Efficient Performance Modeling for Evolving Software

FOSD Meeting 2019

Stefan Mühlbauer

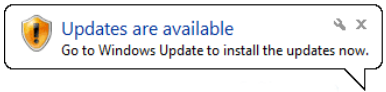
March 11, 2019

Bauhaus-Universität Weimar

Intelligent Software Systems Group

Poor Software Performance

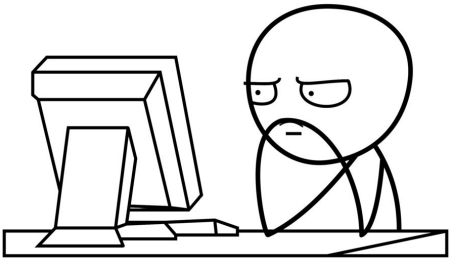
Poor Software Performance



Poor Software Performance

Poor Software Performance

Has anyone ever experienced this?



Poor Software Evolution

Architectural drift:
Insensisitvity about and violations of original architecture

Poor Software Evolution

Architectural drift:
Insensisitvity about and violations of original architecture

Architectural erosion:
Missing coherence and adaptibility makes software brittle (technical debt)

Performance in the Presence of Variability

- Performance-bugs are often configuration-specific [Han and Yu, 2016]

Performance in the Presence of Variability

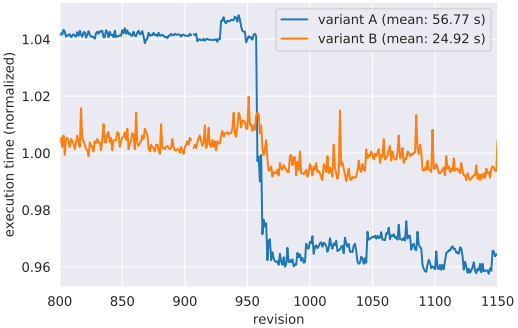
- Performance-bugs are often configuration-specific [Han and Yu, 2016]
- Configuration-specific performance can be predicted (performance influence models)

Performance in the Presence of Variability

- Performance-bugs are often configuration-specific [Han and Yu, 2016]
- Configuration-specific performance can be predicted (performance influence models)
- Performance can evolve independently for configurations!

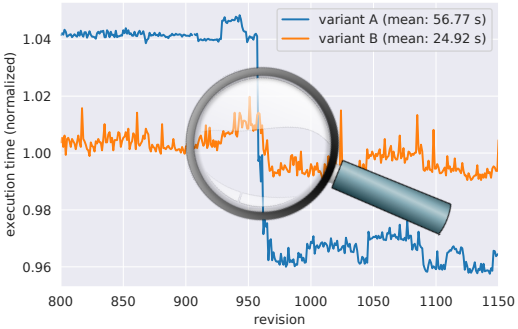
Performance in the Presence of Variability

- Performance-bugs are often configuration-specific [Han and Yu, 2016]
- Configuration-specific performance can be predicted (performance influence models)
- Performance can evolve independently for configurations!
- Example: GNU XZ (file compression)



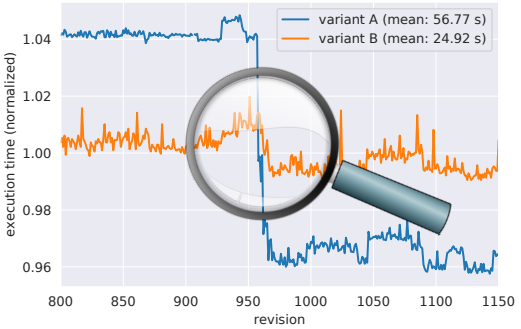
Performance in the Presence of Variability

- Performance-bugs are often configuration-specific [Han and Yu, 2016]
- Configuration-specific performance can be predicted (performance influence models)
- Performance can evolve independently for configurations!
- Example: GNU XZ (file compression)



