R244 - CHERRYPICK

{CherryPick}: **Adaptively unearthing** the best cloud configurations for big data analytics [PDF] usenix.org

O Alipourfard, HH Liu, J Chen… - … USENIX Symposium on …, 2017 - usenix.org

… **CherryPick** with Ernest [37] and show how **CherryPick** … **CherryPick**, a service that selects nearoptimal cloud configurations with high accuracy and low overhead. **CherryPick adaptively** …

Save Cite Cited by 631 Related articles All 21 versions

R244 - Chris Tomy - 2024/11/20

PROBLEM SETUP

- Big data applications (Spark, Hadoop, etc.)
- Goal: find the best cloud configuration given a budget ($$$)

FORMULATION

Cloud configuration: vector $\vec{x}$.

Example components of this vector:

- Number of VMs
- CPU speed (e.g. 2 GHz)
- etc.

(with caveats!)

For a workload w:

- $P(\vec{x})$ is price (per unit time)
- $T(\vec{x}; w)$ is the running time
- $C(\vec{x}; w) = P(\vec{x}) \times T(\vec{x}; w)$

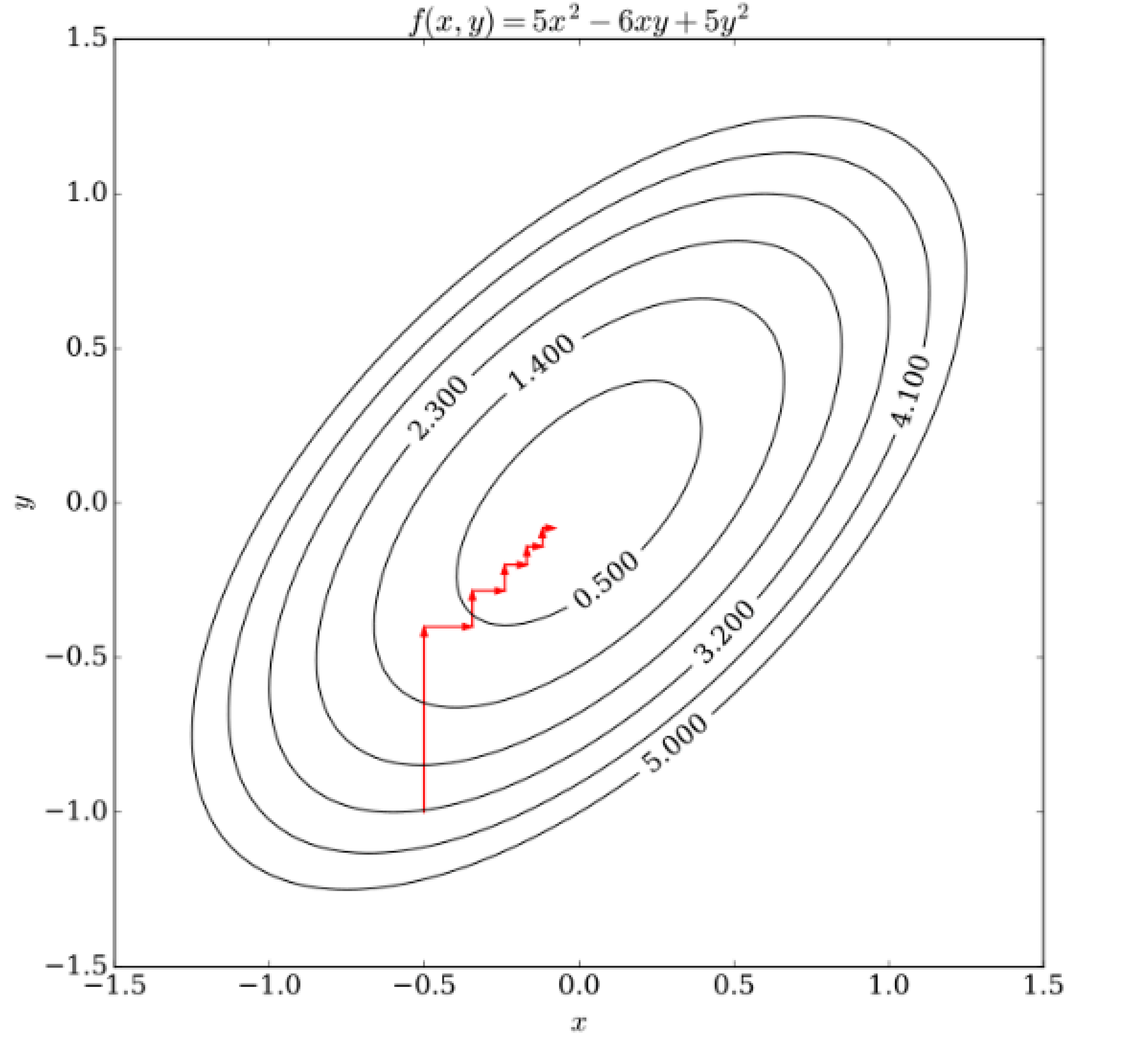
$$\min_{\vec{x}} C(\vec{x}; w)$$

subject to some constraints e.g. max running time

$$T(\vec{x}) \leq T_m$$

Search-based (over $\vec{x}$)

- Brute-force
- Random search with budget
- Co-ordinate descent


$f(x, y) = 5x^2 - 6xy + 5y^2$
2.300
1.400
0.500
3.200
4.100
5.000
x
y
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5

Modelling:

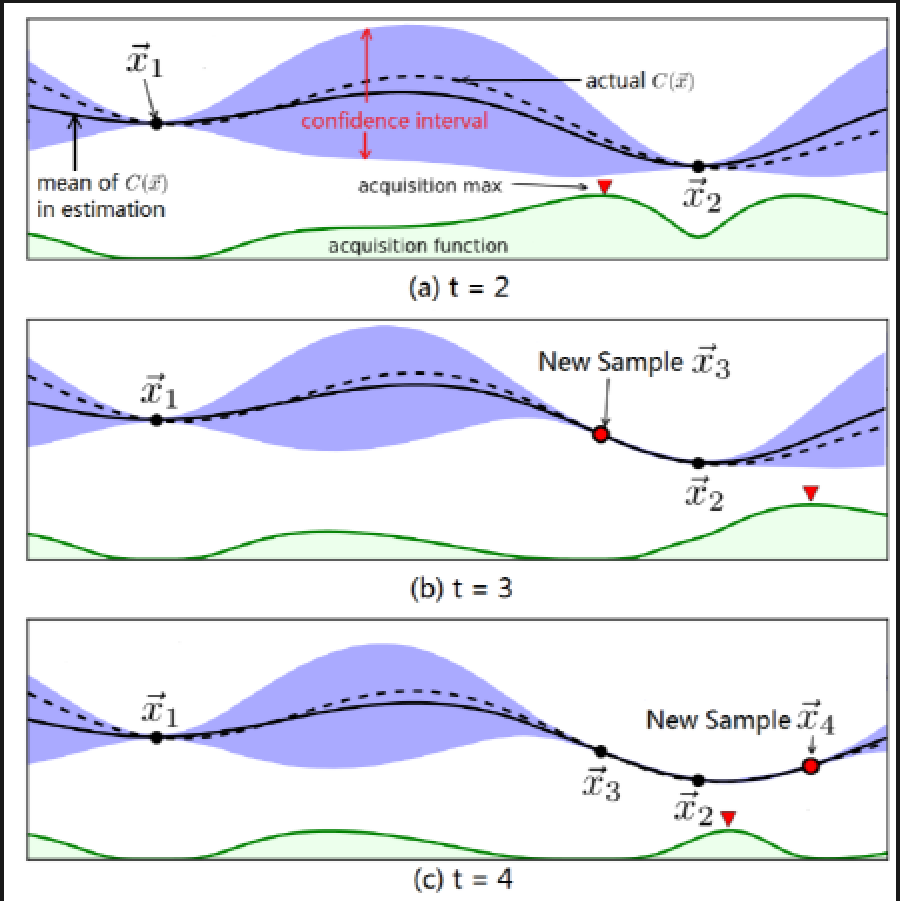
- Parametric modelling: Ernest
  - $t = \theta_0 + \theta_1 f_1 + \theta_2 f_2 + \ldots$
  - Hand-chosen features f_i
- Non-parametric modelling: GPs

