Empirical performance modeling of GPU kernels using active learning

Paul D. Hovland

Joint work with

P. Balaprakash, K. Rupp, A. Mametjanov, R. B. Gramacy, S. M. Wild

Mathematics and Computer Science Division
Argonne National Laboratory
Argonne, IL

ParCo 2013 - Symposium: Application Autotuning for HPC, Munich, Germany

August 22, 2013

Motivation

GPU computing

- performance gains for algorithms exposing fine grained parallelism
- improvements of about an order of magnitude over CPUs

Autotuning in GPU getting more difficult

- mapping even simple linear algebra operations to the underlying hardware can be difficult
- high cardinality of the set of kernel instantiations (block sizes for better cache reuse and the number of concurrent threads)
- long execution times to find high performing variant

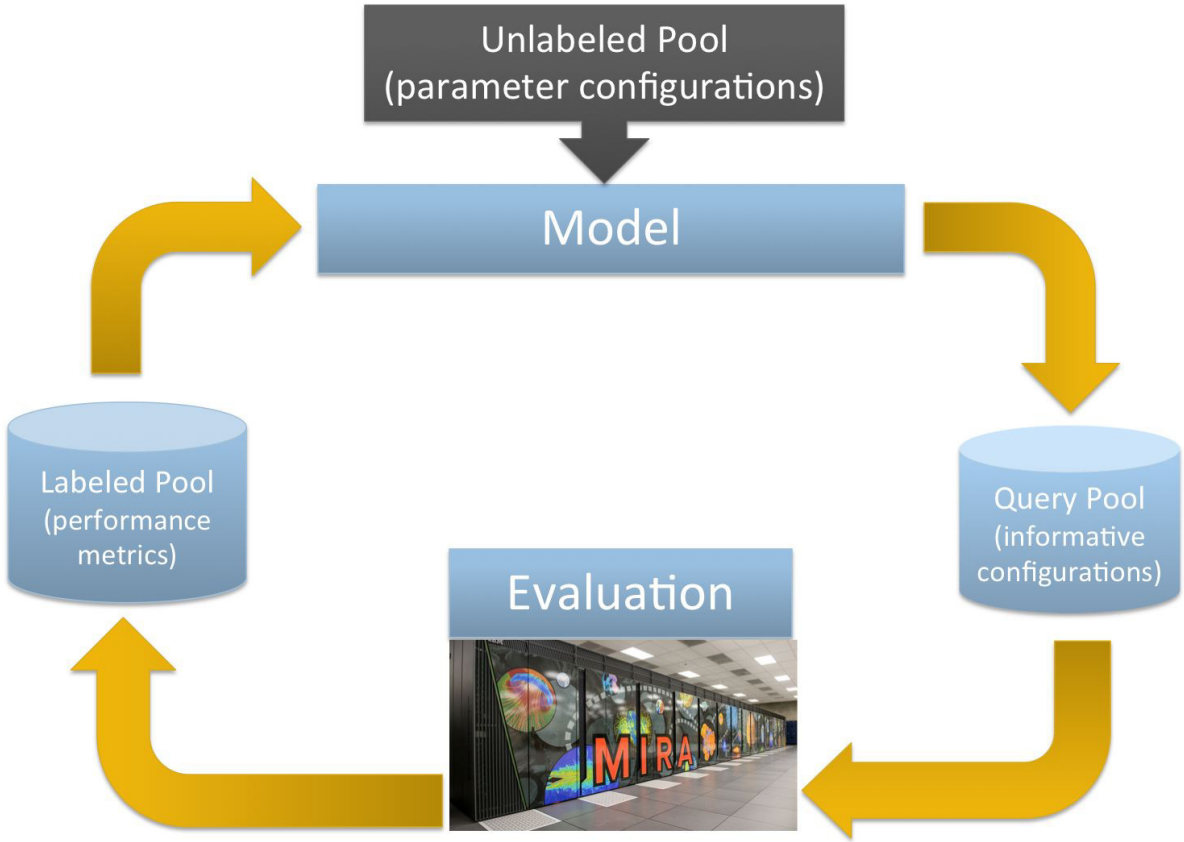
Performance portability

- GPU computing presents a significant burden for application developers

Machine learning for performance modeling

- performance modeling may provide a much more efficient methodology for coping with high-dimensional search spaces in auto tuning
- 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 auto tuning