Performance in the Presence of Variability

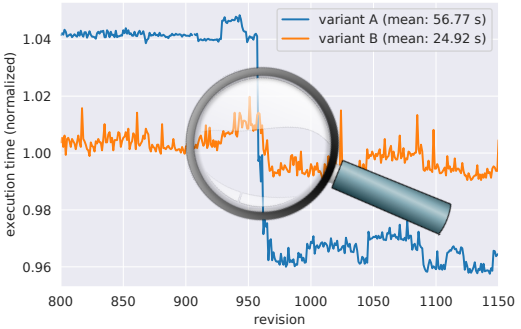
- Performance-bugs are often configuration-specific [Han and Yu, 2016]
- Configuration-specific performance can be predicted (performance influence models)
- Performance can evolve independently for configurations!
- Example: GNU XZ (file compression)



r957: liblzma: Add lzma_memcmplen() for **fast memory comparison**. This commit **just adds the function**. Its uses will be in separate commits.

Performance in the Presence of Variability

- Performance-bugs are often configuration-specific [Han and Yu, 2016]
- Configuration-specific performance can be predicted (performance influence models)
- Performance can evolve independently for configurations!
- Example: GNU XZ (file compression)



r957: liblzma: Add lzma_memcmplen() for **fast memory comparison**. This commit **just adds the function**. Its uses will be in separate commits.

r958: liblzma: Use lzma_memcmplen() in the **match finders**.

Case Study: Performance Evolution

- Empirical analysis of performance of 6+ real-world systems
  - Configurable software systems
  - Libraries with micro-benchmarks

Case Study: Performance Evolution

- Empirical analysis of performance of 6+ real-world systems
  - Configurable software systems
  - Libraries with micro-benchmarks

x264

lrzip

GNU XZ

Case Study: Performance Evolution

- Empirical analysis of performance of 6+ real-world systems
  - Configurable software systems
  - Libraries with micro-benchmarks

x264

lrzip

GNU XZ

Case Study: Performance Evolution

- Empirical analysis of performance of 6+ real-world systems
  - Configurable software systems
  - Libraries with micro-benchmarks

x264

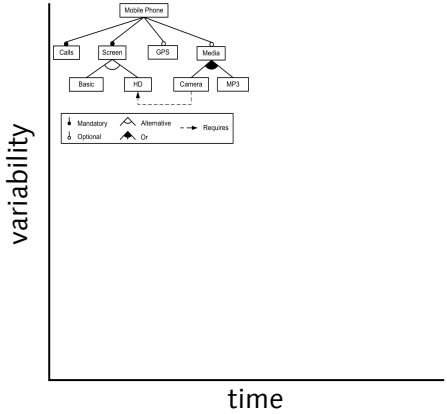
lrzip

GNU XZ

NumPy

SciPy

astropy



Case Study: Performance Evolution

- Empirical analysis of performance of 6+ real-world systems
  - Configurable software systems
  - Libraries with micro-benchmarks

x264

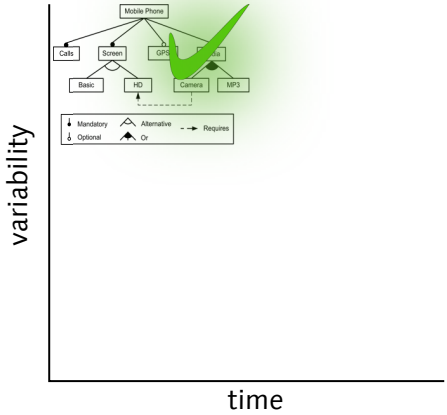
lrzip

GNU XZ

NumPy

SciPy

astropy



Case Study: Performance Evolution

- Empirical analysis of performance of 6+ real-world systems
  - Configurable software systems
  - Libraries with micro-benchmarks

