Active-Learning-Based Surrogate Models for Empirical Performance Tuning

Prasanna Balaprakash

Joint work with

R. B. Gramacy* and S. M. Wild

Mathematics and Computer Science Division
Argonne National Laboratory
Argonne, IL

*Booth School of Business
University of Chicago, IL

The 10th workshop of the INRIA-Illinois-ANL Joint Laboratory, NCSA, IL, 2013

Motivation: road to 10^{18} by 2018

No exascale for you! — H. Simon, LBNL, 2013

- power is a primary design constraint
- exponential growth of parallelism
- compute growing 2x faster than memory and bandwidth
- data movement cost more than that of FLOPS
- need more heterogeneity
- hardware errors

The Rest of This Talk:
Tackling the Tornado

Automatic Performance Tuning

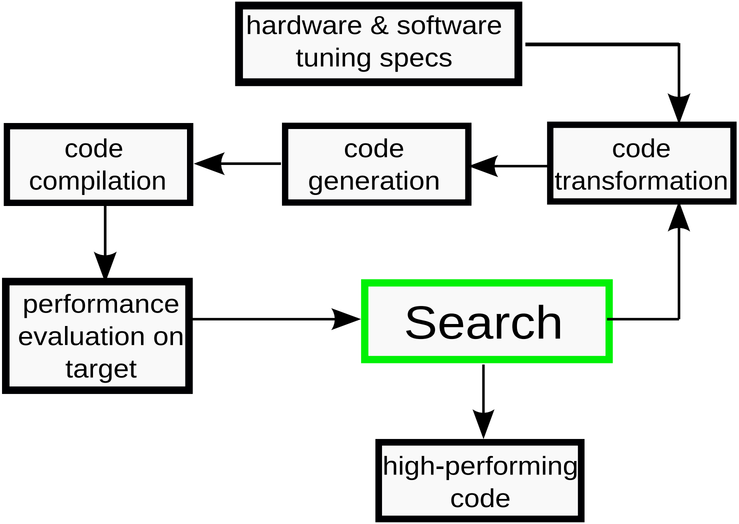
Performance Modeling

Active Learning

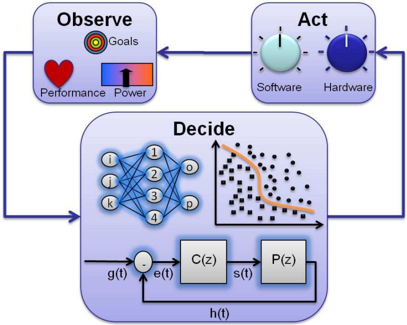
Experimental Results

Automatic performance tuning

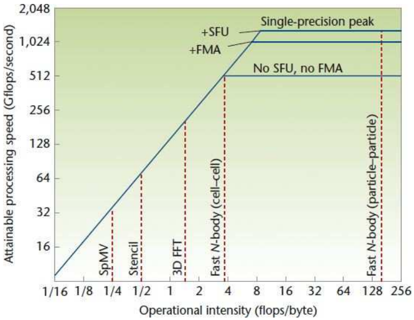
Given an application & a target architecture:



Performance models in autotuning



See [H. Hoffmann, World Changing Ideas, SA 2009]



See [S. Williams et al., ACM 2009]

- insights on important knobs that impacts performance
- avoid running the corresponding code configuration on the target
- can help prune large search spaces

Machine learning for performance modeling

- algebraic performance models increasingly challenging
- statistical performance models: an effective alternative
- small number of input-output points obtained from empirical evaluation
- deployed to test and/or aid search, compiler, and autotuning

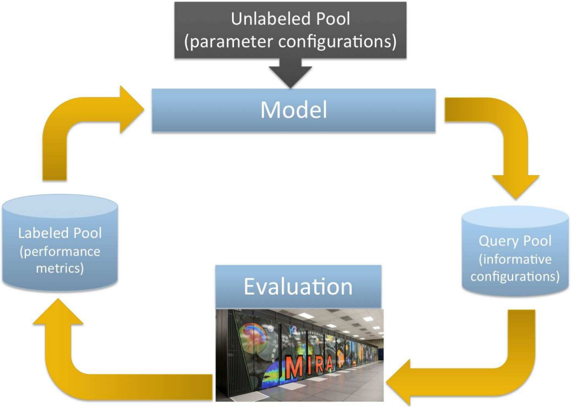
Machine learning for performance modeling