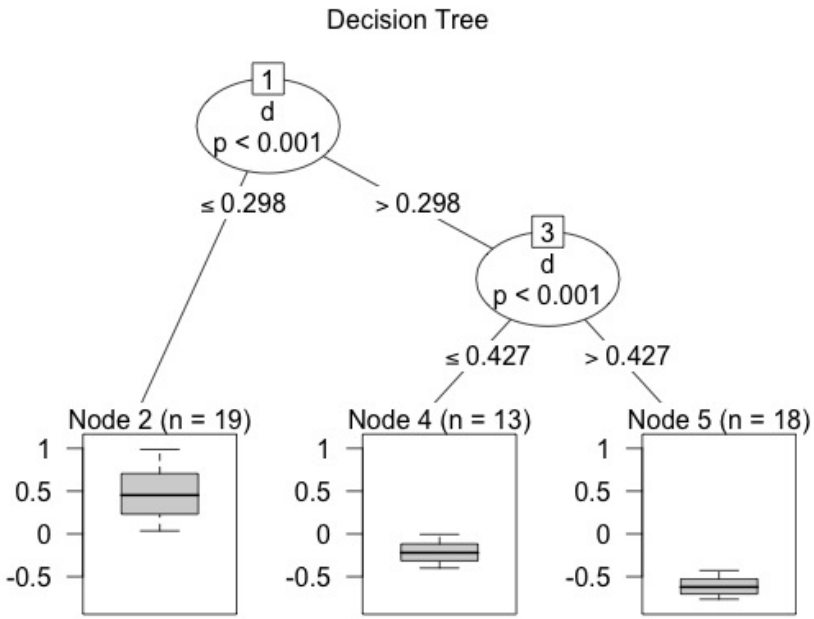
Active learning for performance modeling



- 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]

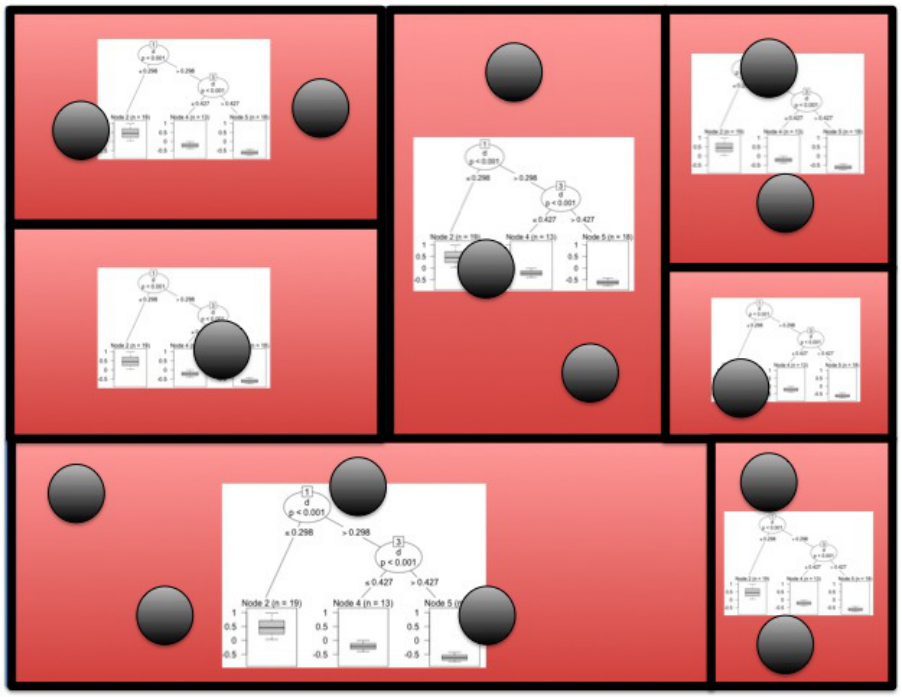


Algorithm

- 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]

