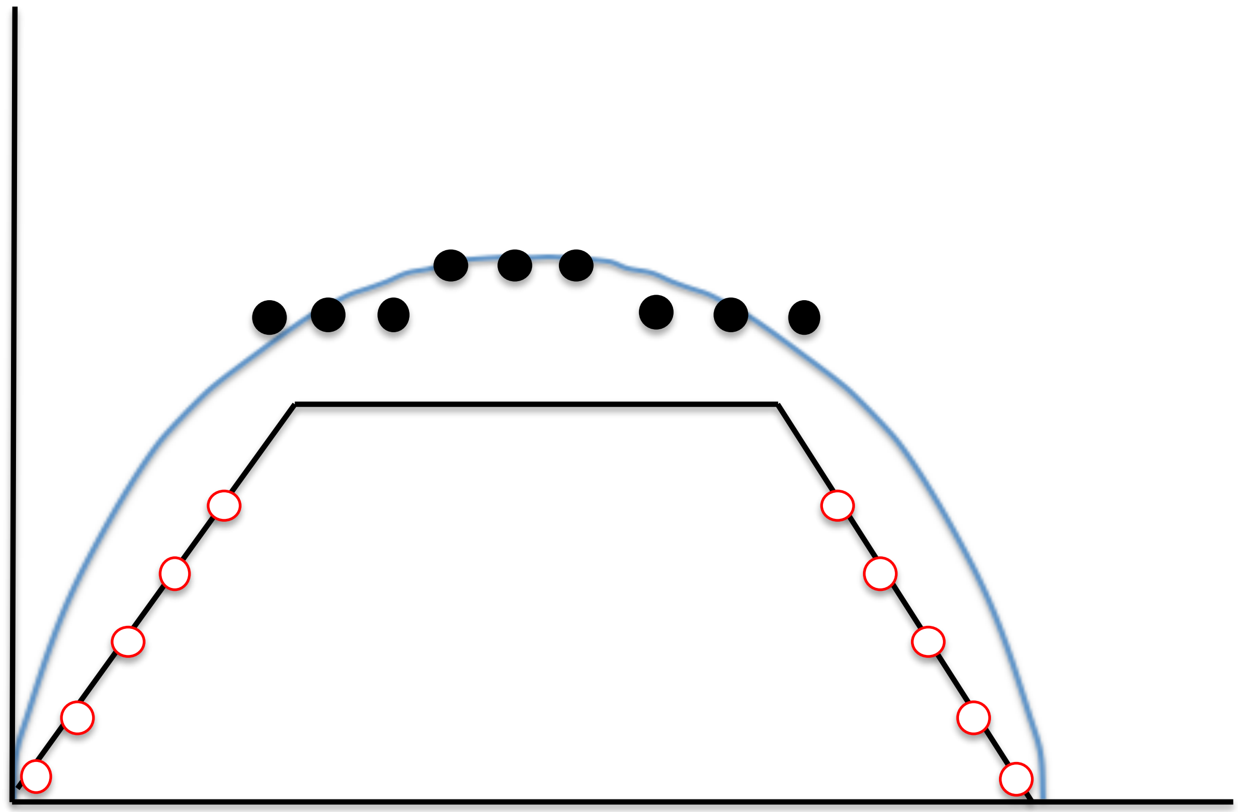
Replace Nearest Region 2 (RNR2)

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



Replace Nearest Region2 (RNN2)

- Maintain distribution, piecewise approximation



Weighting

- Give more relevance to some samples

Fit better the model around **real** samples

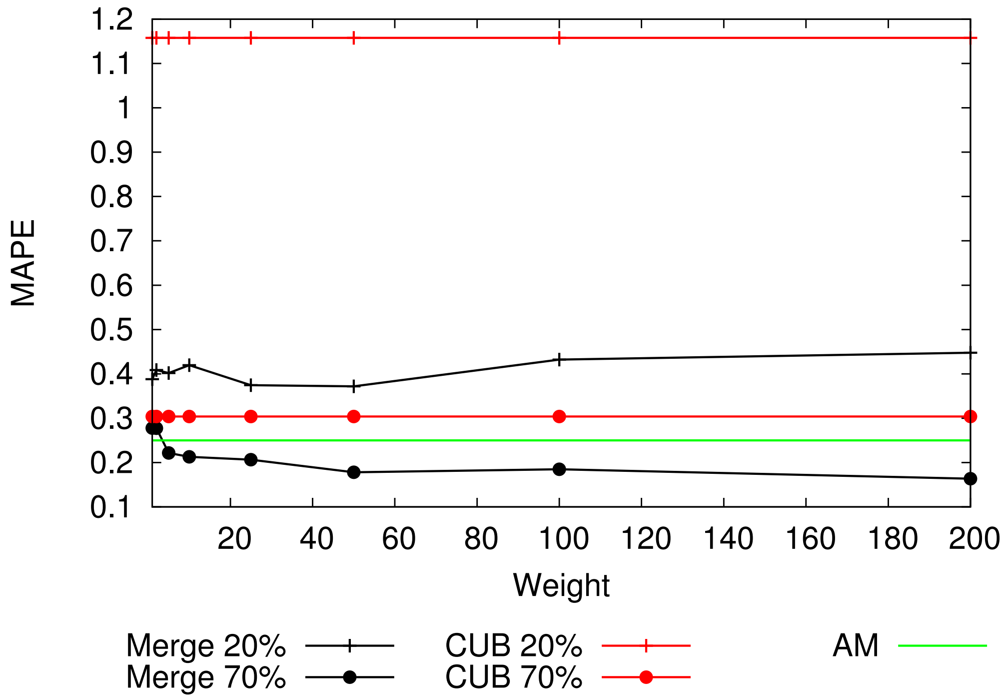
- "Trust" **real** samples more than synthetic ones
- Useful especially in Merge

Too high can cause over-fitting!

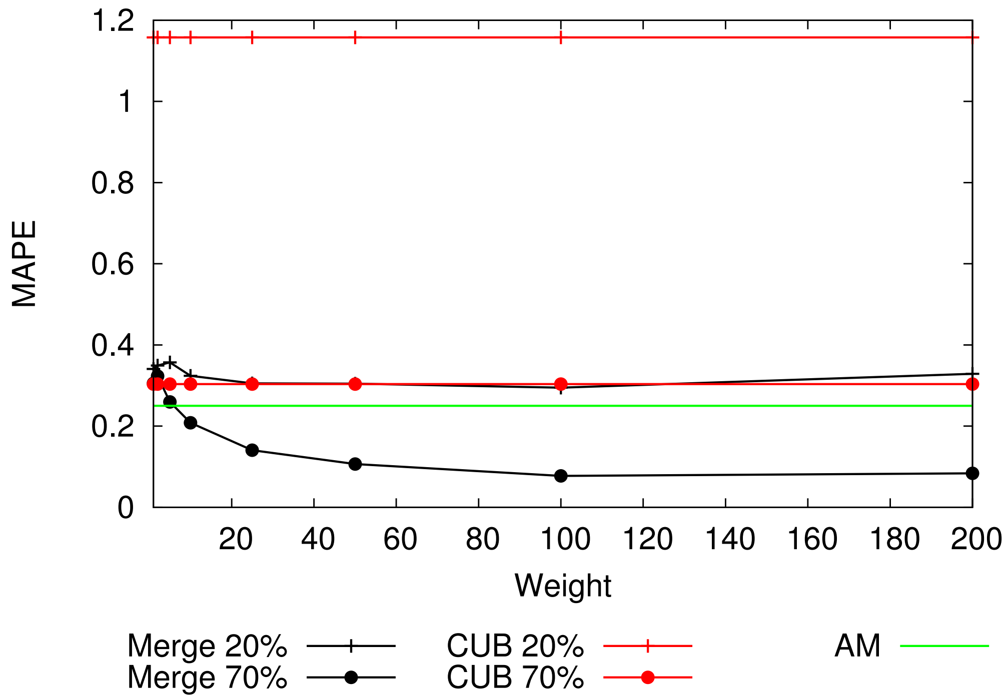
- Learner too specialized only in some regions

Merge , TOB

1 K synthetic samples



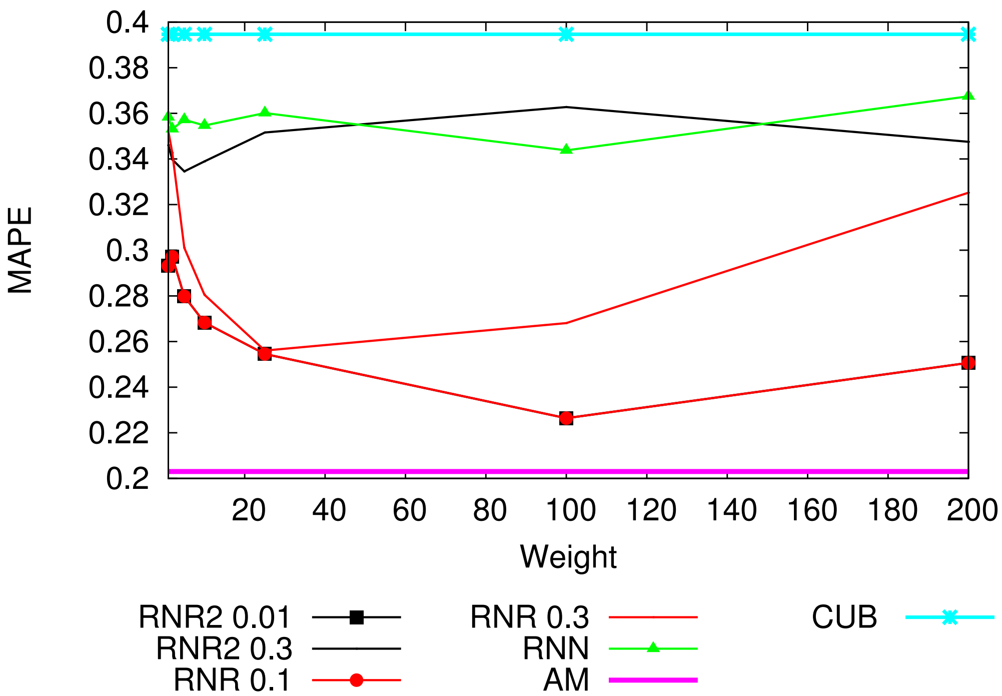
10 K synthetic samples



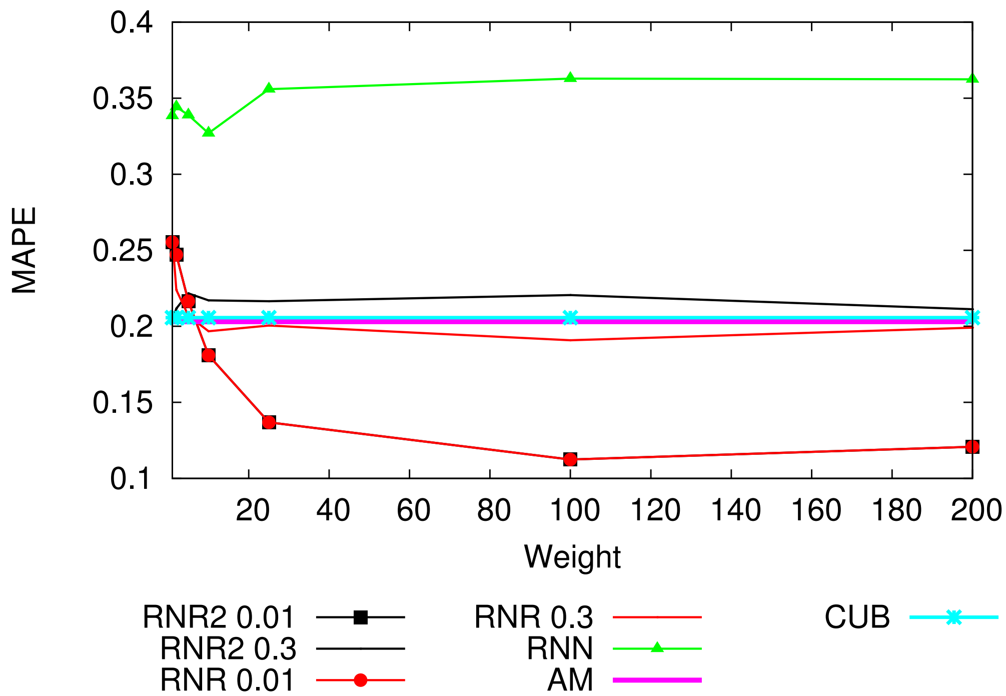
- Weighting more real samples reduces error