x264

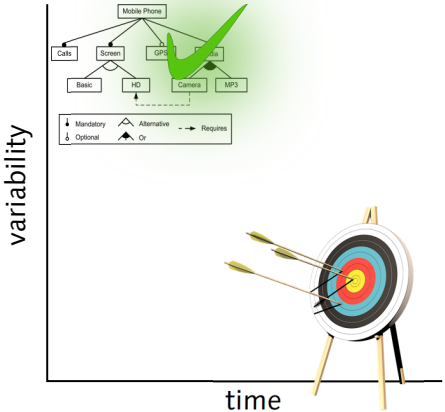
lrzip

GNU XZ

NumPy

SciPy

astropy



Case Study: Performance Evolution

- Empirical analysis of performance of 6+ real-world systems
  - Configurable software systems
  - Libraries with micro-benchmarks

x264

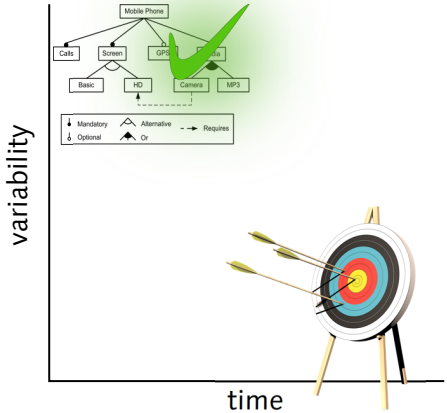
lrzip

GNU XZ

NumPy

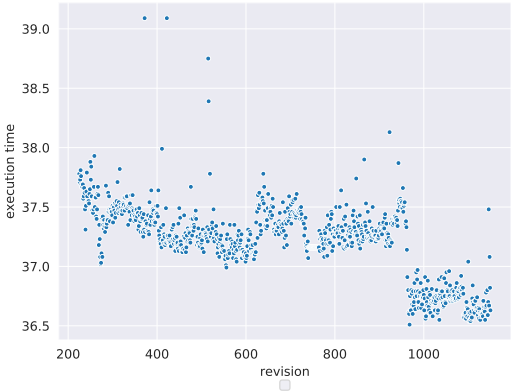
SciPy

astropy



RQ_1: What are characteristics of performance evolution?

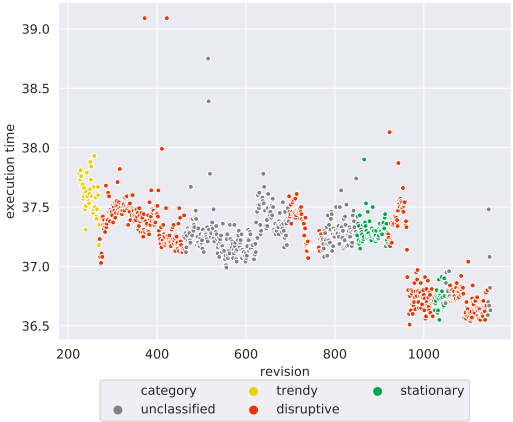
Reasons to (de-)select prediction models



RQ_1: What are characteristics of performance evolution?

Reasons to (de-)select prediction models

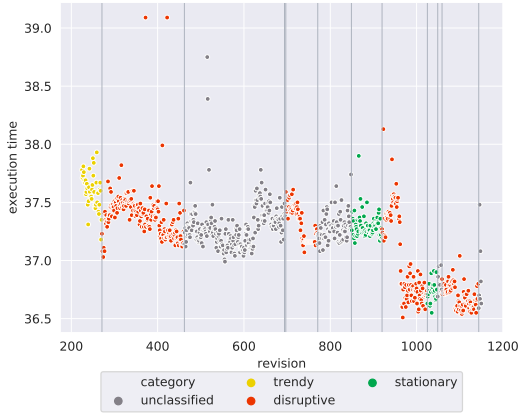
- Time-series characteristics?
  - Stationarity
  - Trends
  - Change-points



RQ_1: What are characteristics of performance evolution?