BAYESIAN OPTIMISATION REFRESHER

- Choose prior: $\mathbf{f} \sim N(\mu, K)$
- Random starting points: D
- Iteratively:
  - Compute predictive posterior $p(\mathbf{f}_\star | D)$
  - Pick $\vec{x}$ where $\max \alpha(\vec{x})$
  - Update D

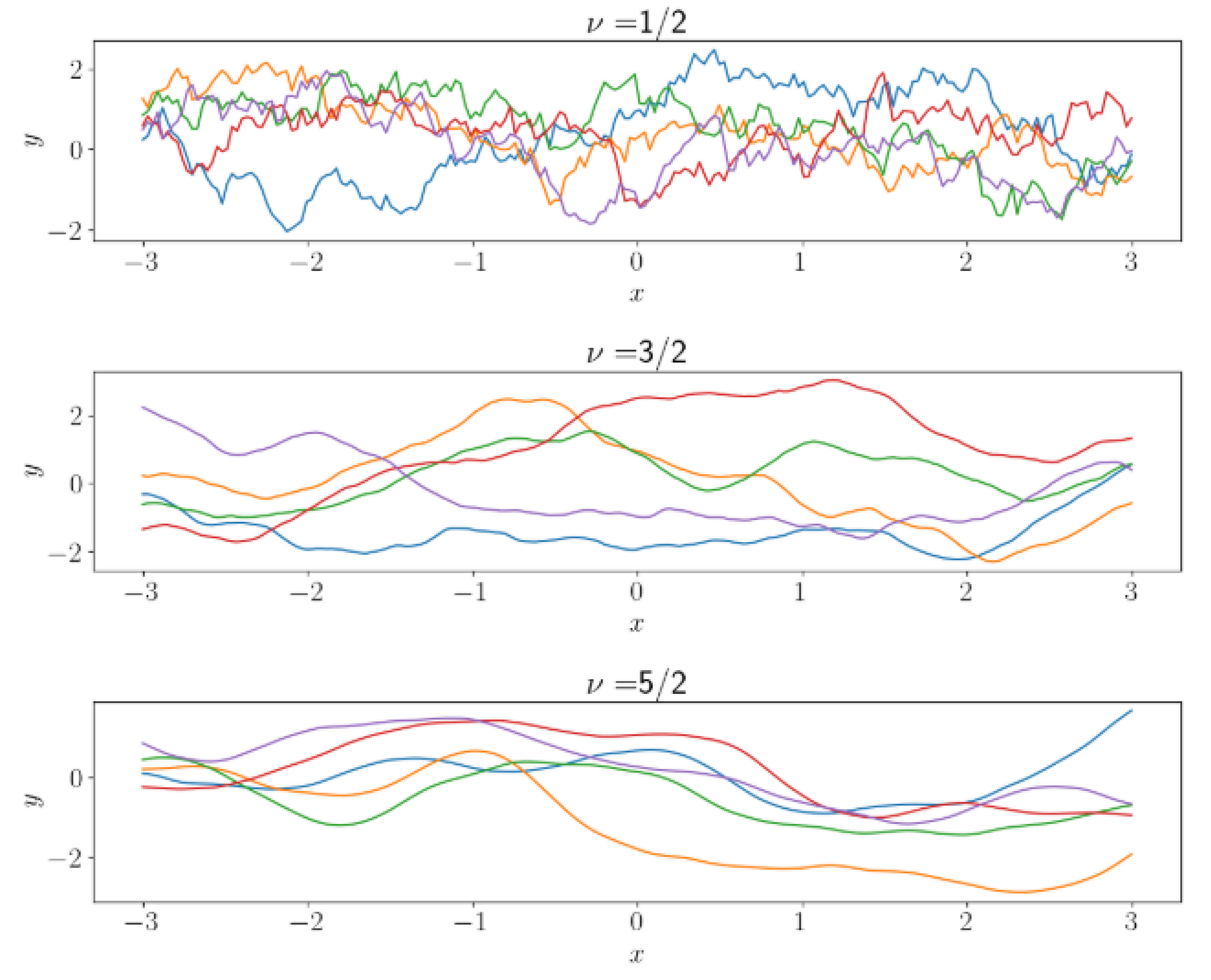
CHERRYPICK'S APPROACH

Search configuration space $\vec{x}$ with BO.