Replace, KVS

20%

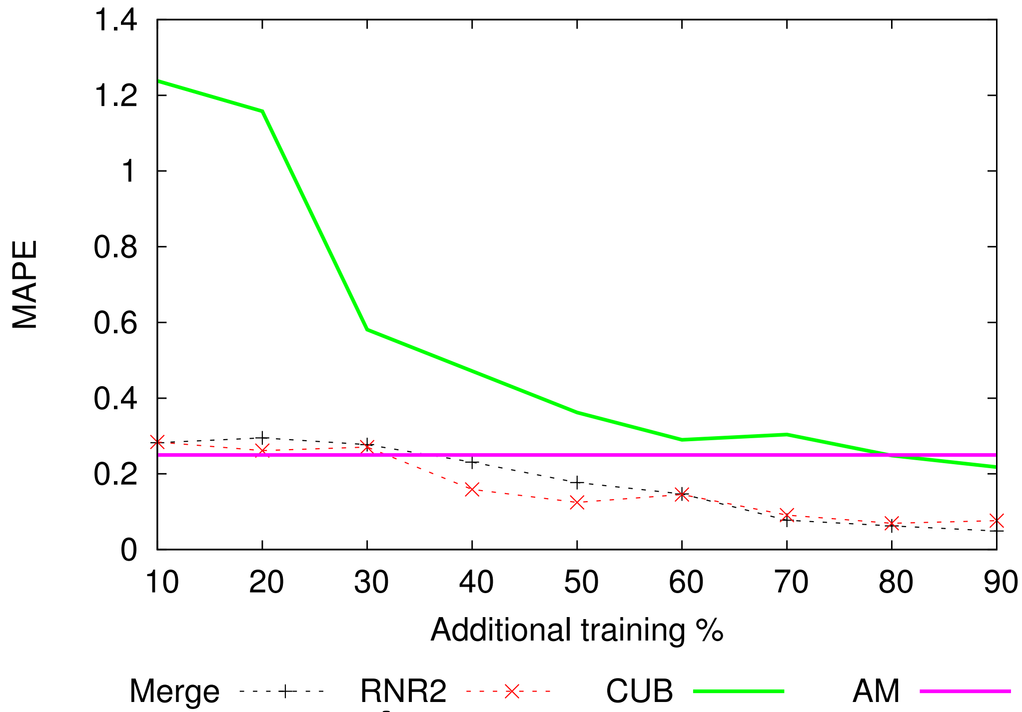


70%



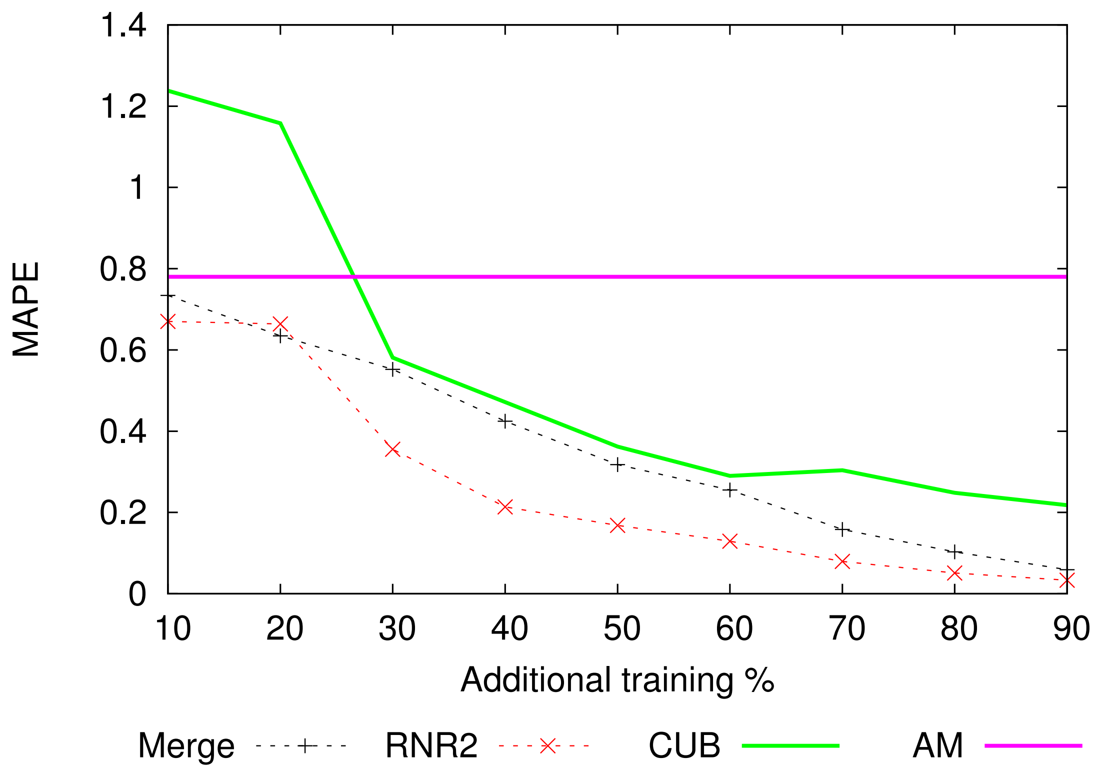
- Examples of over-fitting

MERGE VS REPLACE (TOB)



- Assuming optimal parameterization
- Merge and Replace seem *very* similar…

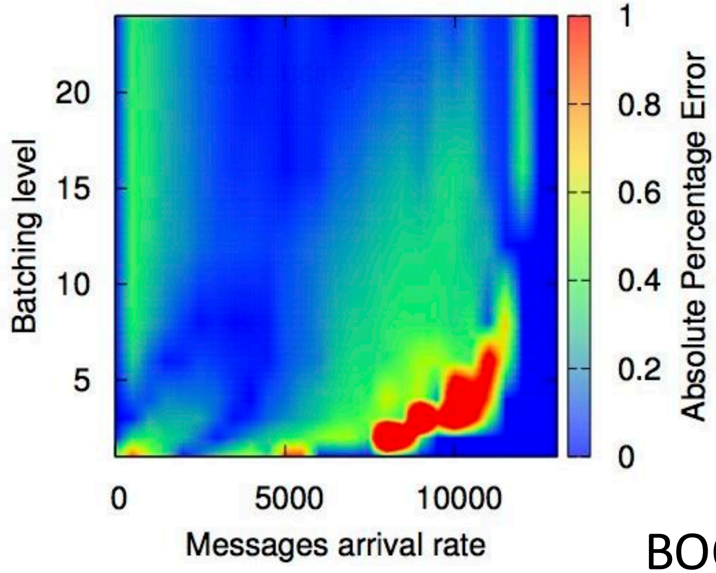
Impact of base model (TOB)



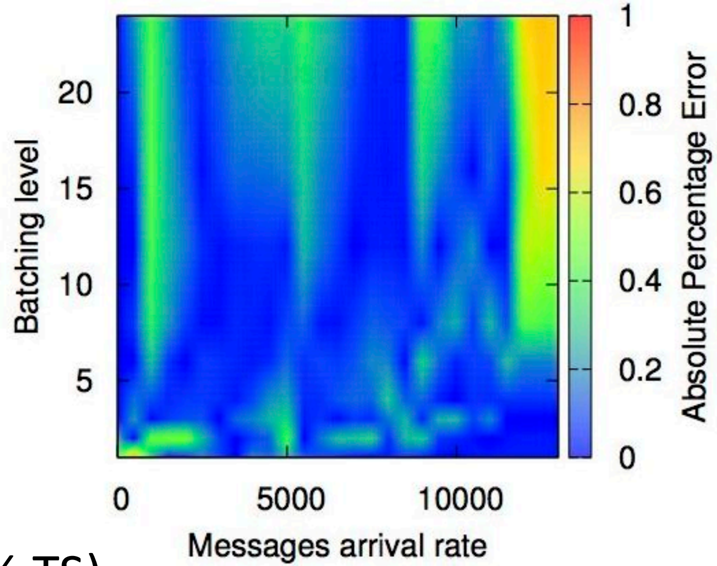
- … BUT replace is better if base model is poor
  - It evicts synthetic samples more aggressively

Visualizing the correction (STOB)

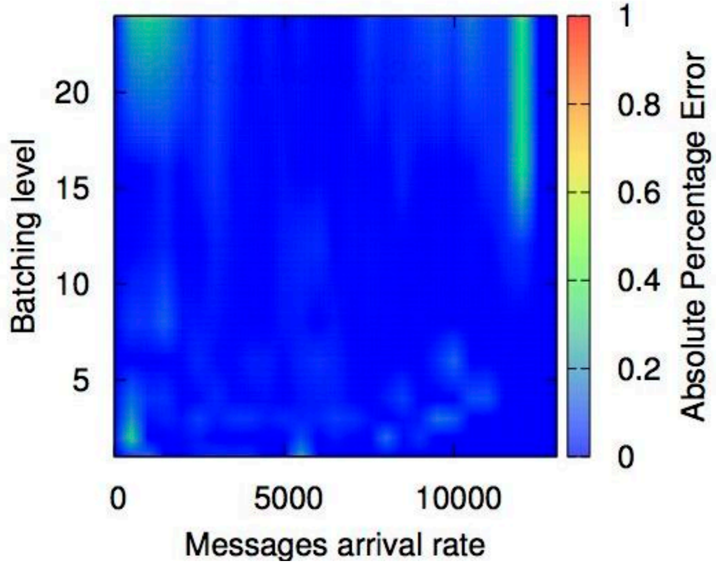
BASE MODEL



PURE ML (70% TS)