Reasons to (de-)select prediction models

- Time-series characteristics?
  - Stationarity
  - Trends
  - Change-points
- On what scales do patterns exist?
  - Release-to-release version
  - Feature-model iterations
  - Merges, Bugfixes, …



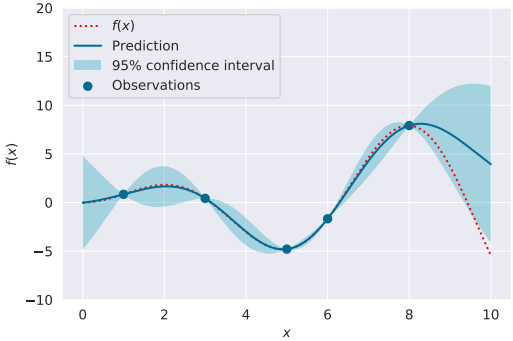
RQ_2: Can we model performance evolution efficiently?

Disruptive changes are hard to pinpoint,
can we search for them?

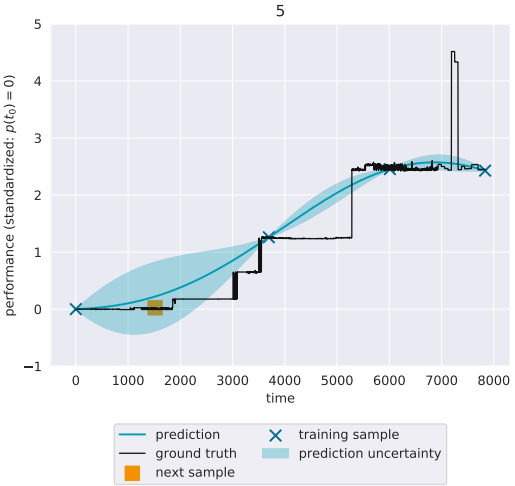
RQ_2: Can we model performance evolution efficiently?

Disruptive changes are hard to pinpoint, can we search for them?

- Gaussian Process Regression for time-series [Roberts et al., 2012]
- Framework for actively learning time-series when obtaining samples is expensive



RQ_2: Adaptive Learning with Gaussian Processes (GP)

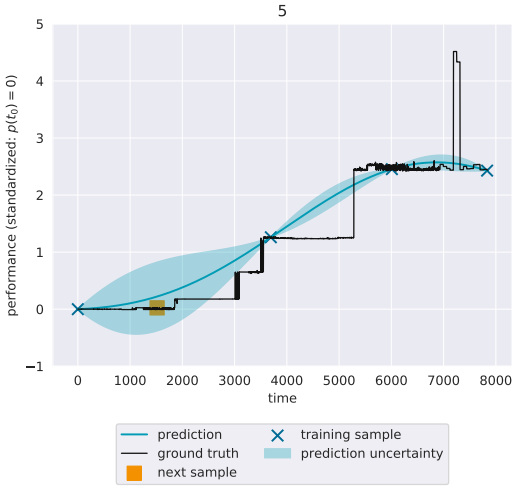


(1) Initial random sampling of k revisions.

(2) Training of GP regressor with sample set (MLE).

(3) Prediction

RQ_2: Adaptive Learning with Gaussian Processes (GP)



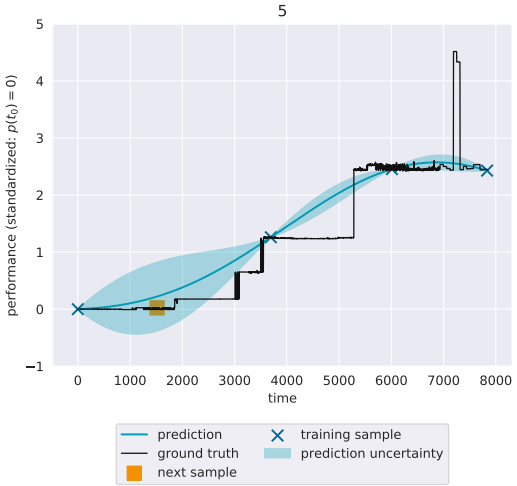
(1) Initial random sampling of k revisions.

(2) Training of GP regressor with sample set (MLE).

(3) Prediction

(4) Acquisition of next sample with maximum prediction variance

RQ_2: Adaptive Learning with Gaussian Processes (GP)



(1) Initial random sampling of k revisions.