- algebraic performance models increasingly challenging
- statistical performance models: an effective alternative
- small number of input-output points obtained from empirical evaluation
- deployed to test and/or aid search, compiler, and autotuning

Goal

efficiently using HPC systems to minimize the number of expensive evaluations on the target machine

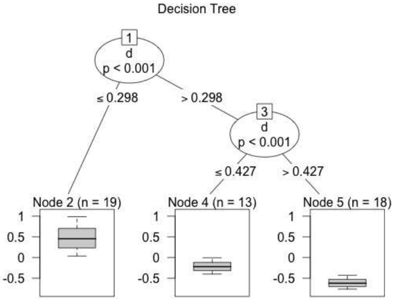
Active learning for performance modeling



- key idea: greater accuracy with fewer training points when allowed to choose the training data
- actively query the model to assess predictive variance

Active learning using dynaTrees

- Based on a classical nonparametric (do not rely on data belonging to any particular distribution) modeling technique [M. Taddy et al. 2011]

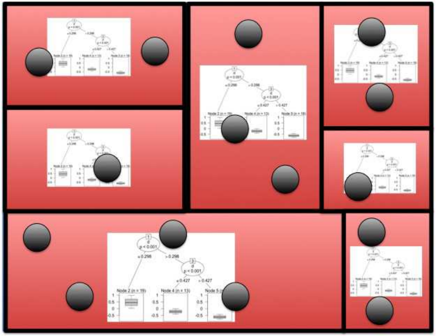


Algorithm

- trees to represent input-output relationships using binary recursive partitioning

Active learning using dynaTrees

- Based on a classical nonparametric (do not rely on data belonging to any particular distribution) modeling technique [M. Taddy et al. 2011]

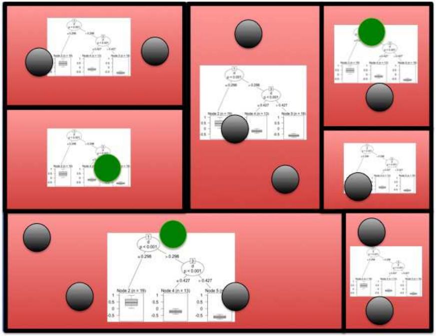


Algorithm

- trees to represent input-output relationships using binary recursive partitioning
- the covariate space is partitioned into a set of hyper-rectangles
- a simple tree model is fit within each rectangle

Active learning using dynaTrees

- Based on a classical nonparametric (do not rely on data belonging to any particular distribution) modeling technique [M. Taddy et al. 2011]

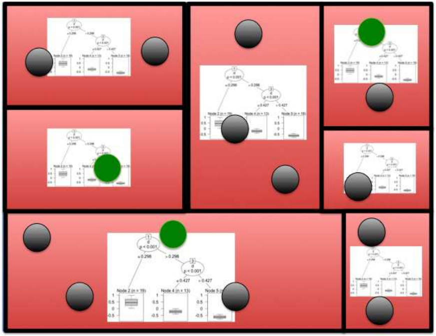


Algorithm

- trees to represent input-output relationships using binary recursive partitioning
- the covariate space is partitioned into a set of hyper-rectangles
- a simple tree model is fit within each rectangle
- generate a pool of unlabeled points

Active learning using dynaTrees

- Based on a classical nonparametric (do not rely on data belonging to any particular distribution) modeling technique [M. Taddy et al. 2011]