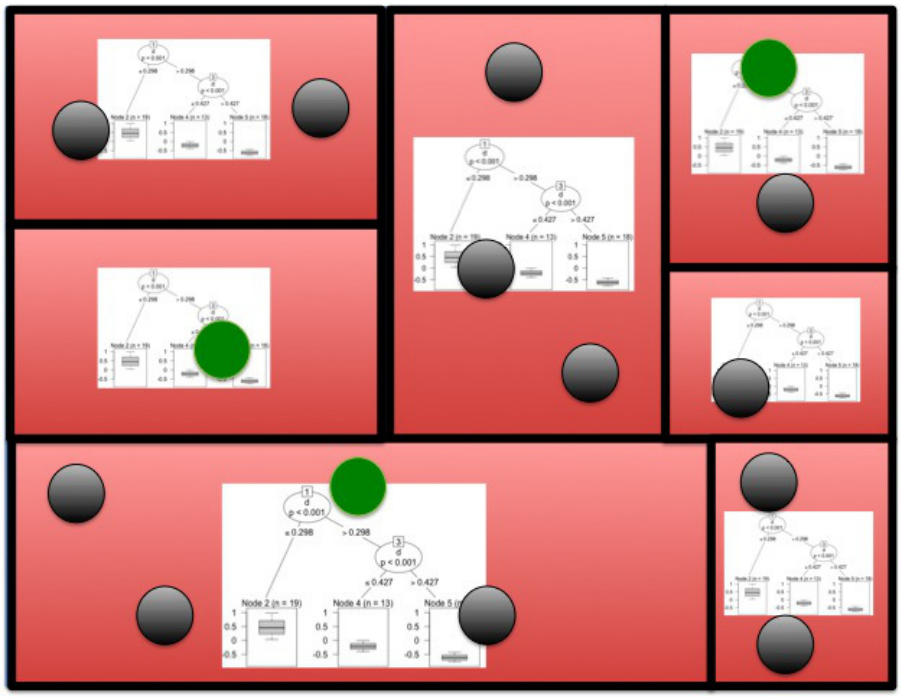


Algorithm

- 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]

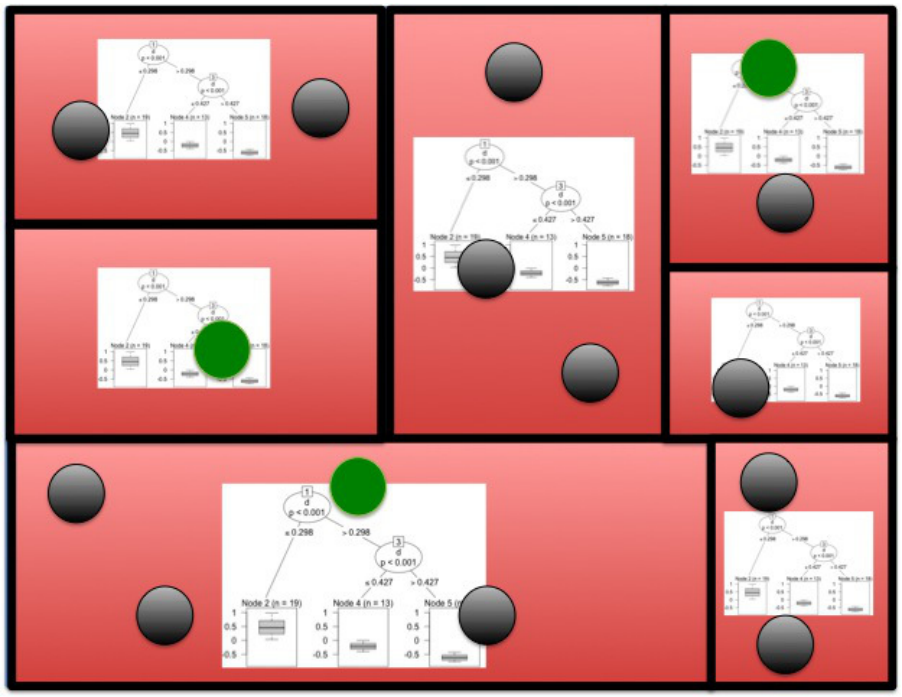


Algorithm

- 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]