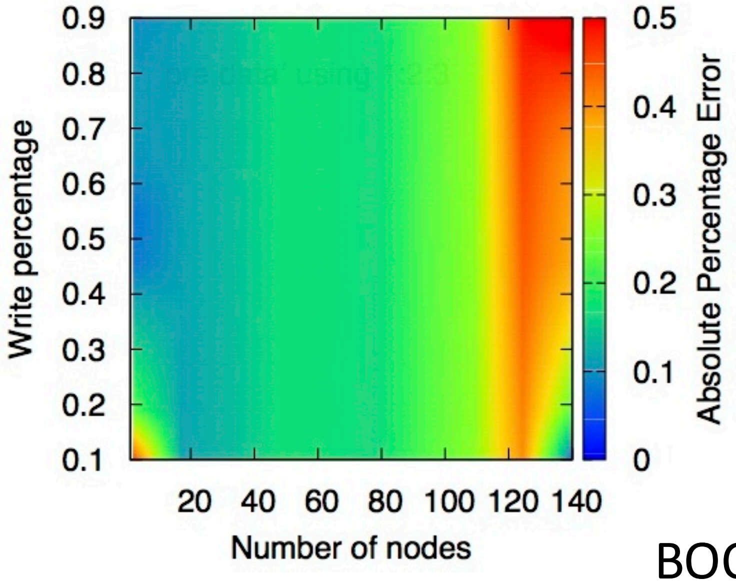


BOOTSTRAPPED ML (70% TS)

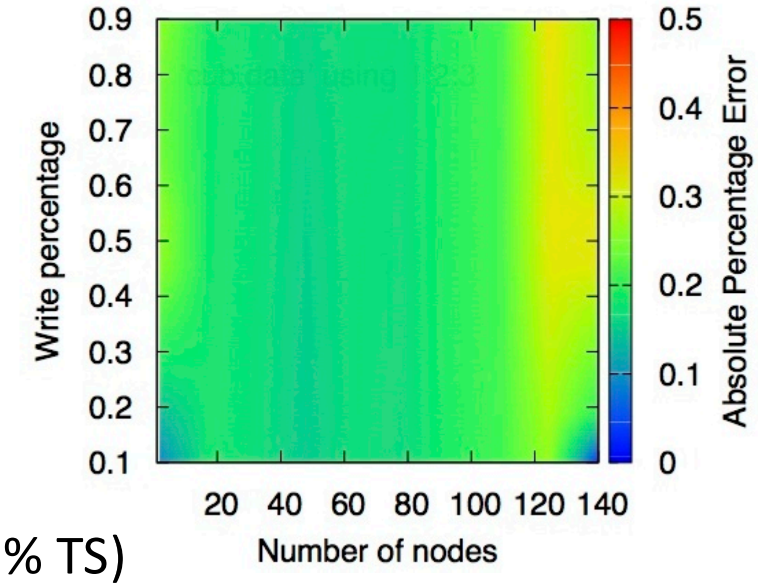


Visualizing the correction (KVS)

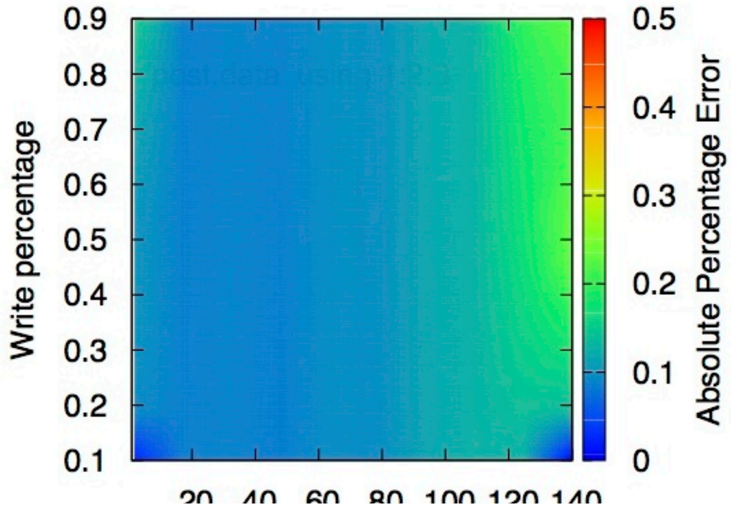
BASE MODEL



PURE ML (70% TS)

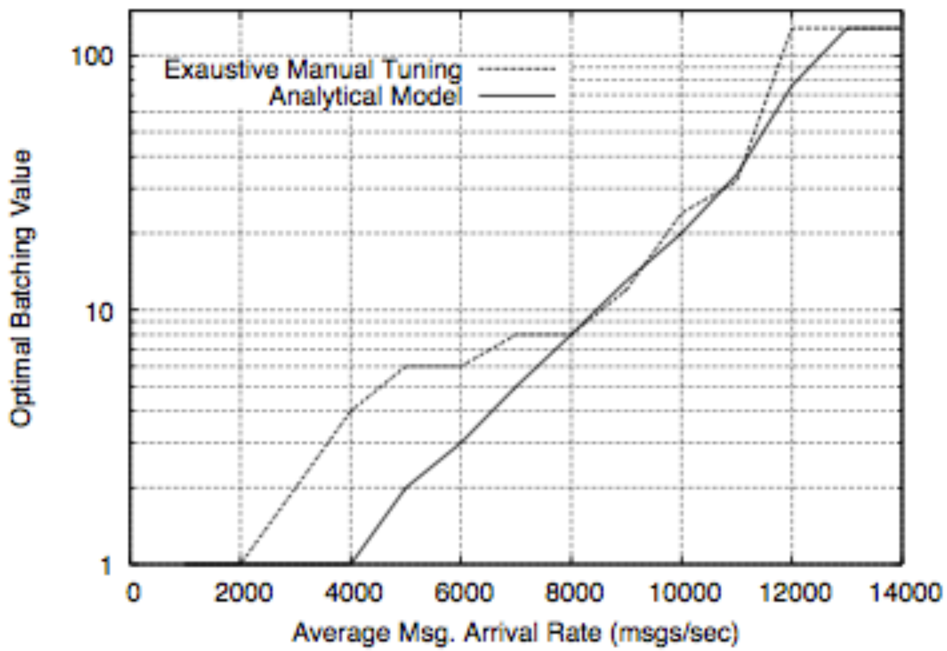


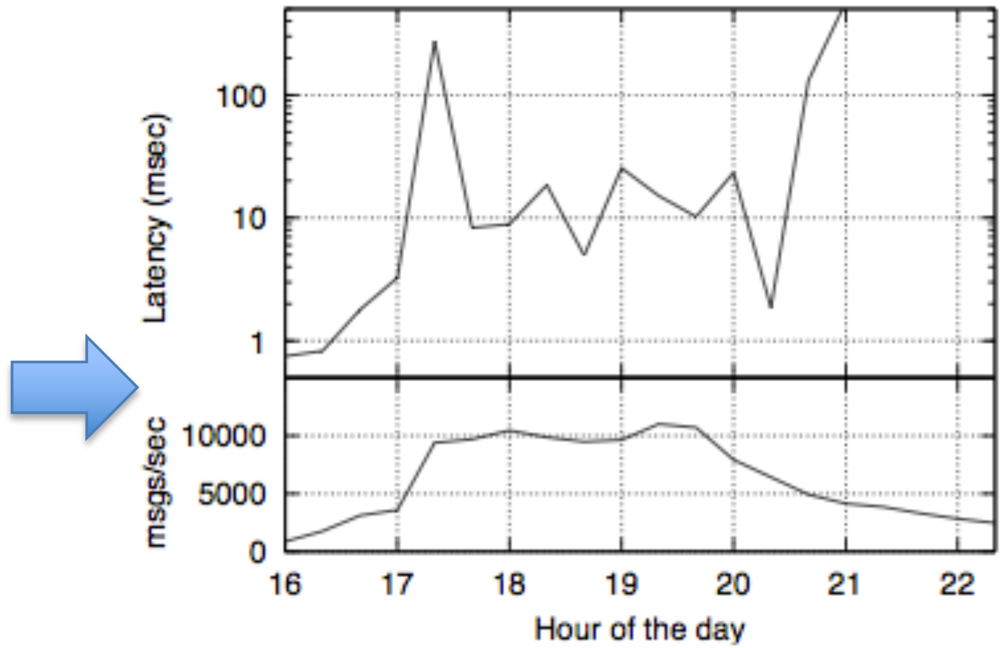
BOOTSTRAPPED ML (70% TS)



BOOTSTRAPPING in RL [59]

- Optimize batching level in STOB
- Base AM already presented





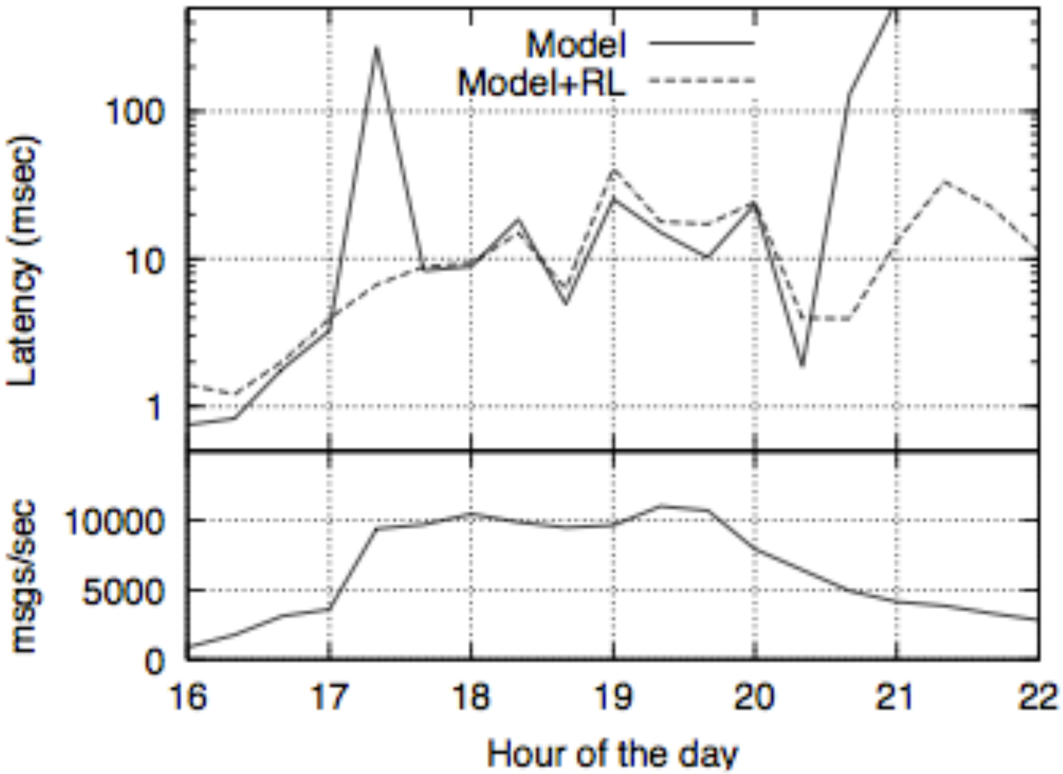
Hybrid RL in STOB

- UCB: find optimal batch size (b*) for a given msg. arrival rate (m)
  - Discretize m domain into M={m_min…m_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