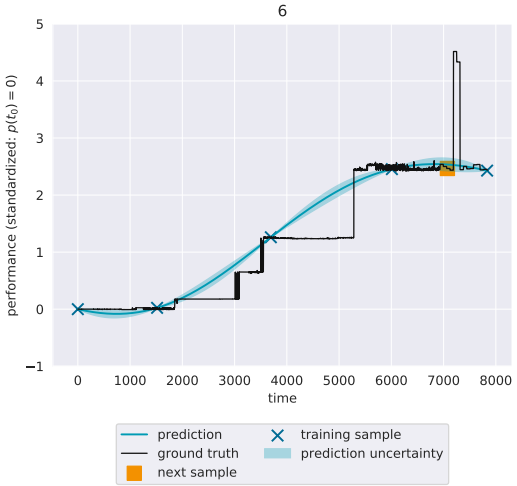
(2) Training of GP regressor with sample set (MLE).

(3) Prediction

(4) Acquisition of next sample with maximum prediction variance

(5) Repeat (2-4) until termination criteria is met

RQ_2: Adaptive Learning with Gaussian Processes (GP)



(1) Initial random sampling of k revisions.

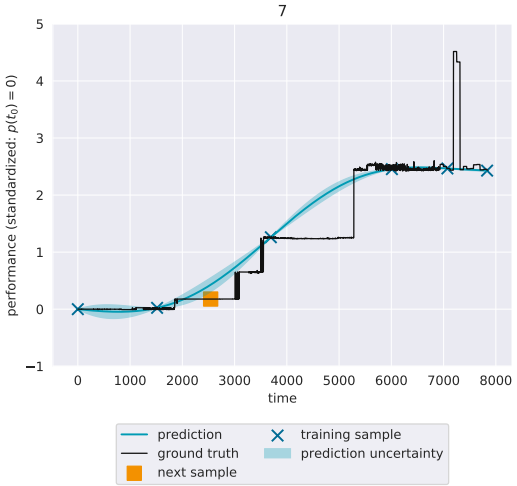
(2) Training of GP regressor with sample set (MLE).

(3) Prediction

(4) Acquisition of next sample with maximum prediction variance

(5) Repeat (2-4) until termination criteria is met

RQ_2: Adaptive Learning with Gaussian Processes (GP)



(1) Initial random sampling of k revisions.

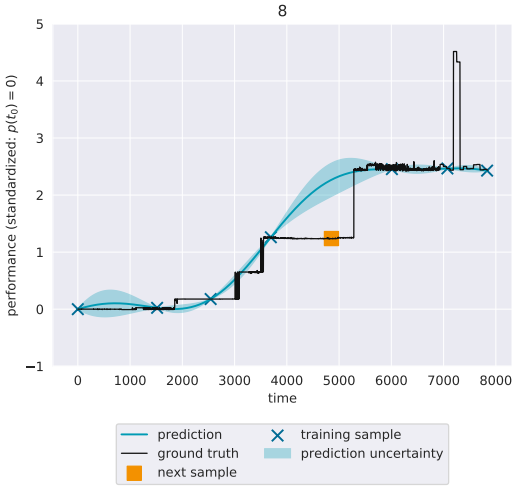
(2) Training of GP regressor with sample set (MLE).

(3) Prediction

(4) Acquisition of next sample with maximum prediction variance

(5) Repeat (2-4) until termination criteria is met

RQ_2: Adaptive Learning with Gaussian Processes (GP)



(1) Initial random sampling of k revisions.

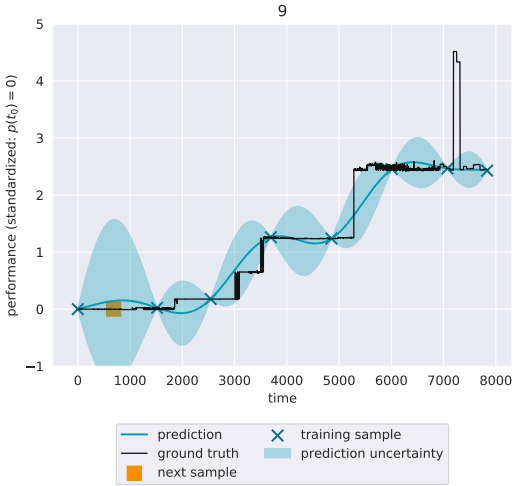
(2) Training of GP regressor with sample set (MLE).