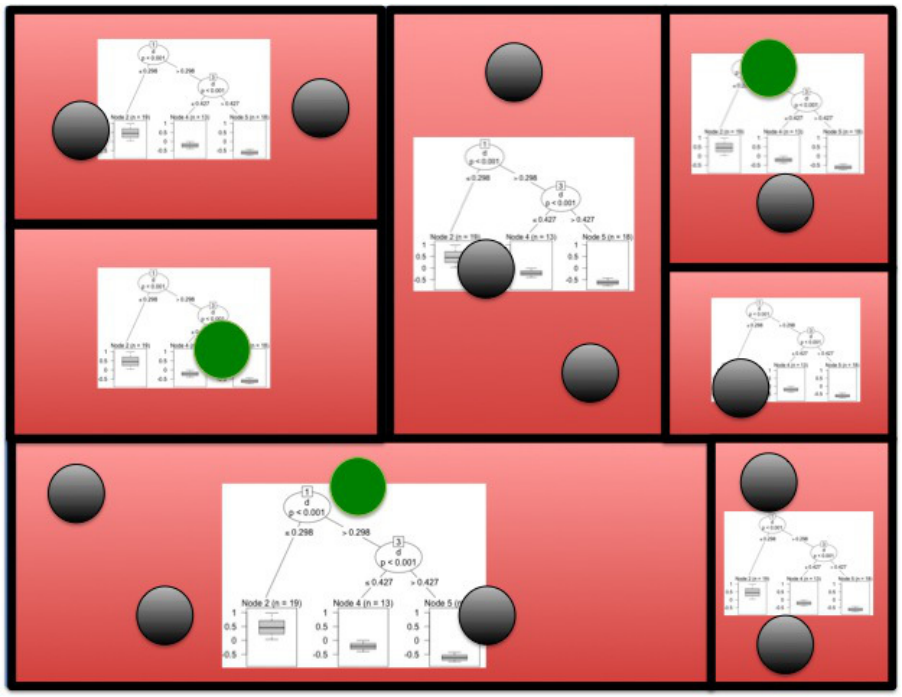


Algorithm

- 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]