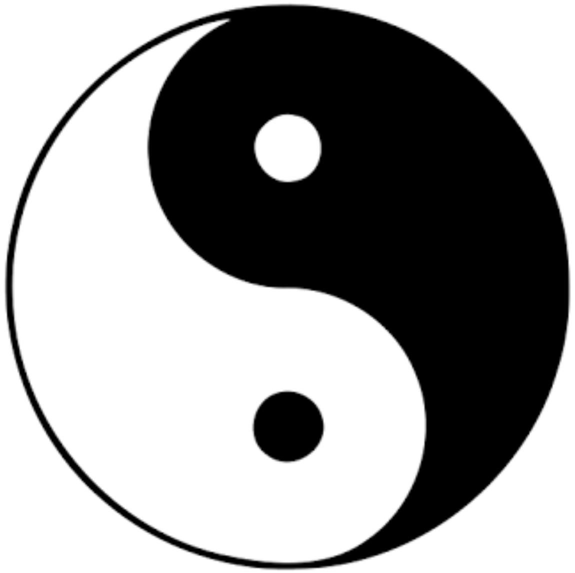
Bootstrapped model

- 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



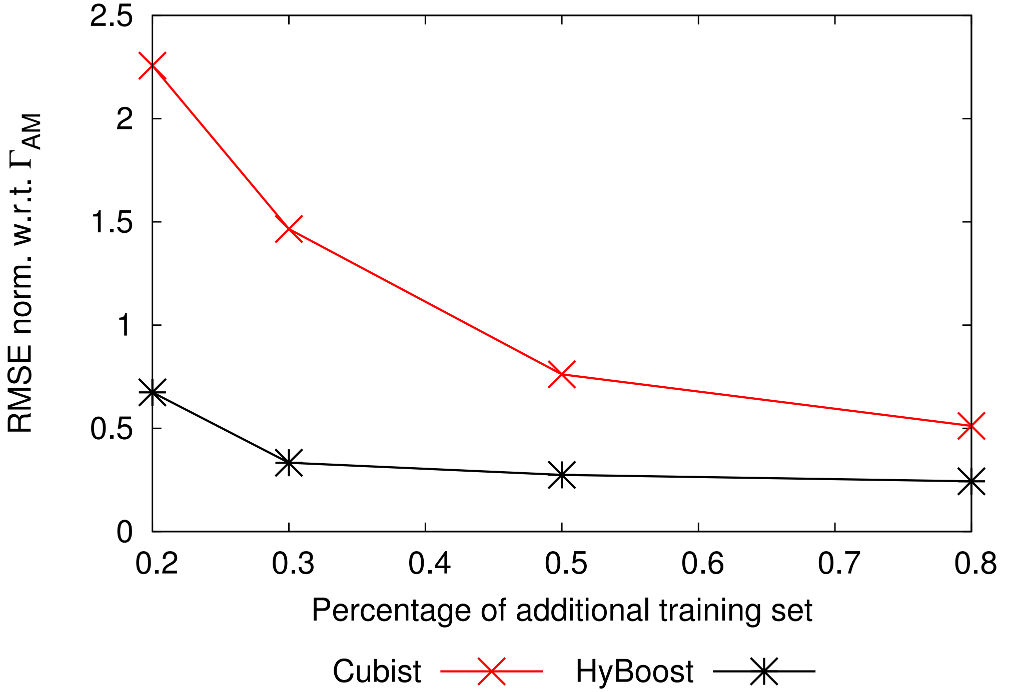
Hybrid Ensemble [26]

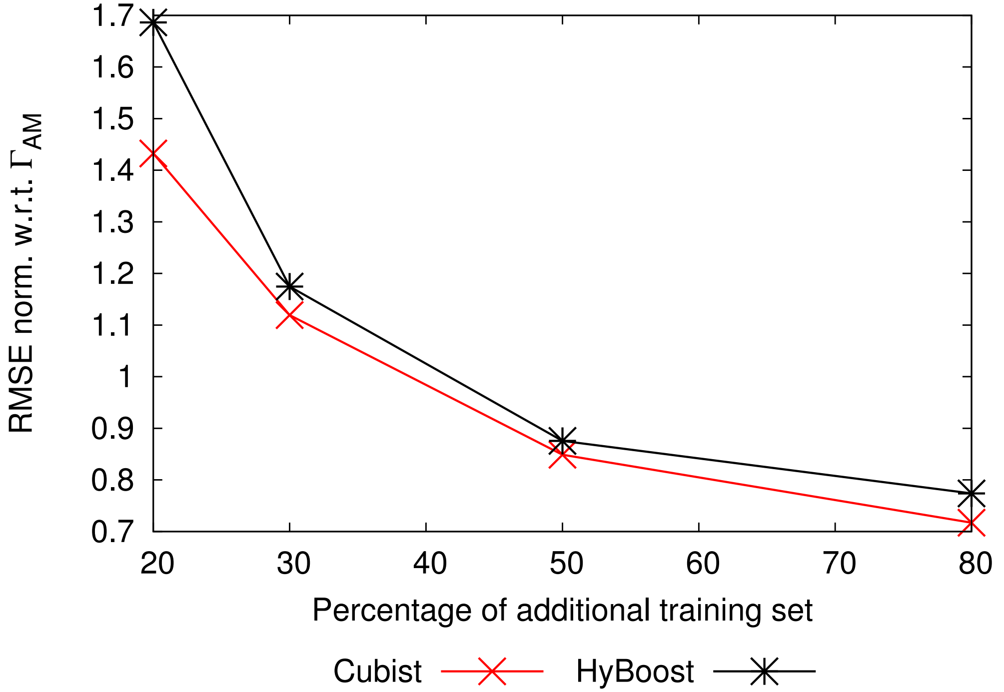
- Combine output of AM and ML
  - Hybrid boosting: correct errors of single models
  - KNN: select best model depending on query
  - Probing: train ML only where AM is not accurate

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}

BOOSTING: sensitivity





- 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

Implemented solution: SC-TAS

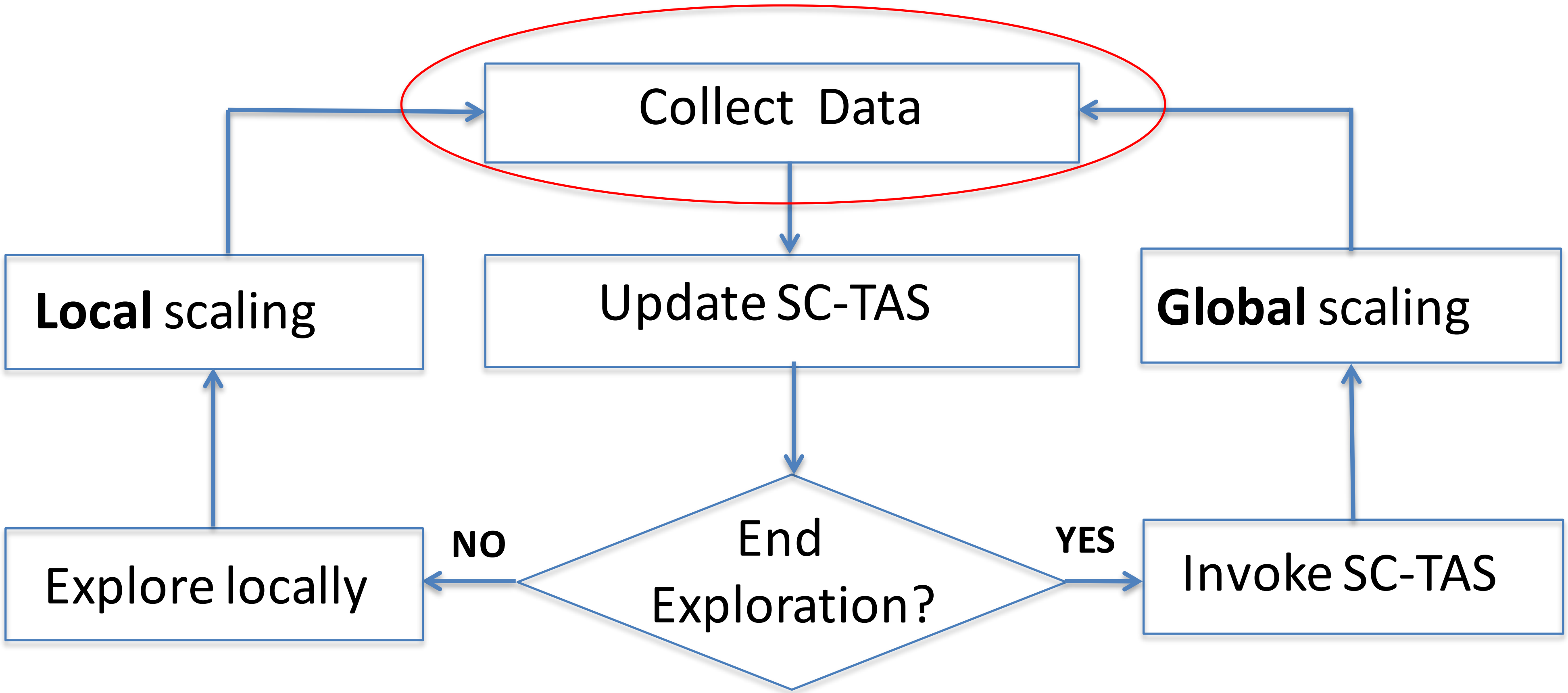
Exploration + modeling + Machine Learning

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)