(3) Prediction

(4) Acquisition of next sample with maximum prediction variance

(5) Repeat (2-4) until termination criteria is met

RQ_2: Adaptive Learning with Gaussian Processes (GP)



(1) Initial random sampling of k revisions.

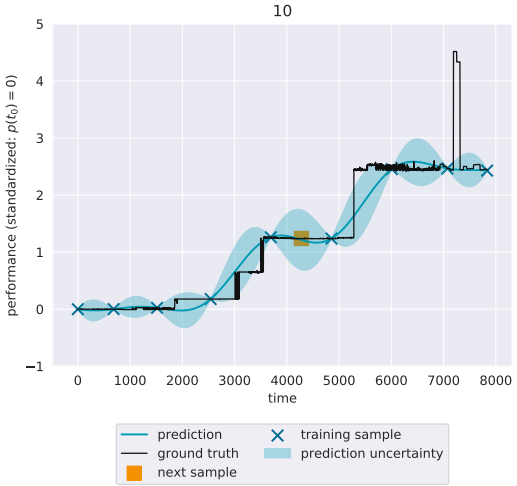
(2) Training of GP regressor with sample set (MLE).

(3) Prediction

(4) Acquisition of next sample with maximum prediction variance

(5) Repeat (2-4) until termination criteria is met

RQ_2: Adaptive Learning with Gaussian Processes (GP)



(1) Initial random sampling of k revisions.

(2) Training of GP regressor with sample set (MLE).

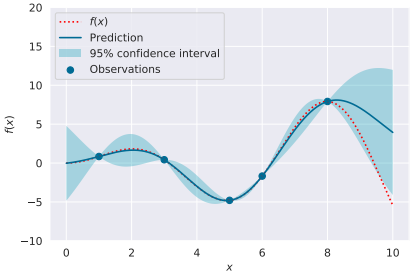
(3) Prediction

(4) Acquisition of next sample with maximum prediction variance

(5) Repeat (2-4) until termination criteria is met

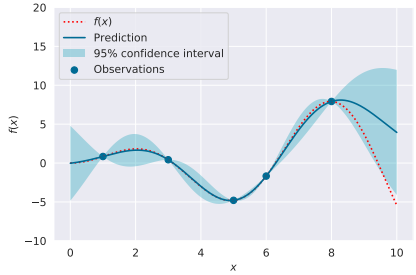
RQ_2: Evaluation Plan

- Which acquisition strategy minimizes prediction error?
  - Uncertainty-aware [Roberts et al., 2012]
  - Bisection/binary search [Heger et al., 2013]
  - Random/uniform sampling as baseline



RQ_2: Evaluation Plan

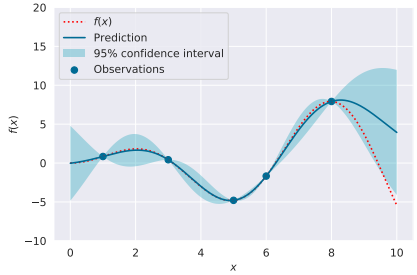
- Which acquisition strategy minimizes prediction error?
  - Uncertainty-aware [Roberts et al., 2012]
  - Bisection/binary search [Heger et al., 2013]
  - Random/uniform sampling as baseline



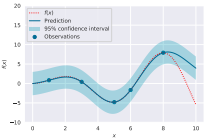
- Termination criteria: How robust is adaptive learning?
  - Uniform prediction variance threshold

RQ_2: Evaluation Plan

- Which acquisition strategy minimizes prediction error?
  - Uncertainty-aware [Roberts et al., 2012]
  - Bisection/binary search [Heger et al., 2013]
  - Random/uniform sampling as baseline