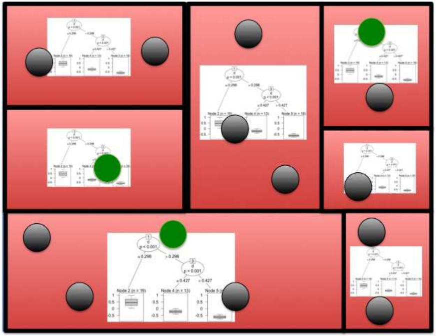


Algorithm

- trees to represent input-output relationships using binary recursive partitioning
- the covariate space is partitioned into a set of hyper-rectangles
- a simple tree model is fit within each rectangle
- generate a pool of unlabeled points
- selection: maximize the expected reduction in predictive variance

Active learning using dynaTrees

- Based on a classical nonparametric (do not rely on data belonging to any particular distribution) modeling technique [M. Taddy et al. 2011]



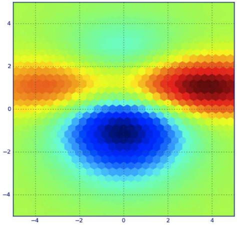
Algorithm

- trees to represent input-output relationships using binary recursive partitioning
- the covariate space is partitioned into a set of hyper-rectangles
- a simple tree model is fit within each rectangle
- generate a pool of unlabeled points
- selection: maximize the expected reduction in predictive variance

sequential!

Active learning with concurrent evaluations

- batch (n_b) of inputs, taken collectively, will lead to updates that are better than one-at-a-time schemes



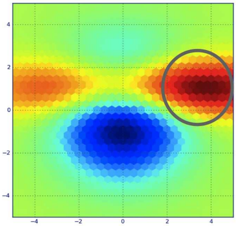
The `ab-dynaTree` algorithm

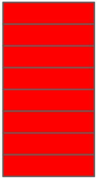
- select points and evaluate concurrently

Active learning with concurrent evaluations

- batch (n_b) of inputs, taken collectively, will lead to updates that are better than one-at-a-time schemes

The ab-dynaTree algorithm

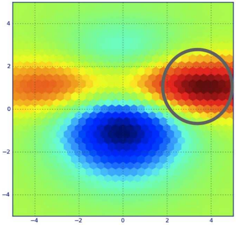




- select points and evaluate concurrently
- issue: other configurations in the batch become less informative

Active learning with concurrent evaluations

- batch (n_b) of inputs, taken collectively, will lead to updates that are better than one-at-a-time schemes

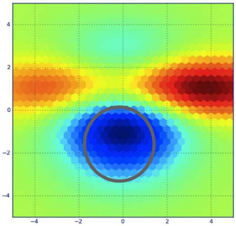


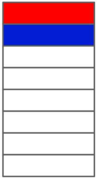
The ab-dynaTree algorithm

- select points and evaluate concurrently
- issue: other configurations in the batch become less informative
- condition sampling on tentative evaluations

Active learning with concurrent evaluations

- batch (n_b) of inputs, taken collectively, will lead to updates that are better than one-at-a-time schemes



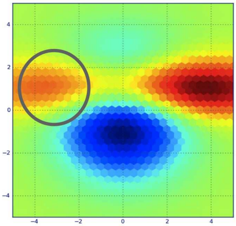


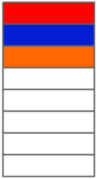
The ab-dynaTree algorithm

- select points and evaluate concurrently
- issue: other configurations in the batch become less informative
- condition sampling on tentative evaluations
- $\mu(x_{prev}) \leftarrow \mu_{pred}(x_{prev})$;
  $\implies \sigma^2(x_{prev}) \leftarrow 0$

Active learning with concurrent evaluations

- batch (n_b) of inputs, taken collectively, will lead to updates that are better than one-at-a-time schemes



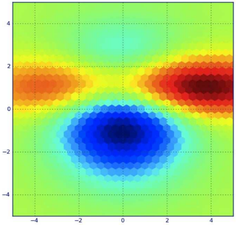


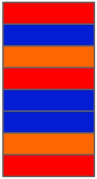
The `ab-dynaTree` algorithm

- select points and evaluate concurrently
- issue: other configurations in the batch become less informative
- condition sampling on tentative evaluations
- $\mu(x_{prev}) \leftarrow \mu_{pred}(x_{prev})$; $\implies \sigma^2(x_{prev}) \leftarrow 0$
- better exploration