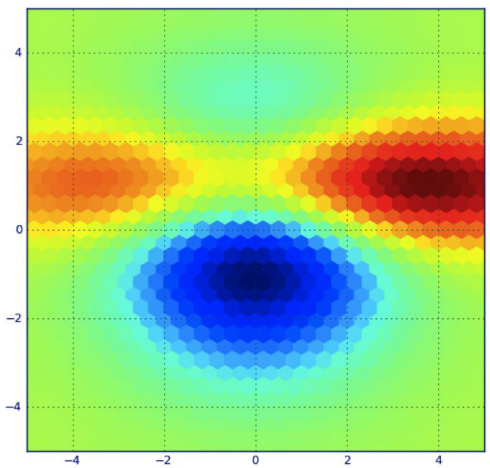
Algorithm

- 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

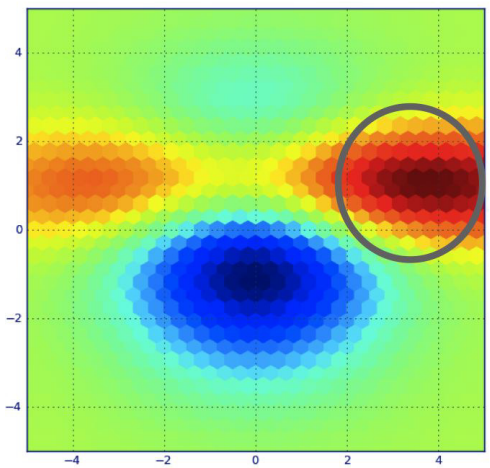


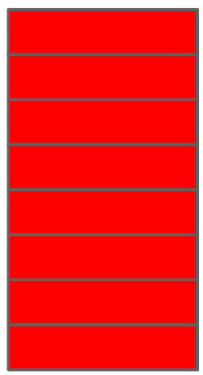
The `ab-dynaTree` algorithm

- 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



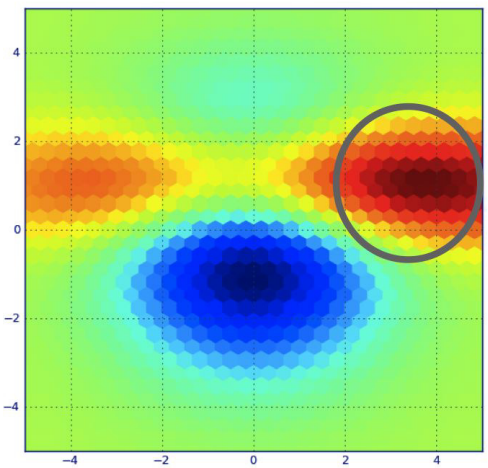


The `ab-dynaTree` algorithm

- 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

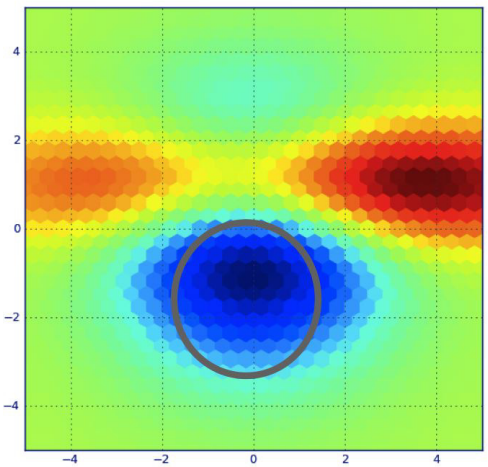


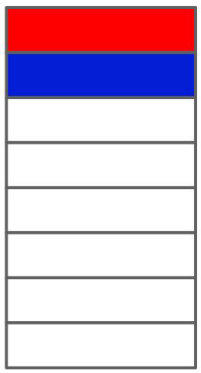
The `ab-dynaTree` algorithm

- 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



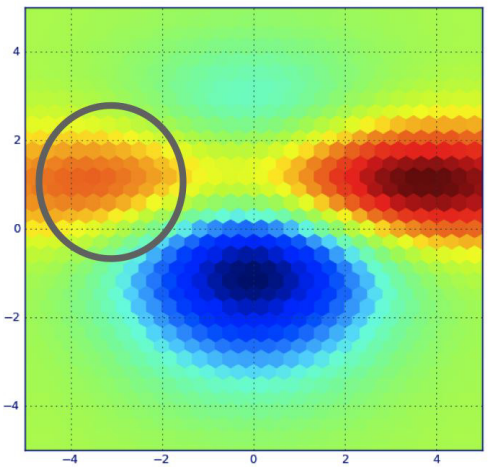


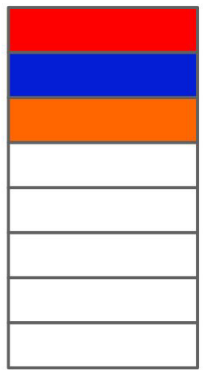
The `ab-dynaTree` algorithm

- 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



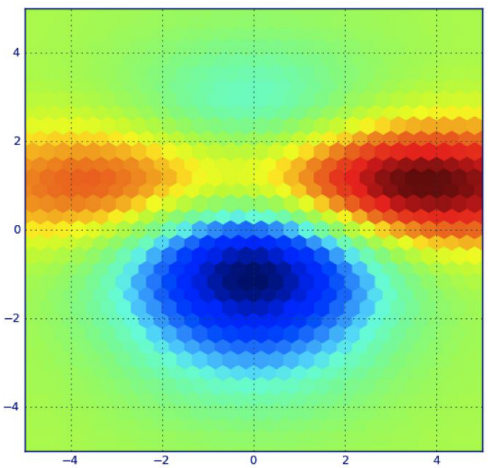


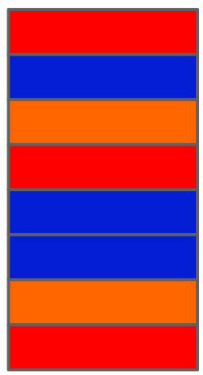
The `ab-dynaTree` algorithm

- 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

See [Balaprakash et al., Cluster'13]

The ab-dynaTree algorithm for GPU kernels

- kernels executed in sequence on a single device (absence of a GPU cluster)
- $n_b = 1$: "serial version" of ab-dynaTree
- $n_b > 1$: model is updated only after these n_b serial evaluations

The `ab-dynaTree` algorithm for GPU kernels

- kernels executed in sequence on a single device (absence of a GPU cluster)
- $n_b = 1$: "serial version" of `ab-dynaTree`
- $n_b > 1$: model is updated only after these n_b serial evaluations

Goal

does batching still provide significant benefits over the classical sequential strategy?