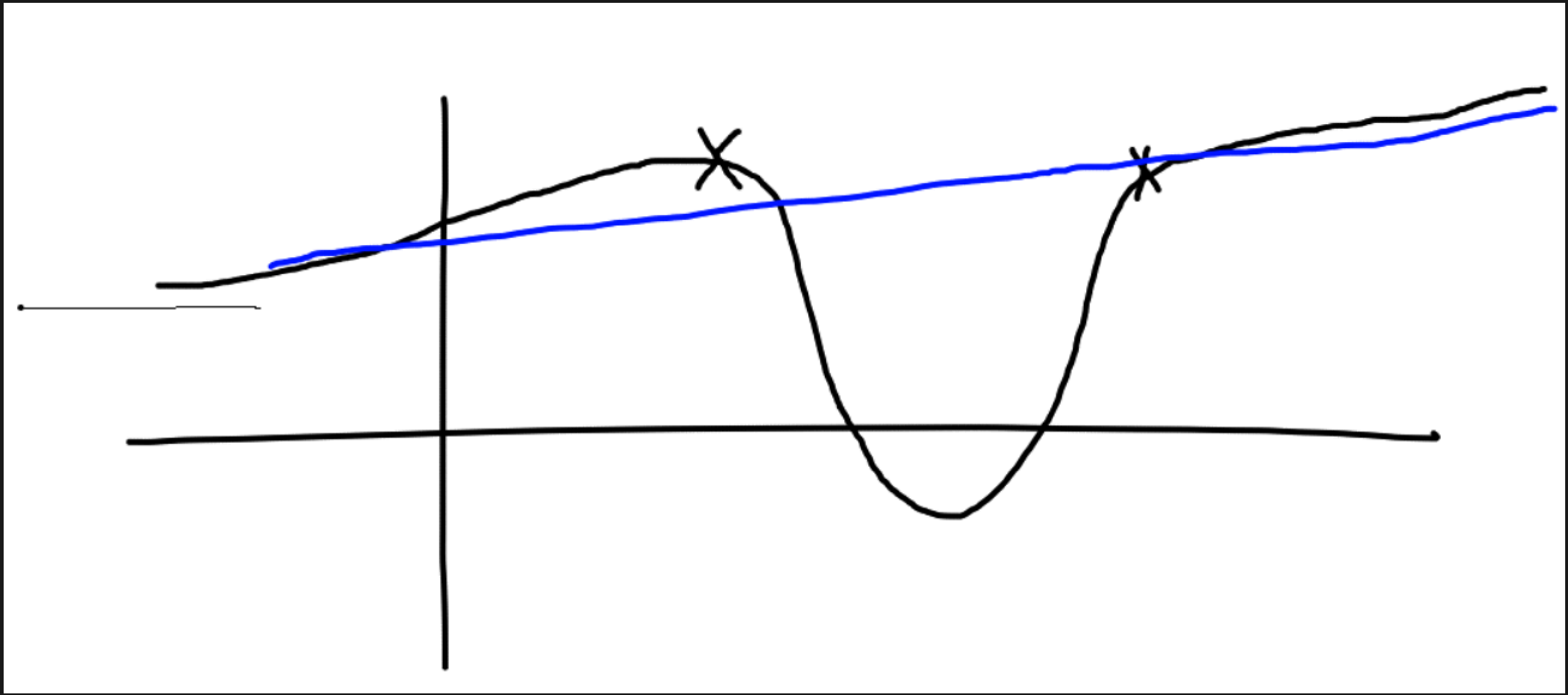
COVARIANCE KERNEL

- GP prior: Matérn5/2 kernel
- $k(\vec{x}_i, \vec{x}_j; \sigma^2, \nu)$



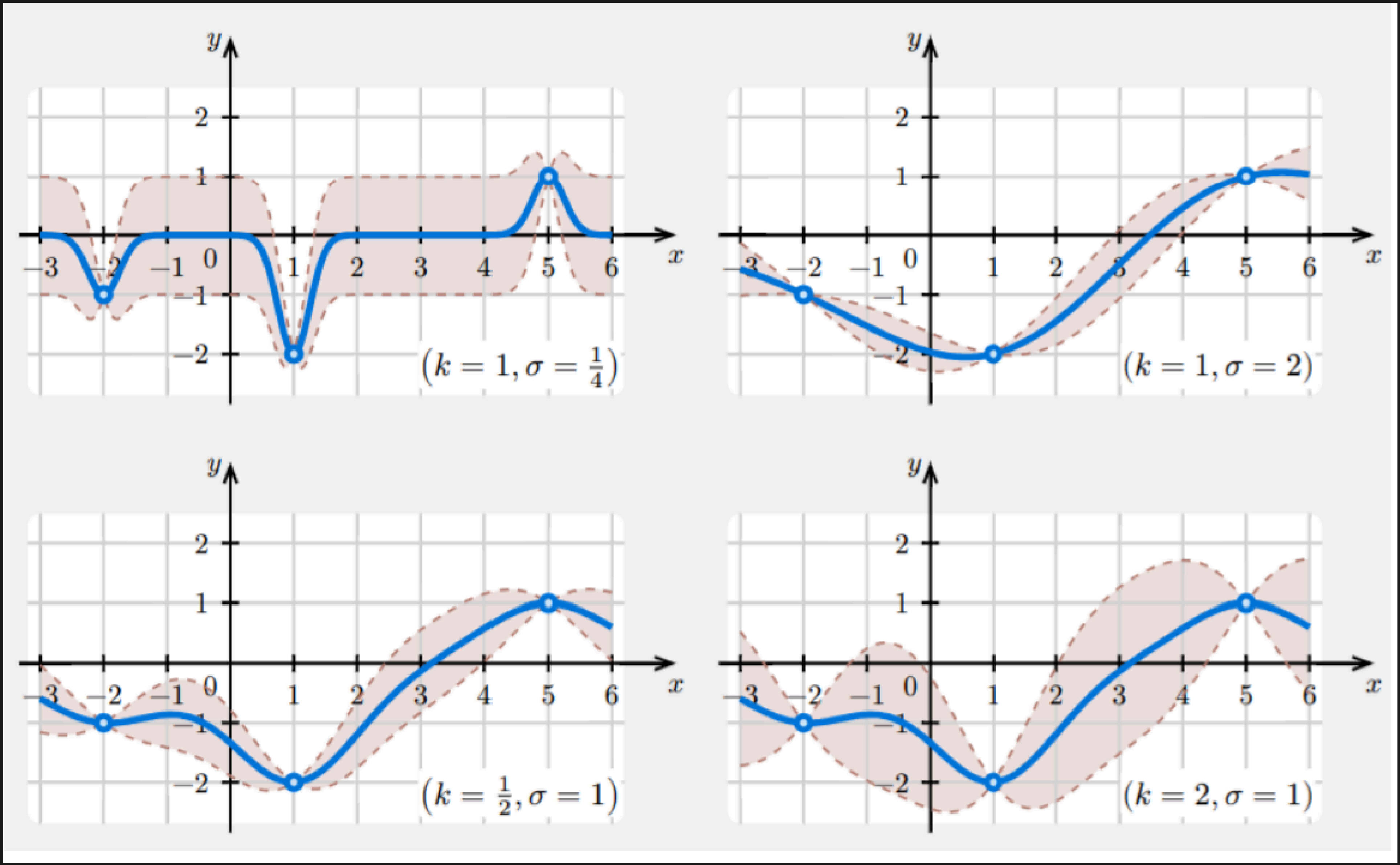
IMPORTANCE OF COVARIANCE

Sufficiently bad prior + acquisition can make convergence impossible!



Blue: surrogate model. Black: true function.

RBF hyperparameters example:



Confidence intervals vary greatly.