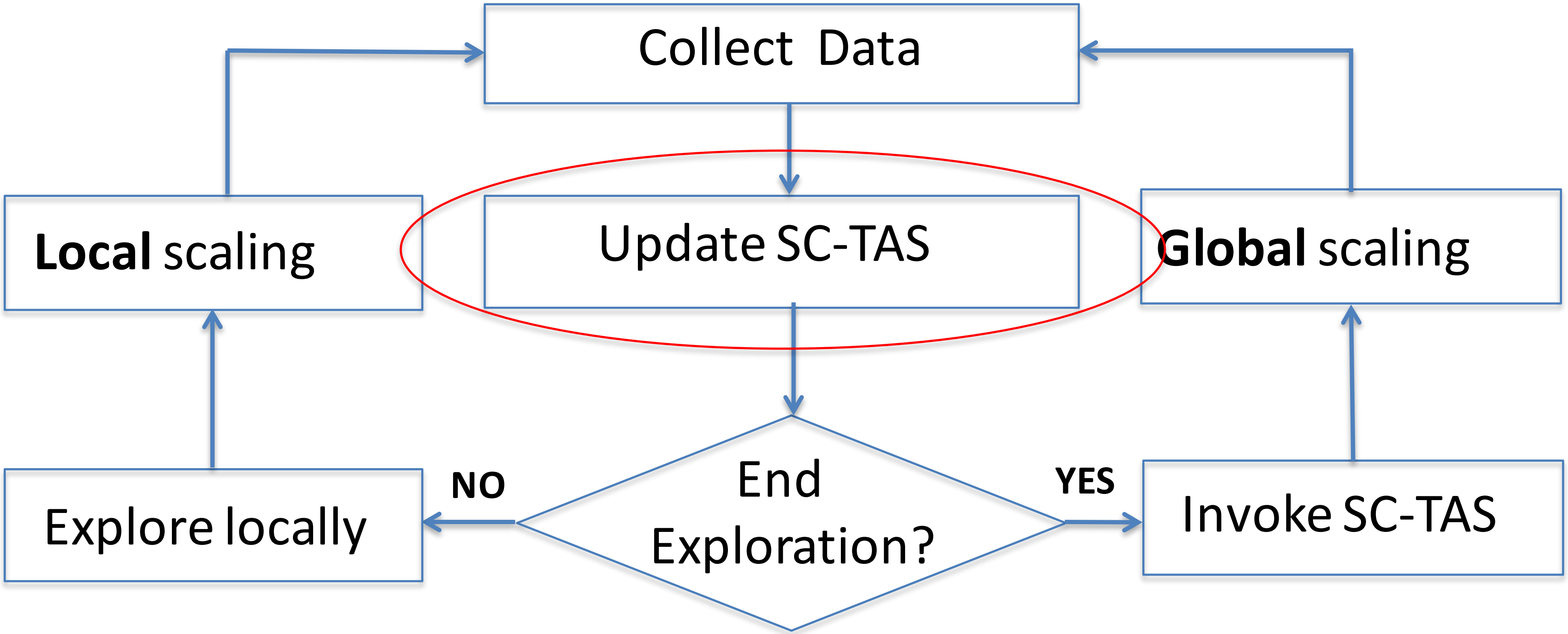
Increase accuracy

SC-TAS control loop



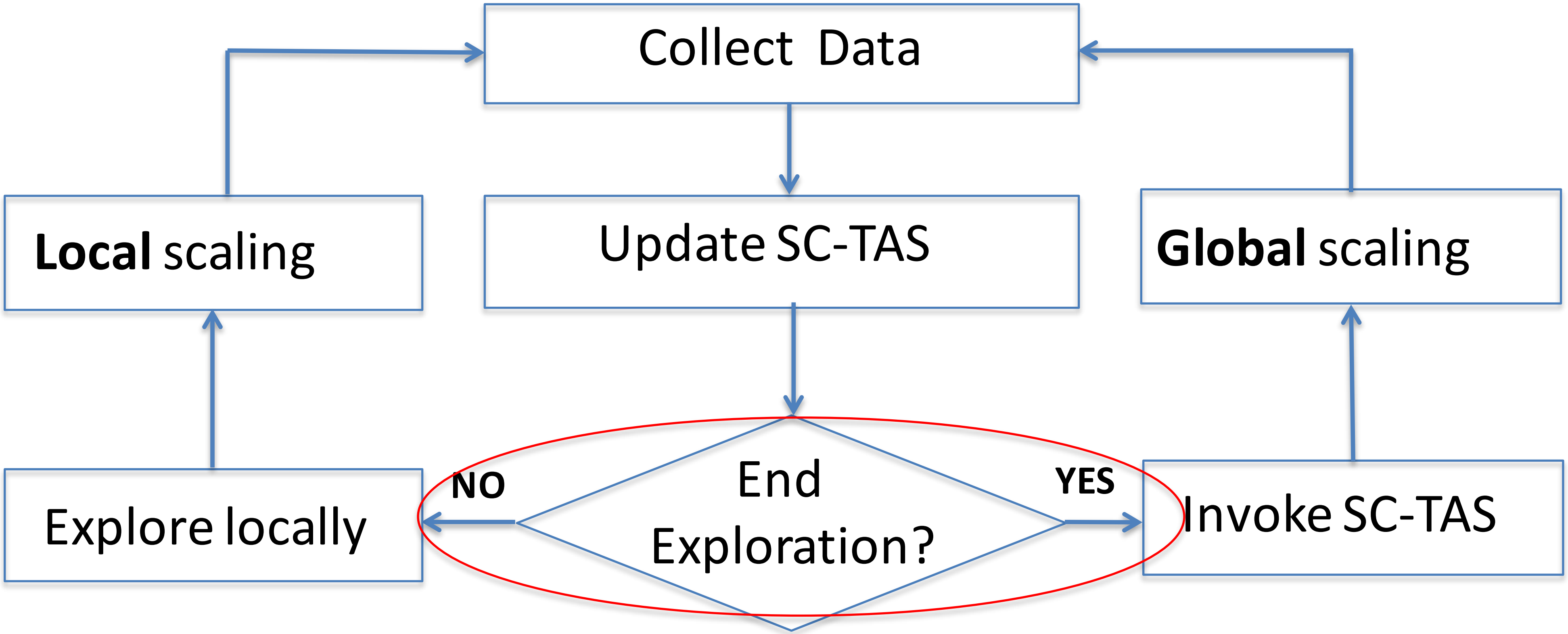
- Workload, #thread, #nodes, TAS' error

SC-TAS control loop



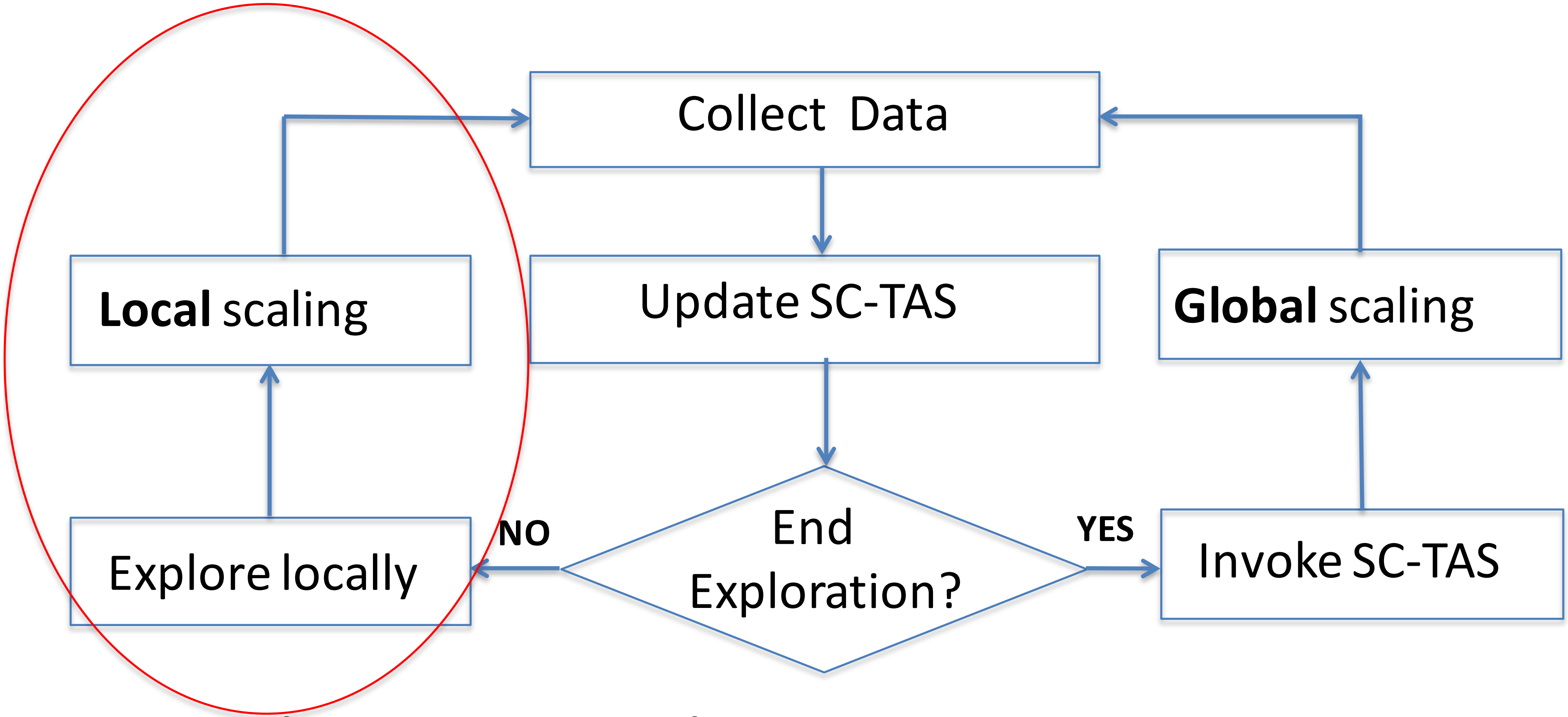
- Re-train hyboost “patching” ML

SC-TAS control loop



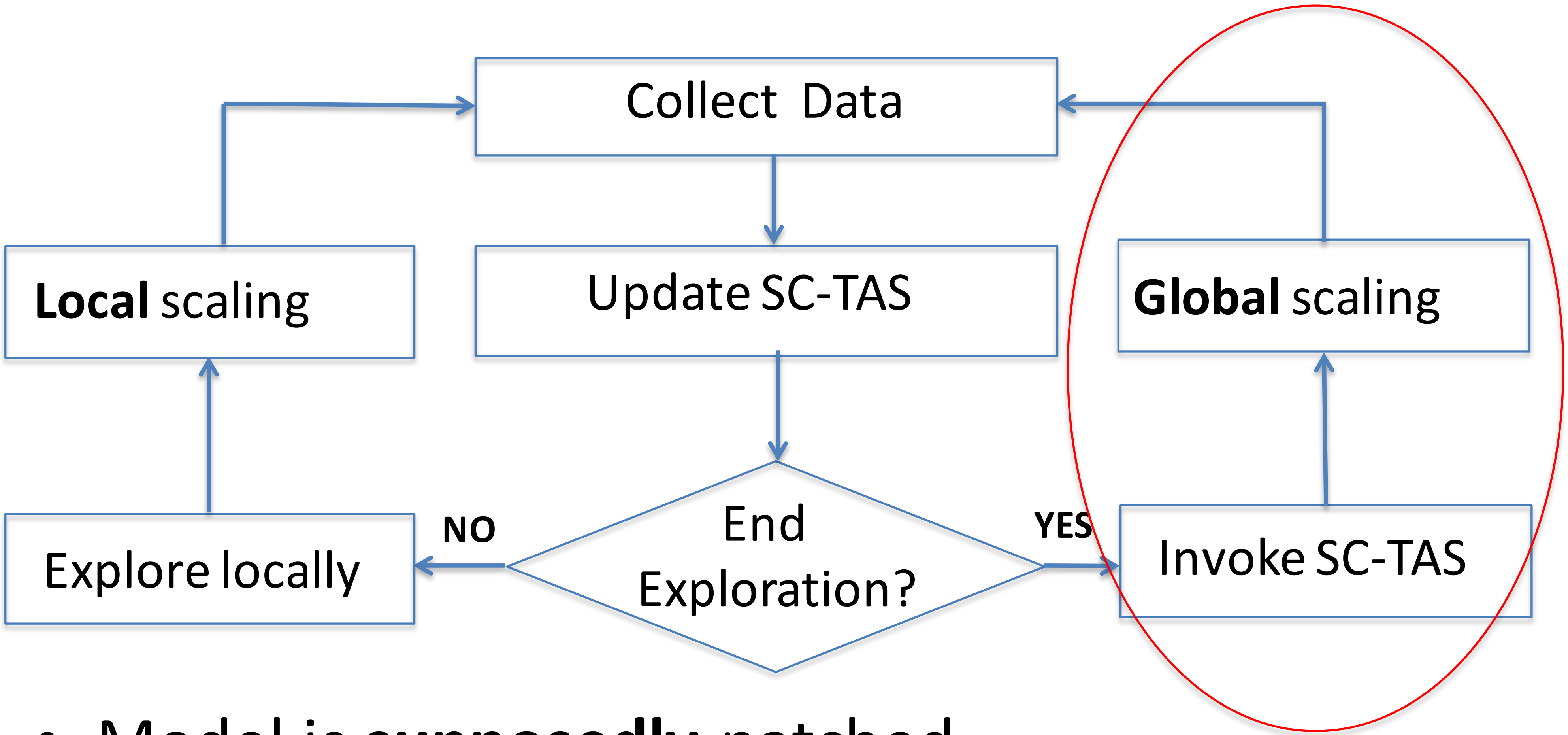
- Yes if min<#thread exploration<max &&
- Accuracy of the patched model considered “OK”