Active learning with concurrent evaluations

- batch (n_b) of inputs, taken collectively, will lead to updates that are better than one-at-a-time schemes





The ab-dynaTree algorithm

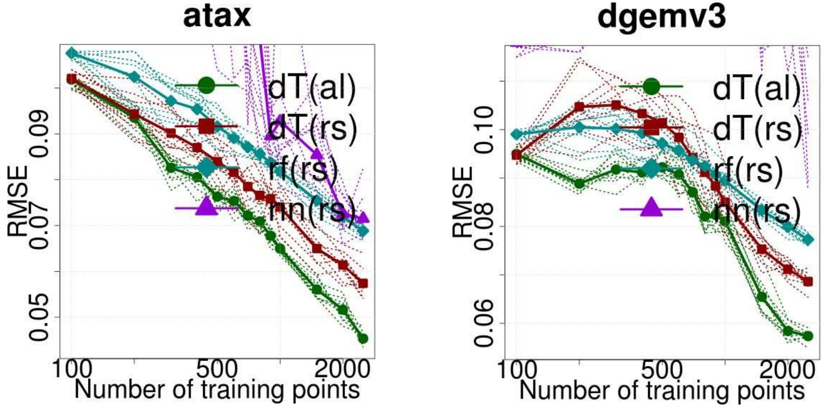
- select points and evaluate concurrently
- issue: other configurations in the batch become less informative
- condition sampling on tentative evaluations
- $\mu(x_{prev}) \leftarrow \mu_{pred}(x_{prev})$; $\implies \sigma^2(x_{prev}) \leftarrow 0$
- better exploration
- leads to better surrogates with minimum evaluations

Experimental setup

- SPAPT test suite [Balaprakash, Norris, & Wild, ICCS '12]
    - elementary linear algebra, linear solver, stencil codes, elementary data mining
    - SPAPT problem = code + set of transformations + parameter specifications + constraints + input size
    - Orio framework [Hartono, Norris, & Sadayappan, IPDPS '09]
- `ab-dynaTree` algorithm with a maximum budget of $2,500$ evaluations ($\mathcal{X}_{\text{out}}$, $\mathcal{Y}_{\text{out}}$)
- three non linear regression algorithms: dynaTrees algorithm (dT), random forest (rf), neural networks (nn)
- active learning (al) variants: ($\mathcal{X}_{\text{out}}$,$\mathcal{Y}_{\text{out}}$) as the training set
- random sampling (rs) variants: 2,500 randomly chosen points
- test set $\mathcal{T}_{25\%}$: the subset of data points whose mean run times are within the lower 25% quartile of the empirical distribution for the run times
- root-mean-squared error (RMSE) as a measure of prediction accuracy

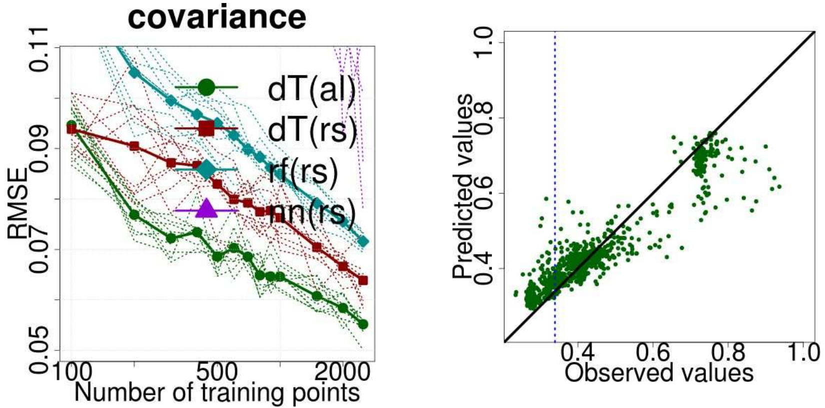
Modeling runtimes of SPAPT kernels

Intel Nehalem: 2.53 GHz processors, 64 KB L1 cache, and 36 GB memory



◇ **Double win: Better RMSE, less evaluations** (=time/evaluation)

