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:

- Search
  - high-performing code
  - code transformation
  - code generation
  - code compilation
  - performance evaluation on target
  - hardware & software tuning specs
Performance models in autotuning

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

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

See [S. Williams et al., ACM 2009]
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 \textit{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 \textit{ab-dynaTree} algorithm

- select points and evaluate concurrently
- issue: other configurations in the batch become less informative
- condition sampling on tentative evaluations
- \( \mu(x_{prev}) \leftarrow \mu_{pred}(x_{prev}) ; \quad \sigma^2(x_{prev}) \leftarrow 0 \)
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:
  \[
  \mu(x_{prev}) \leftarrow \mu_{\text{pred}}(x_{prev});
  \quad \sigma^2(x_{prev}) \leftarrow 0
  \]
- 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
- \[ \mu(x_{prev}) \leftarrow \mu_{pred}(x_{prev}); \quad \sigma^2(x_{prev}) \leftarrow 0 \]
- 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 \((X_{\text{out}}, Y_{\text{out}})\)

- three non linear regression algorithms: dynaTrees algorithm (dT), random forest (rf), neural networks (nn)

- active learning (al) variants: \((X_{\text{out}}, Y_{\text{out}})\) as the training set

- random sampling (rs) variants: 2,500 randomly chosen points

- test set \(T_{25\%}\): 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

- **atax**
- **dgemv3**

- 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 $T_{25\%}$ 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.

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

diamond 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


https://github.com/brnorris03/Orio

→ Thank you!