Experimental setup

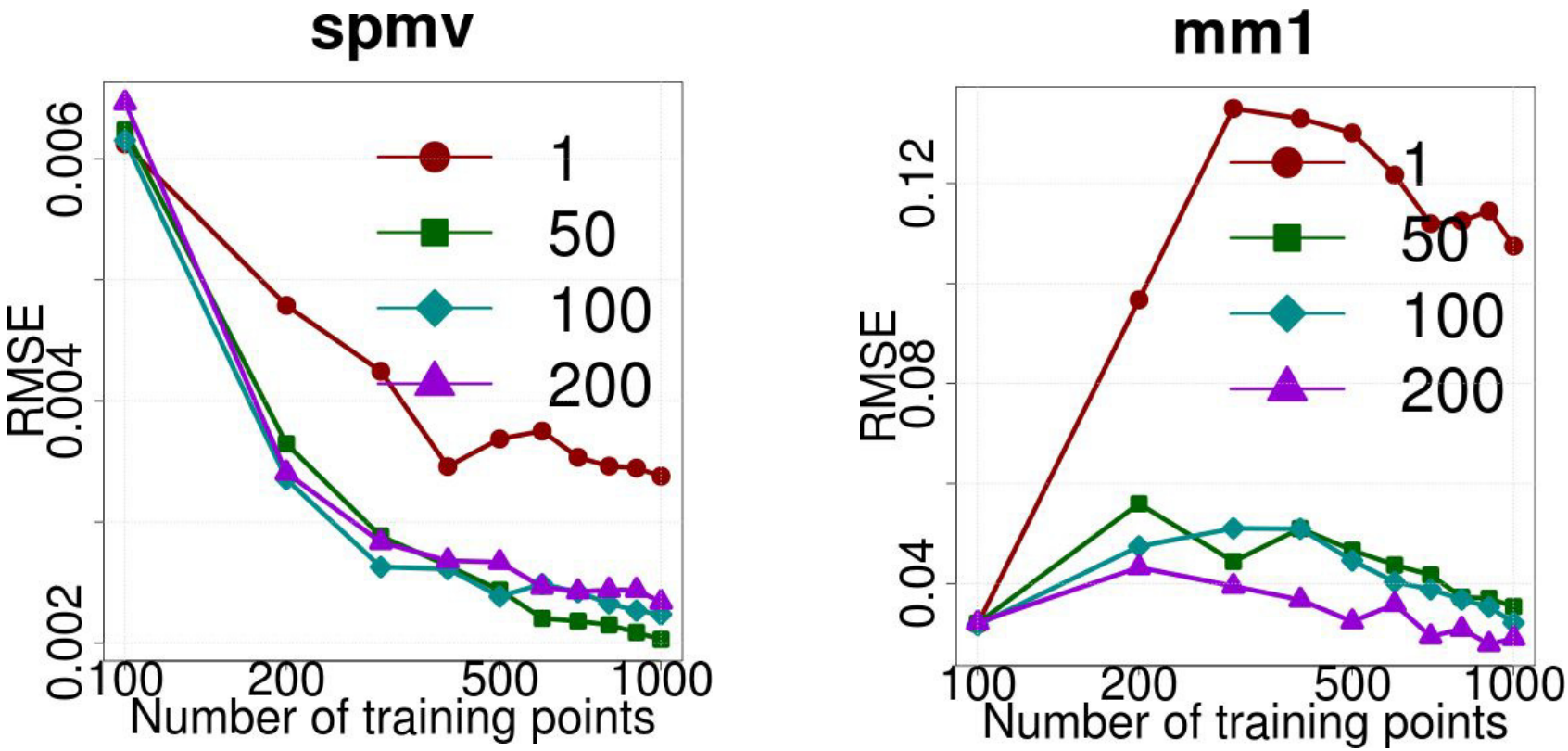
Problems	Operations	Graphic Card	# Param.	Valid Conf. $\lvert\mathcal{X}_p\rvert$
vc1	vector copy	Nvidia GeForce GTX 285	4	2,560
vc2	vector copy	Nvidia GeForce GTX 470	4	2,560
vc3	vector copy	Nvidia Tesla C2050	4	2,560
vc4	vector copy	AMD Radeon HD 7970	4	2,560
vdot	vector dot product	Nvidia Tesla C2050	6	6,144
axpy	vector-scalar product	Nvidia Tesla C2050	6	7,680
spmv	sparse matrix-vector	Nvidia Tesla K20X	4	10,752
mm1	matrix multiplication	Nvidia GeForce GTX 470	10	8,465
mm2	matrix multiplication	AMD Radeon HD 7970	10	3,568

- surrogate models to minimize execution times
- all valid configurations evaluated and stored in a lookup table

Experimental setup

- `ab-dynaTree` algorithm with a maximum budget of $1,000$ 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: 1,000 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

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
- on **7 out of 9** problems, large batch size beneficial

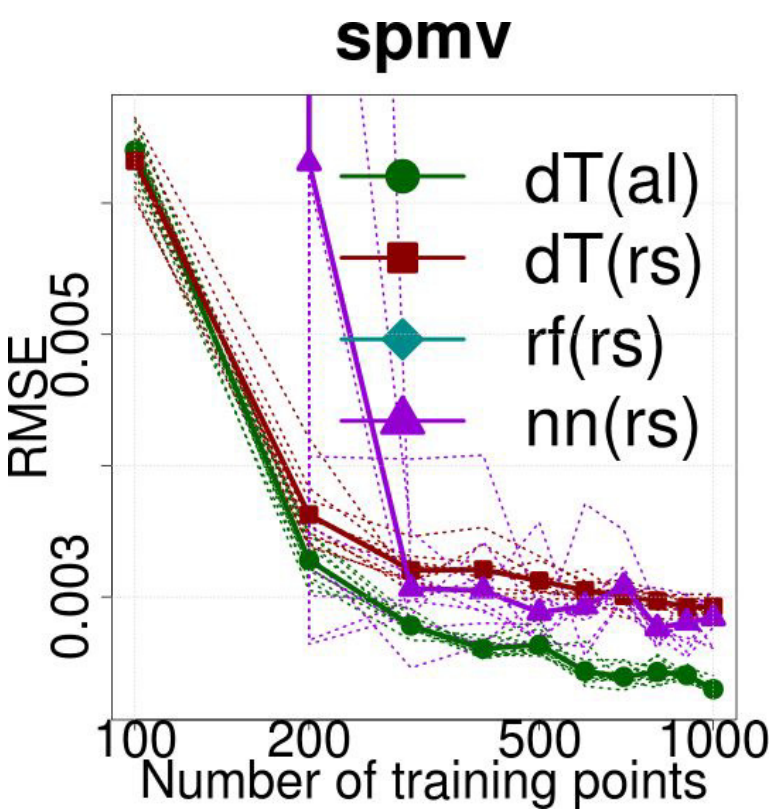
Comparison between regression algorithms