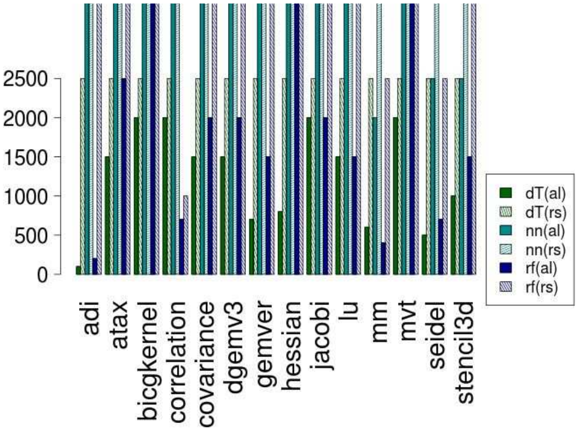
Modeling runtimes of SPAPT kernels



◇ 14/14 SPAPT problems active learning variants performs better than random search variants

Modeling runtimes of SPAPT kernels

◇ dT(rs) with 2,500 evaluations as a baseline



◇ **Savings up to a factor of six**

Comparison between regression algorithms

Table: RMSE averaged over 10 replications on the $\mathcal{T}_{25\%}$ test set for 2,500 training points: *italics* (**bold**) when a variant is significantly *worse* (**better**) than dT(al) according to a t-test with significance (alpha) level 0.05.

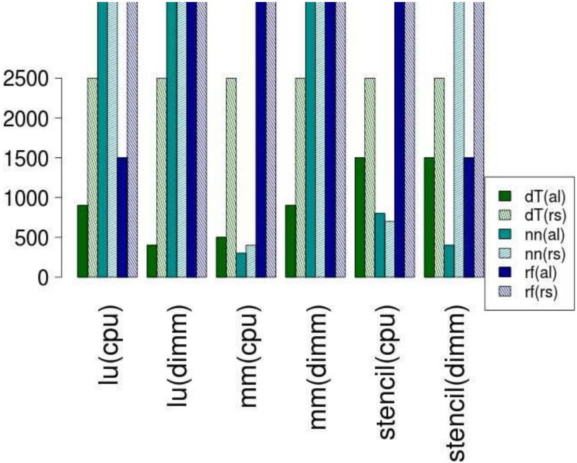
Problem	dT(al)	dT(rs)	nn(al)	nn(rs)	rf(al)	rf(rs)
adi	0.021	*0.025*	*0.034*	*0.031*	*0.022*	*0.025*
atax	0.045	*0.057*	*0.064*	*0.072*	*0.056*	*0.069*
bicgkernel	0.021	*0.024*	*0.038*	*0.043*	*0.032*	*0.038*
correlation	0.060	*0.066*	*0.212*	*0.199*	**0.053**	**0.057**
covariance	0.055	*0.064*	*0.104*	*0.114*	*0.059*	*0.072*
dgemv3	0.057	*0.069*	*0.100*	*0.137*	*0.065*	*0.077*
gemver	0.100	*0.120*	*0.155*	*0.180*	*0.103*	*0.132*
hessian	0.045	*0.054*	*0.059*	*0.070*	*0.070*	*0.094*
jacobi	0.029	*0.045*	*0.058*	*0.057*	*0.044*	*0.053*
lu	0.037	*0.060*	*0.072*	*0.084*	*0.050*	*0.067*
mm	0.064	*0.079*	*0.078*	*0.079*	**0.061**	*0.075*
mvt	0.032	*0.036*	*0.044*	*0.053*	*0.044*	*0.053*
seidel	0.076	*0.097*	*0.092*	*0.098*	*0.080*	*0.095*
stencil3d	0.080	*0.100*	*0.100*	*0.122*	*0.084*	*0.105*