SC-TAS control loop



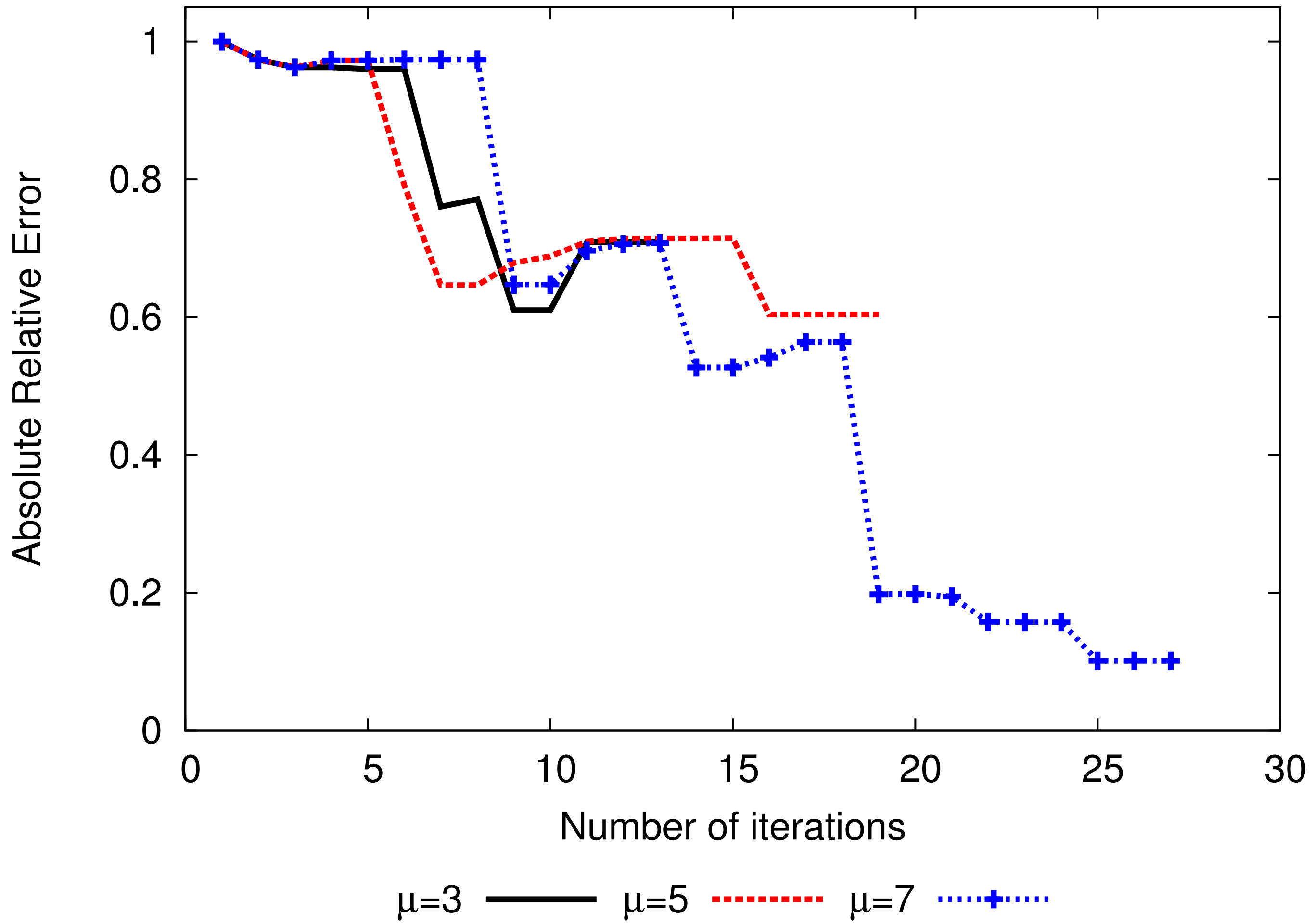
- Patch is not enough
- Change # of threads and repeat

SC-TAS control loop



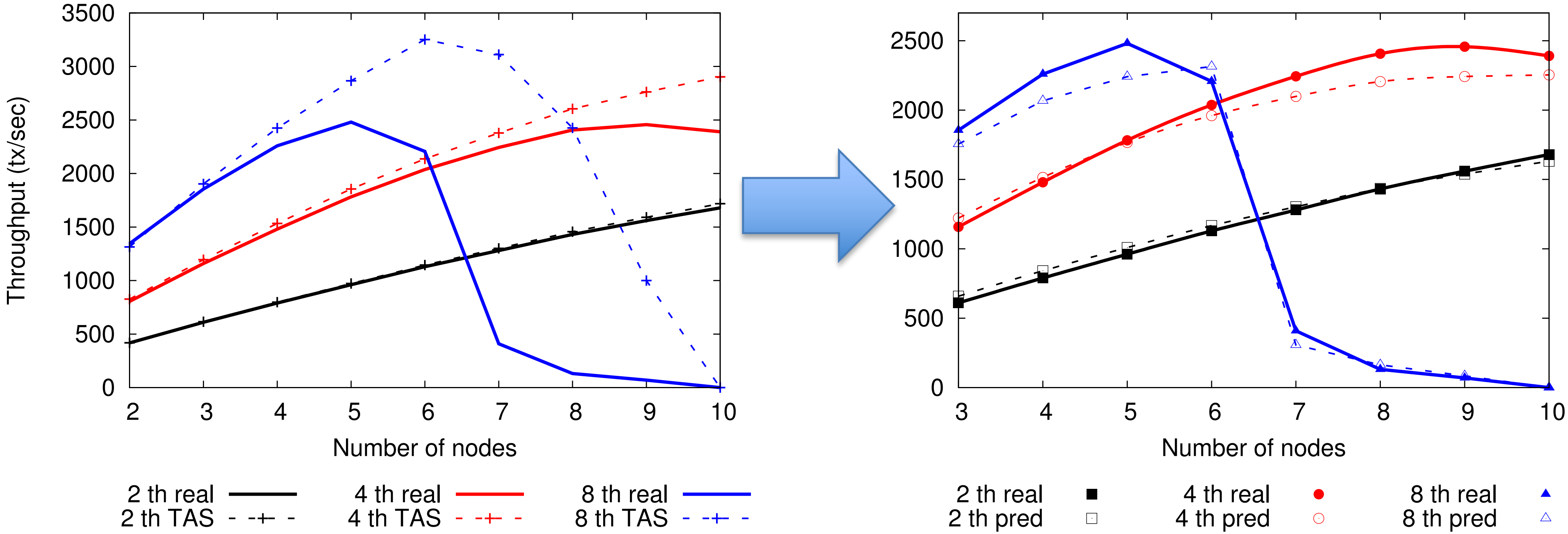
- Model is **supposedly** patched
- Invoke the patched model

Dynamics of SC-TAS



- μ = min # of thread explorations per node

SC-TAS: before and after



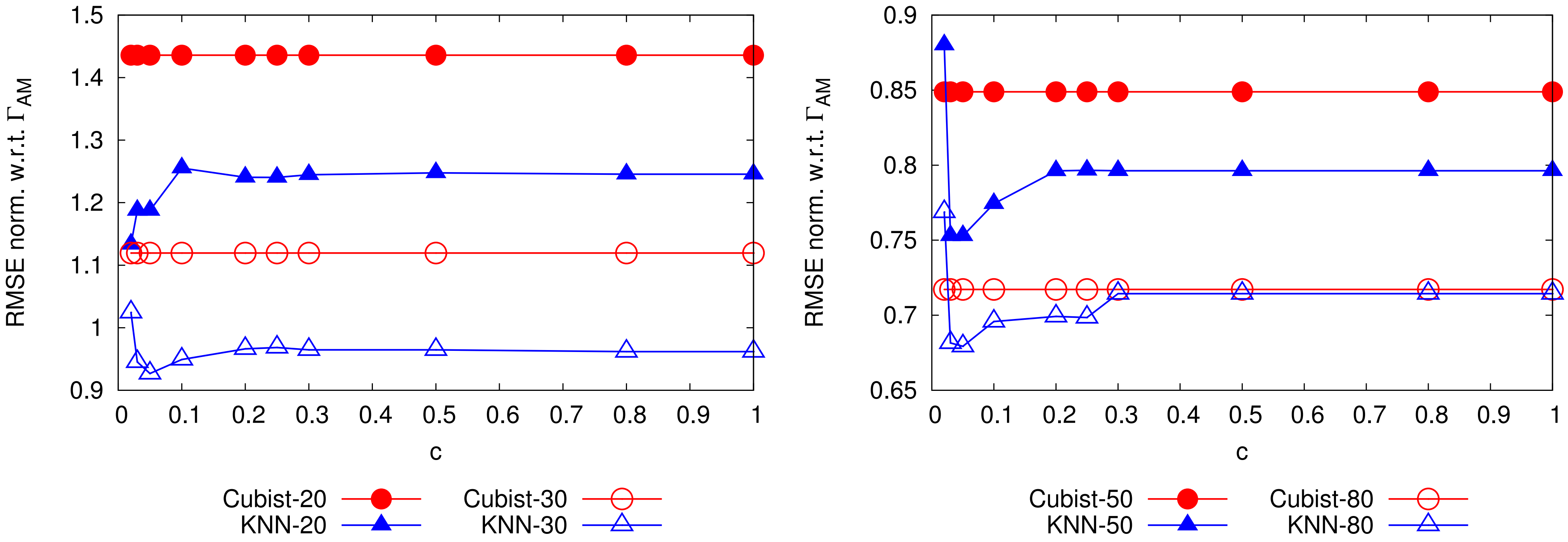
Hybrid Ensemble [26]

- 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 $ML_1 \ldots ML_N$ 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 → collapse to AM
- High cut-off && high % training → 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