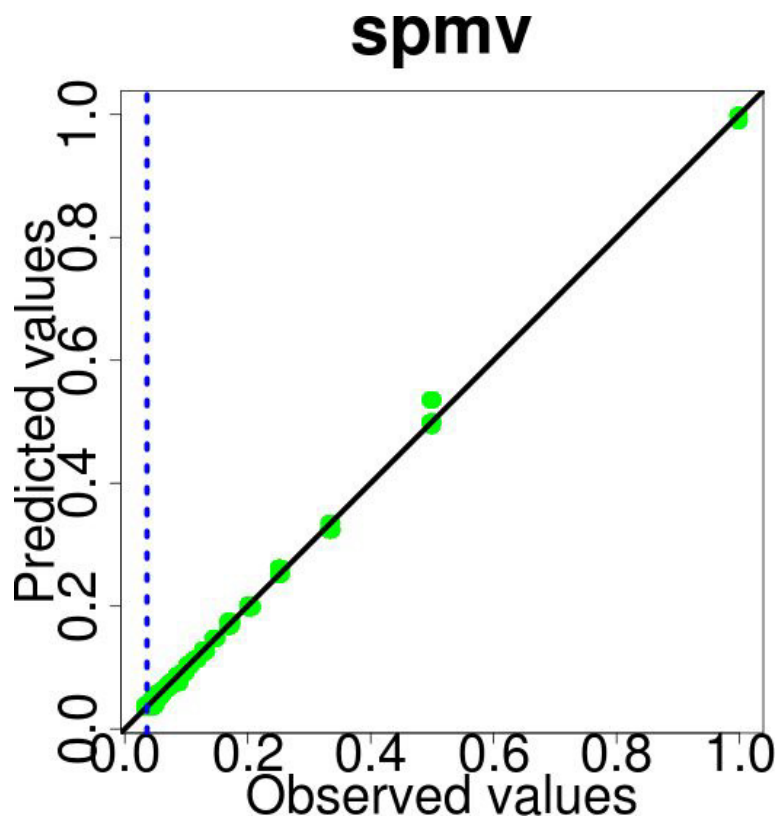
Table : RMSE averaged over 10 replications on the $\mathcal{T}_{25\%}$ test set for 1,000 training points. The value is typeset in *italics* (**bold**) when a variant is significantly *worse* (**better**) than dT(al) according to a t-test with significance (alpha) level 0.05.

Problem	dT(al)	dT(rs)	nn(al)	nn(rs)	rf(al)	rf(rs)
vc1	0.035	*0.050*	0.036	*0.044*	*0.122*	*0.179*
vc2	0.041	*0.067*	0.039	*0.058*	*0.137*	*0.173*
vc3	0.124	*0.201*	0.131	*0.173*	*0.262*	*0.372*
vc4	0.026	*0.043*	*0.031*	*0.038*	*0.124*	*0.153*
vdot	0.009	*0.014*	0.010	*0.016*	*0.012*	*0.021*
axpy	0.014	*0.022*	0.012	*0.017*	*0.016*	*0.029*
spmv	0.002	*0.003*	0.002	*0.003*	*0.008*	*0.014*
mm1	0.029	*0.045*	0.027	*0.040*	0.028	*0.046*
mm2	0.036	*0.053*	**0.030**	*0.043*	**0.031**	*0.052*

- al variants of dT, rf, and nn obtain lower RMSE than rs variants
- dT(al) completely dominates the three random sampling variants
- dT(al) obtains lower average RMSE than does rf(al)
- dT(al) and nn(al) are similar due to expensive parameter tuning of nn