- ◇ dT(al) and nn(al) are similar due to expensive parameter tuning of nn

Modeling power in HPC kernels

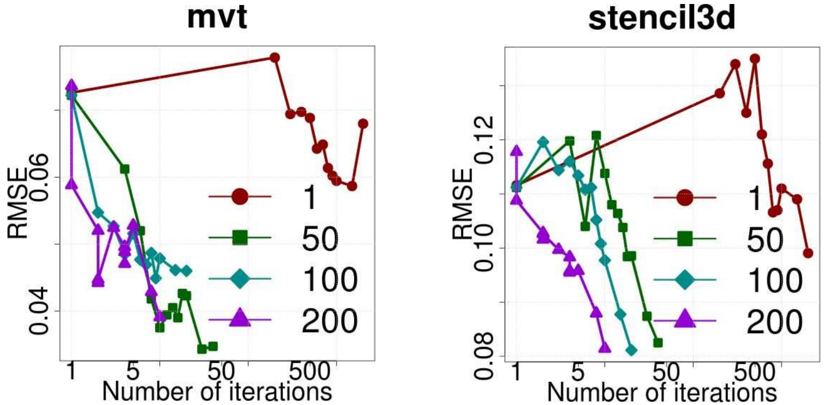
Intel Xeon E5530, 32 KB L1, 256 KB L2 (data from [Tiwari et al., IPDPSW '12])

- dT(rs) with 2,500 evaluations as a baseline



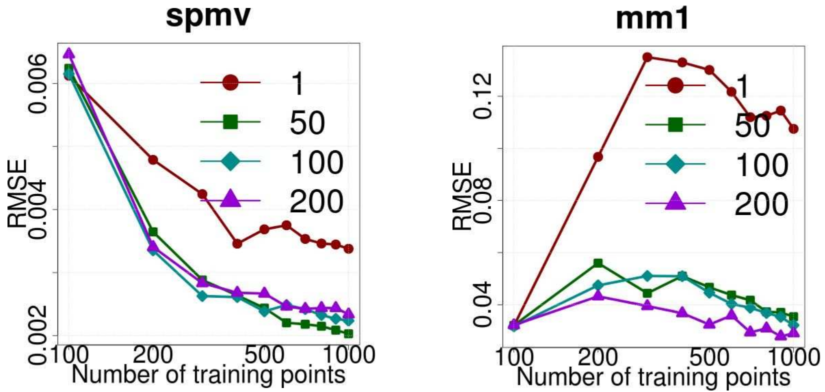
- **savings up to a factor of four**

Impact of batch size (n_b) in `ab-dynaTree`



- $n_b > 1$: explore and identify multiple regions in the input space
- $n_b = 1$: high probability of sampling from only one promising region

Impact of batch size (n_b) in `ab-dynaTree` for GPU kernels



◇ on **7 out of 9** GPU problems, large batch size beneficial even when concurrent evaluations are not feasible

Summary

- `ab-dynaTree` for developing empirical performance models
- active learning as an effective data acquisition strategy
- batch mode of provides significant benefits over the classical, serial mode: high degree of exploration

use active learning for empirical performance modeling

Future work

- asynchronous model updates
- multiobjective surrogate modeling
- structure exploiting numerical optimization algorithms
- deployment of `ab-dynaTree` in autotuning search algorithms

References

- P. Balaprakash, R. Gramacy, and S. M. Wild. Active-learning-based surrogate models for empirical performance tuning. IEEE Cluster, 2013
- P. Balaprakash, A. Tiwari, and S. M. Wild. Multi-objective optimization of HPC kernels for performance, power, and energy. PMBS 13, 2013
- P. Balaprakash, S. M. Wild, and P. Hovland, Can search algorithms save large-scale automatic performance tuning? ICCS 2011
- P. Balaprakash, S. M. Wild, and B. Norris. SPAPT: Search problems in automatic performance tuning, ICCS 2012
- A. Hartono, B. Norris, and P. Sadayappan. Annotation-based empirical performance tuning using Orio, IPDPS, 2009

Orio `https://github.com/brnorris03/Orio`

→ Thank you!