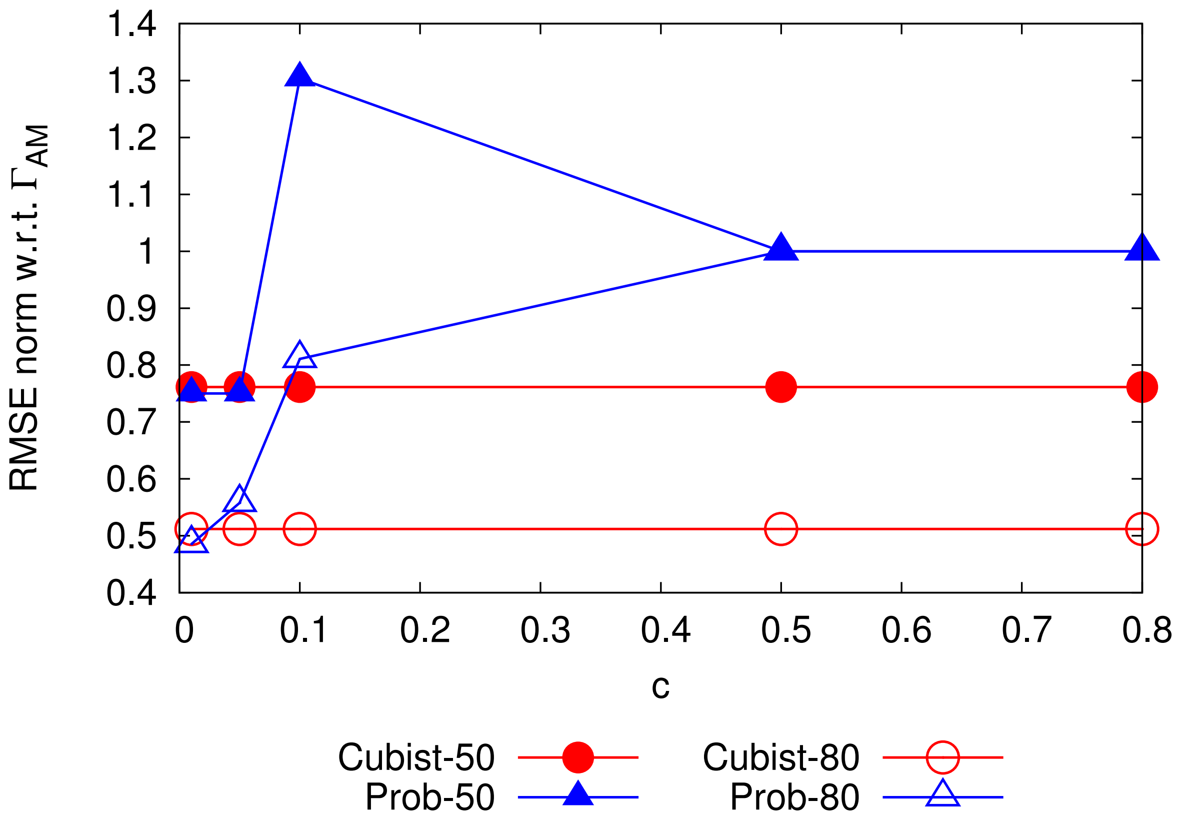
- Differences with KNN
  - In KNN, ML is trained on all samples:
    - Here, only when AM is found to be inaccurate
  - In KNN, voting decides on ML vs AM:
    - Here, binary classifier determines in <u>which regions</u> the AM is inaccurate

Probing at work

1. D_{ML} = empty set
2. Train a classifier: for each x in D
   - If error of AM on x > cut-off, map x to ML and add x to D_{ML}
   - Else map x to AM
3. Train ML on D_{ML}

- QUERY for input z
  - If classify(z) = AM, return AM(z); else return ML(z)

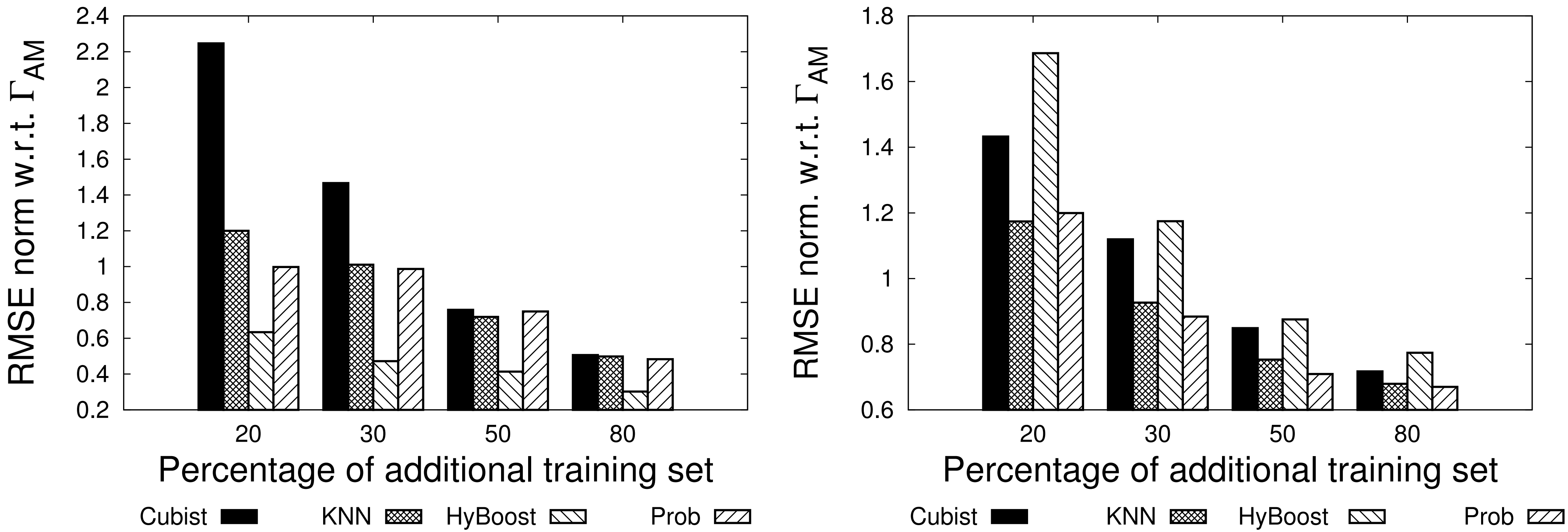
Probing Sensitivity (KVS)



- High cut-off → collapses to AM
- Low cut-off → 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

[1] Mert Akdere, Ugur Çetintemel, Matteo Riondato, Eli Upfal, and Stanley B. Zdonik. Learning-based query performance modeling and prediction. In *Proceedings of the 2012 IEEE 28th International Conference on Data Engineering*, ICDE '12, pages 390–401, Washington, DC, USA, 2012. IEEE Computer Society.

[2] Naomi S Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.

[3] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 2002.

[4] Forest Baskett, K. Mani Chandy, Richard R. Muntz, and Fernando G. Palacios. Open, closed, and mixed networks of queues with different classes of customers. *J. ACM*, 22(2):248–260, April 1975.

[5] Richard Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957.

[6] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, 13(1):281–305, February 2012.

[7] Philip A Bernstein and Nathan Goodman. Concurrency control in distributed database systems. *ACM Computing Surveys (CSUR)*, 13(2):185–221, 1981.

[8] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., 2006.

[9] Leo Breiman. Bagging predictors. *Mach. Learn.*, 24(2):123–140, August 1996.

[10] Leo Breiman. Stacked regressions. *Machine learning*, 24(1):49–64, 1996.

[11] Christian Cachin, Rachid Guerraoui, and Luís Rodrigues. *Introduction to Reliable and Secure Distributed Programming (2. ed.)*. Springer, 2011.

[12] Rich Caruana , Alexandru Niculescu-Mizil , Geoff Crew , Alex Ksikes Ensemble selection from libraries of models. In *Proc. of ICML*, 2004.

[13] Jin Chen, G. Soundararajan, and C. Amza. Autonomic provisioning of backend databases in dynamic content web servers. In *Proceedings of the 2006 IEEE International Conference on Autonomic Computing*, ICAC '06, pages 231–242, Washington, DC, USA, 2006. IEEE Computer Society.

[14] Jin Chen, G. Soundararajan, S. Ghanbari, and C. Amza. Model ensemble tools for self-management in data centers. In *Data Engineering Workshops (ICDEW), 2013 IEEE 29th International Conference on*, pages 36–43, April 2013.

[15] Tianshi Chen, Qi Guo, Ke Tang, Olivier Temam, Zhiwei Xu, Zhi-Hua Zhou, and Yunji Chen. Archranker: A ranking approach to design space exploration. *SIGARCH Comput. Archit. News*, 42(3):85–96, June 2014.

[16] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 1995.

[17] Maria Couceiro, Diego Didona, Lus Rodrigues, and Paolo Romano. Self-tuning in distributed transactional memory. In Rachid Guerraoui and Paolo Romano, editors, *Transactional Memory. Foundations, Algorithms, Tools, and Applications*, volume 8913 of *Lecture Notes in Computer Science*, pages 418–448. Springer International Publishing, 2015.

[18] Maria Couceiro, Paolo Romano, and Luis Rodrigues. A machine learning approach to performance prediction of total order broadcast protocols. In *Self-Adaptive and Self-Organizing Systems (SASO), 2010 4th IEEE International Conference on*, pages 184–193. IEEE, 2010.

[19] Maria Couceiro, Paolo Romano, and Luis Rodrigues. Polycert: Polymorphic self-optimizing replication for in-memory transactional grids. In *Proceedings of the 12th ACM/IFIP/USENIX International Conference on Middleware*, Middleware'11, pages 309–328, Berlin, Heidelberg, 2011. Springer-Verlag.

[20] Maria Couceiro, Pedro Ruivo, Paolo Romano, and Luis Rodrigues. Chasing the optimum in replicated in-memory transactional platforms via protocol adaptation. In *Proceedings of the 2013 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, DSN '13, pages 1–12, Washington, DC, USA, 2013. IEEE Computer Society.

[21] Pierangelo Di Sanzo, Francesco Antonacci, Bruno Ciciani, Roberto Palmieri, Alessandro Pellegrini, Sebastiano Peluso, Francesco Quaglia, Diego Rughetti, and Roberto Vitali. A framework for high performance simulation of transactional data grid platforms. In *Proceedings of the 6th International ICST Conference on Simulation Tools and Techniques*, SimuTools '13, pages 63–72, ICST, Brussels, Belgium, Belgium, 2013. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[22] Pierangelo Di Sanzo, Francesco Quaglia, Bruno Ciciani, Alessandro Pellegrini, Diego Didona, Paolo Romano, Roberto Palmieri, and Sebastiano Peluso. A flexible framework for accurate simulation of cloud in-memory data stores. *ArXiv e-prints*, December 2014.

[23] Pierangelo Di Sanzo, Diego Rughetti, Bruno Ciciani, and Francesco Quaglia. Auto-tuning of cloud-based in-memory transactional data grids via machine learning. In *Proceedings of the 2012 Second Symposium on Network Cloud Computing and Applications*, NCCA '12, pages 9–16, Washington, DC, USA, 2012. IEEE Computer Society.

[24] Diego Didona, Daniele Carnevale, Sergio Galeani, and Paolo Romano. An extremum seeking algorithm for message batching in total order protocols. In *SASO*, pages 89–98. IEEE Computer Society, 2012.

[25] Diego Didona, Pascal Felber, Derin Harmanci, Paolo Romano, and Joerg Schenker. Identifying the optimal level of parallelism in transactional memory applications. *Computing Journal*, pages 1–21, December 2013.

[26] Diego Didona, Francesco Quaglia, Paolo Romano, and Ennio Torre. Enhancing performance prediction robustness by combining analytical modeling and machine learning. In *Proceedings of the 2015 ACM/SPEC 6th International Conference on Performance Engineering (ICPE 2015)*, ICPE '15, 2015.

[27] Diego Didona and Paolo Romano. On Bootstrapping Machine Learning Performance Predictors via Analytical Models. *ArXiv e-prints*, October 2014.

[28] Diego Didona and Paolo Romano. Performance modelling of partially replicated in-memory transactional stores. In *Proceedings of the 22nd IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS 2014)*, MASCOTS '14, 2014.