Comparison between regression algorithms

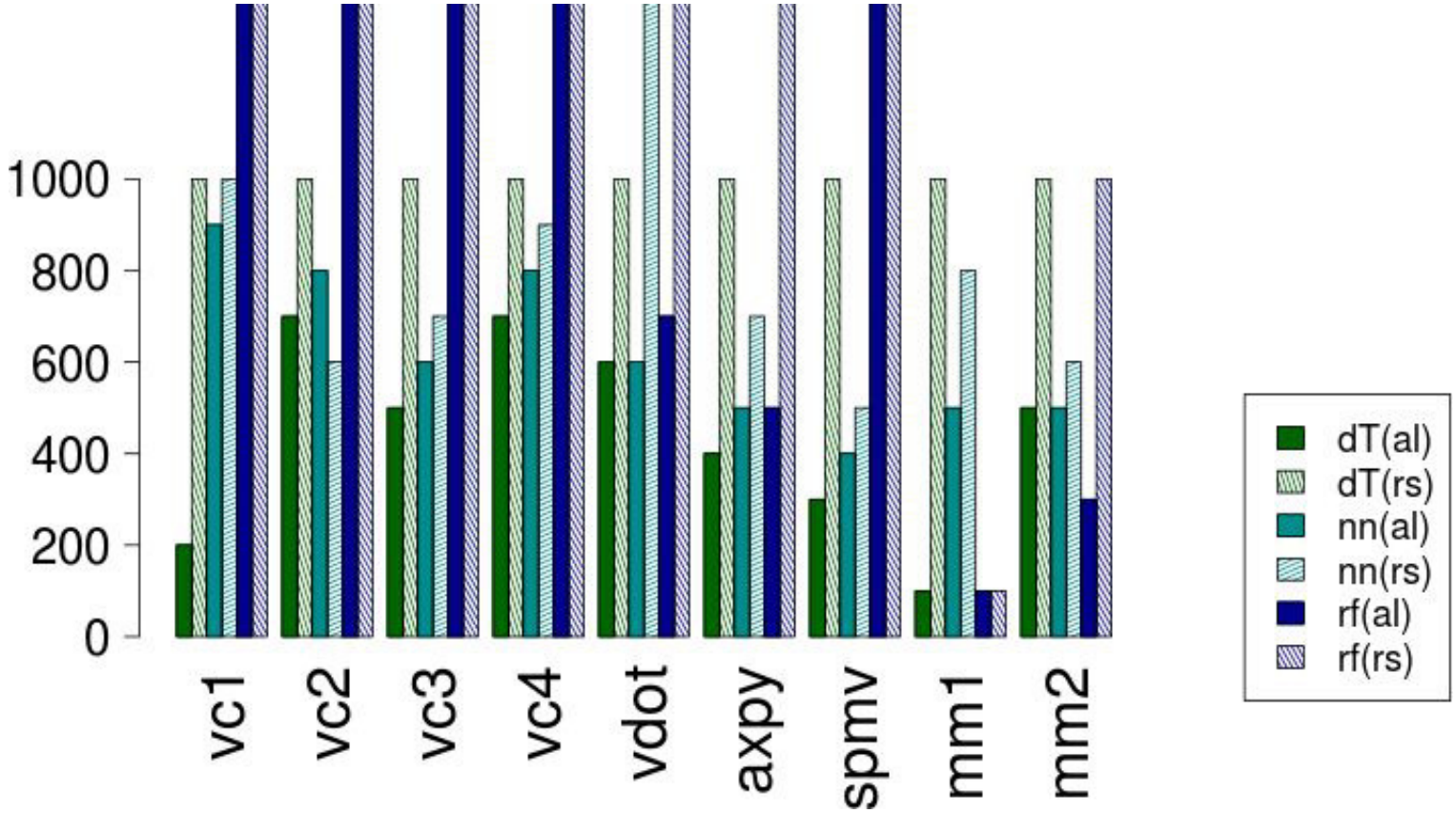




- **Double win: Better RMSE, less training points** (=time/evaluation)
- dT(al) requires relatively fewer evaluations to achieve a smaller RMSE

Comparison between regression algorithms

- number of evaluations required by the variants to reach the RMSE obtained by dT(rs) with 1,000 evaluations.



- on **7 out of 9** problems, dT(al) reaches the RMSE of dT(rs) within 300 to 700 training points; mm1 and mm2 rf(al) outperform dT(al)
- savings up to a factor of three

Summary

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

Future work

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