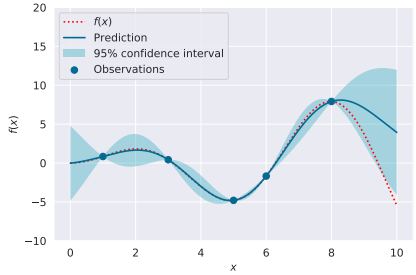


- Termination criteria: How robust is adaptive learning?
  - Uniform prediction variance threshold

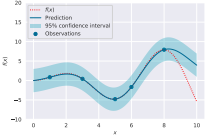


RQ_2: Evaluation Plan

- Which acquisition strategy minimizes prediction error?
  - Uncertainty-aware [Roberts et al., 2012]
  - Bisection/binary search [Heger et al., 2013]
  - Random/uniform sampling as baseline

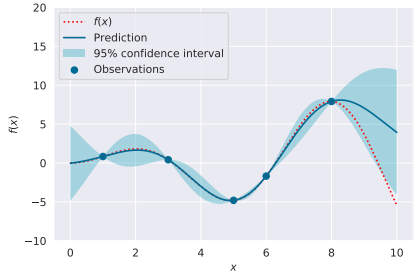


- Termination criteria: How robust is adaptive learning?
  - Uniform prediction variance threshold
  - Minimum variance-change threshold

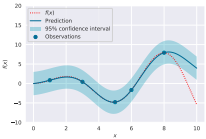


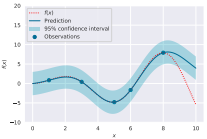
RQ_2: Evaluation Plan

- Which acquisition strategy minimizes prediction error?
  - Uncertainty-aware [Roberts et al., 2012]
  - Bisection/binary search [Heger et al., 2013]
  - Random/uniform sampling as baseline

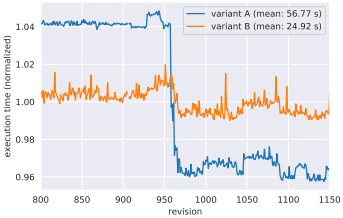


- Termination criteria: How robust is adaptive learning?
  - Uniform prediction variance threshold
  - Minimum variance-change threshold

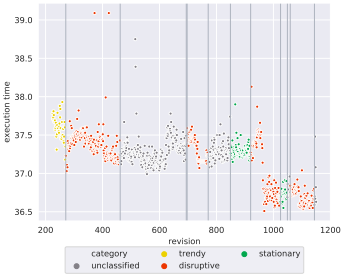




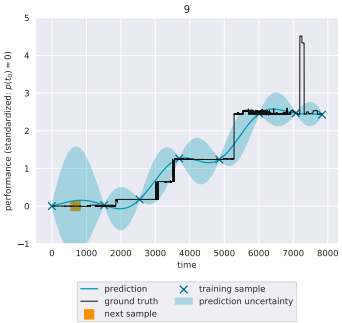
Summary



performance evolves heterogeneously



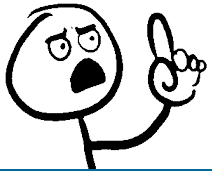
classification of performance-changes



active-learning and estimation of performance evolution

Thank you for your kind attention!

Any questions or suggestions?



References

Han, X. and Yu, T. (2016).
An Empirical Study on Performance Bugs for Highly Configurable Software Systems.
In *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, ESEM '16, pages 23:1–23:10, New York, NY, USA. ACM.

Heger, C., Happe, J., and Farahbod, R. (2013).
Automated root cause isolation of performance regressions during software development.
page 27. ACM Press.

Roberts, S., Osborne, M., Ebden, M., Reece, S., Gibson, N., and Aigrain, S. (2012).
Gaussian processes for time-series modelling.
Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 371(1984):20110550–20110550.

Backup: Performance Assessment

¯_(ツ)_/¯

Backup: Case Study Metrics

¯_(ツ)_/¯