[29] Diego Didona and Paolo Romano. Self-tuning transactional data grids: The cloud-tm approach. In *Proceedings of the Symposium on Network Cloud Computing and Applications, (NCCA)*, pages 113–120. IEEE, 2014.

[30] Diego Didona, Paolo Romano, Sebastiano Peluso, and Francesco Quaglia. Transactional auto scaler: Elastic scaling of replicated in-memory transactional data grids. *ACM Trans. Auton. Adapt. Syst.*, 9(2):11:1–11:32, July 2014.

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

puting (ICAC 14), pages 209–219, Philadelphia, PA, June 2014. USENIX Association.

[32] Thomas G. Dietterich. Ensemble methods in machine learning. In *Proc. of MCS Workshop*, 2000.

[33] Harris Drucker, Chris, Burges* L. Kaufman, Alex Smola, and Vladimir Vapnik. Support vector regression machines. In *Advances in Neural Information Processing Systems 9*, volume 9, pages 155–161, 1997.

[34] Jerome H. Friedman. Stochastic gradient boosting. *Comput. Stat. Data Anal.*, 38(4):367–378, February 2002.

[35] Toy Friedman and Robbert Van Renesse. Packing messages as a tool for boosting the performance of total ordering protocls. In *Proceedings of the 6th IEEE International Symposium on High Performance Distributed Computing*, HPDC '97, pages 233–, Washington, DC, USA, 1997. IEEE Computer Society.

[36] Richard M. Fujimoto. Parallel discrete event simulation. *Commun. ACM*, 33(10):30–53, October 1990.

[37] Archana Ganapathi, Harumi Kuno, Umeshwar Dayal, Janet L. Wiener, Armando Fox, Michael Jordan, and David Patterson. Predicting multiple metrics for queries: Better decisions enabled by machine learning. In *Proceedings of the 2009 IEEE International Conference on Data Engineering*, ICDE '09, pages 592–603, Washington, DC, USA, 2009. IEEE Computer Society.

[38] Saeed Ghanbari, Gokul Soundararajan, Jin Chen, and Cristiana Amza. Adaptive learning of metric correlations for temperature-aware database provisioning. In *Proceedings of the Fourth International Conference on Autonomic Computing*, ICAC '07, pages 26–, Washington, DC, USA, 2007. IEEE Computer Society.

[39] Donald Gross, John F Shortle, James M Thompson, and Carl M Harris. *Fundamentals of queueing theory*. John Wiley & Sons, 2013.

[40] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *The Journal of Machine Learning Research*, 3:1157–1182, 2003.

[41] Martin T. Hagan, Howard B. Demuth, and Mark Beale. *Neural Network Design*. PWS Publishing Co., Boston, MA, USA, 1996.

[42] Mark Hall et al. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18, November 2009.

[43] Herodotos Herodotou, Fei Dong, and Shivnath Babu. No one (cluster) size fits all: automatic cluster sizing for data-intensive analytics. In *Proc. of the ACM Symposium on Cloud Computing (SOCC)*, 2011.

[44] James R Jackson. Networks of waiting lines. *Operations research*, 5(4):518–521, 1957.

[45] Leonard Kleinrock. *Queueing Systems*, volume I: Theory. Wiley Interscience, 1975.

[46] John DC Little. A proof for the queuing formula: L= λ w. *Operations research*, 9(3):383–387, 1961.

[47] Wei-Yin Loh. Classification and regression trees. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(1):14–23, 2011.

[48] Philip M. Long and Rocco A. Servedio. Random classification noise defeats all convex potential boosters. *Mach. Learn.*, 78(3):287–304, March 2010.

[49] Daniel A. Menasce and Virgilio Almeida. *Capacity Planning for Web Services: Metrics, Models, and Methods*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 2001.

[50] Daniel A. Menasce, Lawrence W. Dowdy, and Virgilio A. F. Almeida. *Performance by Design: Computer Capacity Planning By Example*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2004.

[51] T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, Apr 1989.

[52] Matthias Nicola and Matthias Jarke. Performance modeling of distributed and replicated databases. *IEEE Trans. on Knowl. and Data Eng.*, 2000.

[53] J. R. Quinlan. Learning with continuous classes. In *Proceedings of the 5th Australian Joint Conference on Artificial Intelligence (AI)*, pages 343–348. World Scientific, 1992.

[54] J. R. Quinlan. Improved use of continuous attributes in c4.5. *J. Artif. Int. Res.*, 4(1):77–90, March 1996.

[55] J. R. Quinlan. Learning decision tree classifiers. *ACM Comput. Surv.*, 28(1):71–72, March 1996.

[56] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., 1993.

[57] Jia Rao, Xiangping Bu, Cheng-Zhong Xu, Leyi Wang, and George Yin. Vconf: A reinforcement learning approach to virtual machines autoconfiguration. In *Proceedings of the 6th International Conference on Autonomic Computing*, ICAC '09, pages 137–146, New York, NY, USA, 2009. ACM.

[58] M. Reiser and S. S. Lavenberg. Mean-value analysis of closed multichain queuing networks. *J. ACM*, 27(2):313–322, April 1980.

[59] Paolo Romano and Matteo Leonetti. Self-tuning batching in total order broadcast protocols via analytical modelling and reinforcement learning. In *International Conference on Computing, Networking and Communications.*, ICNC, 2011.

[60] Diego Rughetti, Pierangelo Di Sanzo, Bruno Ciciani, and Francesco Quaglia. Machine learning-based self-adjusting concurrency in software transactional memory systems. In *Proc. of the International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, MASCOTS, 2012.

[61] Diego Rughetti, Pierangelo Di Sanzo, Bruno Ciciani, and Francesco Quaglia. Analytical/ml mixed approach for concurrency regulation in software transactional memory. In *IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, CCGRID, 2014.

[62] G. A. Rummery and M. Niranjan. On-line Q-learning using connectionist systems. Technical Report TR 166, Cambridge University Engineering Department, Cambridge, England, 1994.

[63] Nuno Santos and André Schiper. Optimizing paxos with batching and pipelining. *Theor. Comput. Sci.*, 496:170–183, July 2013.

[64] Pierangelo Di Sanzo, Francesco Molfese, Diego Rughetti, and Bruno Ciciani. Providing transaction class-based qos in in-memory data grids via machine learning. In *Proceedings of the 2014 IEEE 3rd Symposium on Network Cloud Computing and Applications (Ncca 2014)*, NCCA '14, pages 46–53, Washington, DC, USA, 2014. IEEE Computer Society.

[65] Robert E. Schapire. The strength of weak learnability. *Mach. Learn.*, 5(2):197–227, July 1990.

[66] Robert E. Schapire. A brief introduction to boosting. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'99, pages 1401–1406, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[67] Bianca Schroeder, Mor Harchol-Balter, Arun Iyengar, Erich Nahum, and Adam Wierman. How to determine a good multi-programming level for external scheduling. In *Proc. of the International Conference on Data Engineering*, ICDE, 2006.

[68] Karan Singh, Engin İpek, Sally A. McKee, Bronis R. de Supinski, Martin Schulz, and Rich Caruana. Predicting parallel application performance via machine learning approaches: Research articles. *Concurr. Comput. : Pract. Exper.*, 19(17):2219–2235, December 2007.

[69] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, pages 2951–2959, 2012.

[70] Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998.

[71] Y. C. Tay. *Analytical Performance Modeling for Computer Systems, Second Edition*. Morgan & Claypool Publishers, 2013.

[72] Gerald Tesauro, Nicholas K. Jong, Rajarshi Das, and Mohamed N. Bennani. On the use of hybrid reinforcement learning for autonomic resource allocation. *Cluster Computing*, 2007.

[73] Eno Thereska and Gregory R. Ganger. Ironmodel: Robust performance models in the wild. *SIGMETRICS Perform. Eval. Rev.*, 36, June 2008.

[74] Chris Thornton, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Auto-weka: Combined selection and hyperparameter optimization of classification algorithms. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, pages 847–855, New York, NY, USA, 2013. ACM.

[75] D. Tsoumakos, I. Konstantinou, C. Boumpouka, S. Sioutas, and N. Koziris. Automated, elastic resource provisioning for nosql clusters using tiramola. In *Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on*, pages 34–41, May 2013.

[76] Laurens JP van der Maaten, Eric O Postma, and H Jaap van den Herik. Dimensionality reduction: A comparative review. Technical Report TiCC-TR 2009-005, Tilburg University, 2009.

[77] Christopher John Cornish Hellaby Watkins. *Learning from delayed rewards.* PhD thesis, University of Cambridge, 1989.

[78] Pawel T. Wojciechowski, Tadeusz Kobus, and Maciej Kokocinski. Model-driven comparison of state-machine-based and deferred-update replication schemes. In *Proceedings of the 2012 IEEE 31st Symposium on Reliable Distributed Systems*, SRDS '12, pages 101–110, Washington, DC, USA, 2012. IEEE Computer Society.

[79] David H. Wolpert. Original contribution: Stacked generalization. *Neural Netw.*, 5(2):241–259, February 1992.

[80] David H. Wolpert. The lack of a priori distinctions between learning algorithms. *Neural Comput.*, 8(7):1341–1390, October 1996.

[81] Pengcheng Xiong, Yun Chi, Shenghuo Zhu, Hyun Jin Moon, Calton Pu, and Hakan Hacigumus. Intelligent management of virtualized resources for database systems in cloud environment. In *Proceedings of the 2011 IEEE 27th International Conference on Data Engineering*, ICDE '11, pages 87–98, Washington, DC, USA, 2011. IEEE Computer Society.

[82] Pengcheng Xiong, Yun Chi, Shenghuo Zhu, Junichi Tatemura, Calton Pu, and Hakan HacigümüŞ. Activesla: A profit-oriented admission control framework for database-as-a-service providers. In *Proceedings of the 2Nd ACM Symposium on Cloud Computing*, SOCC '11, pages 15:1–15:14, New York, NY, USA, 2011. ACM.

[83] Steve Zhang, Ira Cohen, Julie Symons, and Armando Fox. Ensembles of models for automated diagnosis of system performance problems. In *Proceedings of the 2005 International Conference on Dependable Systems and Networks*, DSN '05, pages 644–653, Washington, DC, USA, 2005. IEEE Computer Society.

[84] Ludmila Cherkasova, Kivanc Ozonat, Ningfang Mi, Julie Symons, and Evgenia Smirni. Automated anomaly detection and performance modeling of enterprise applications. *ACM Trans. Comput. Syst.*, 27(3):6:1–6:32, November 2009.

[85] Yongmin Tan, Hiep Nguyen, Zhiming Shen, Xiaohui Gu, Chitra Venkatramani, and Deepak Rajan. Prepare: Predictive performance anomaly prevention for virtualized cloud systems. In *Proceedings of the 2012 IEEE 32Nd International Conference on Distributed Computing Systems*, ICDCS '12, pages 285–294, Washington, DC, USA, 2012. IEEE Computer Society.

[86] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Computing Surveys (CSUR)*, 41(3):15, 2009.

[87] Ira Cohen, Moises Goldszmidt, Terence Kelly, Julie Symons, and Jeffrey S. Chase. Correlating instrumentation data to system states: A building block for automated diagnosis and control. In *Proceedings of the 6th Conference on Symposium on Opearting Systems Design & Implementation - Volume 6*, OSDI'04, pages 16–16, Berkeley, CA, USA, 2004. USENIX Association.