ACQUISITION FUNCTION - MODIFIED EI

- Expected improvement
  - $\alpha(\vec{x}) = E[u(\vec{x})|\vec{x}, D]$, for $u(\vec{x}) = f(\vec{x}_\star) - f(\vec{x})$
  - Closed form: $(f_\star - \mu(\vec{x}))\Phi(Z) + \sigma(\vec{x})\phi(Z)$
- Modified:
  - $EI'(\vec{x}) = P[T(\vec{x}) \leq T_m] \times EI(\vec{x})$

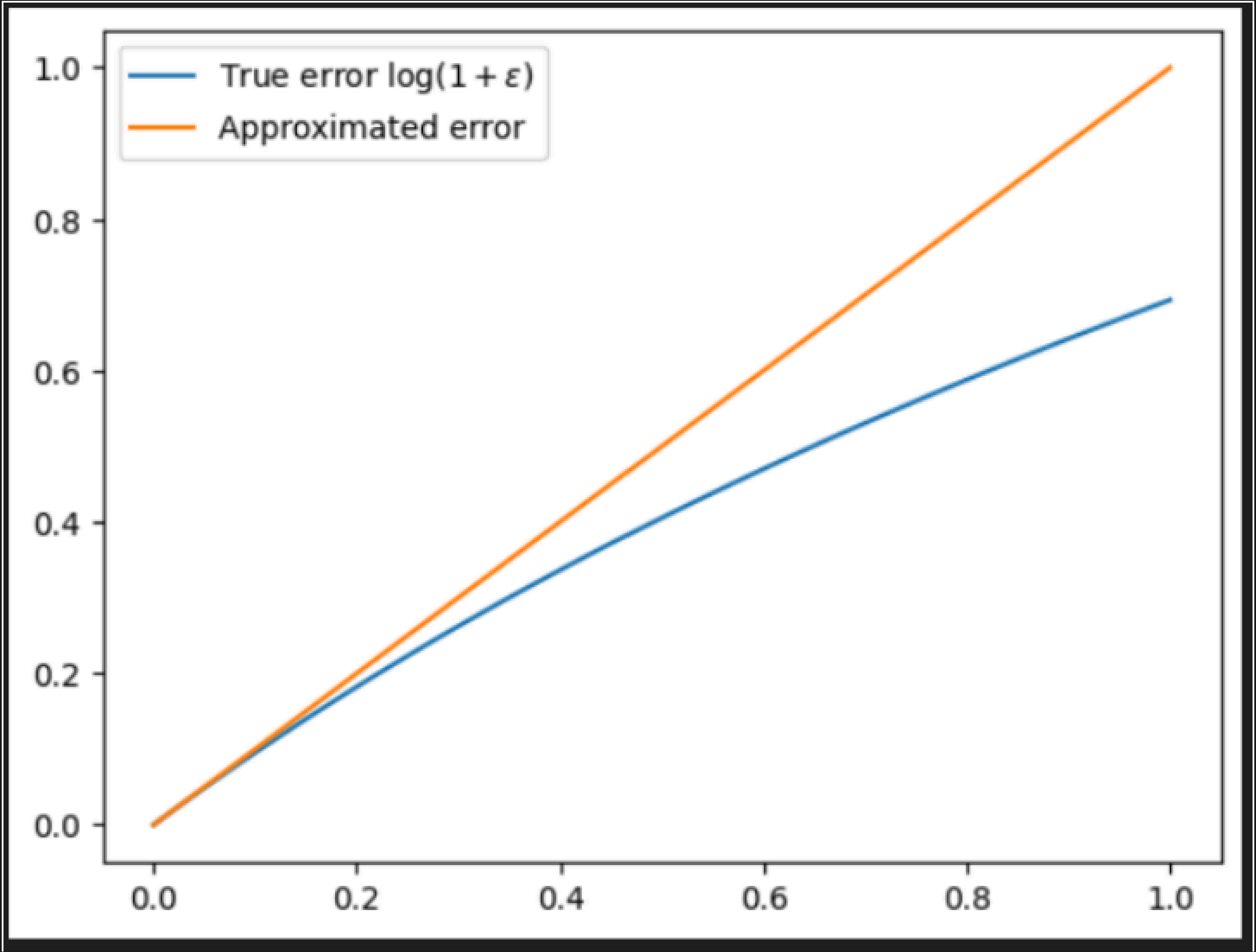
DISCRETIZED FEATURES

To reduce search space:

- Continuous features (1.2 GHz, 3.3 GHz) $\longrightarrow$ discrete (slow, fast)
- Viable due to closed form: $(f_\star - \mu(\vec{x}))\Phi(Z) + \sigma(\vec{x})\phi(Z)$

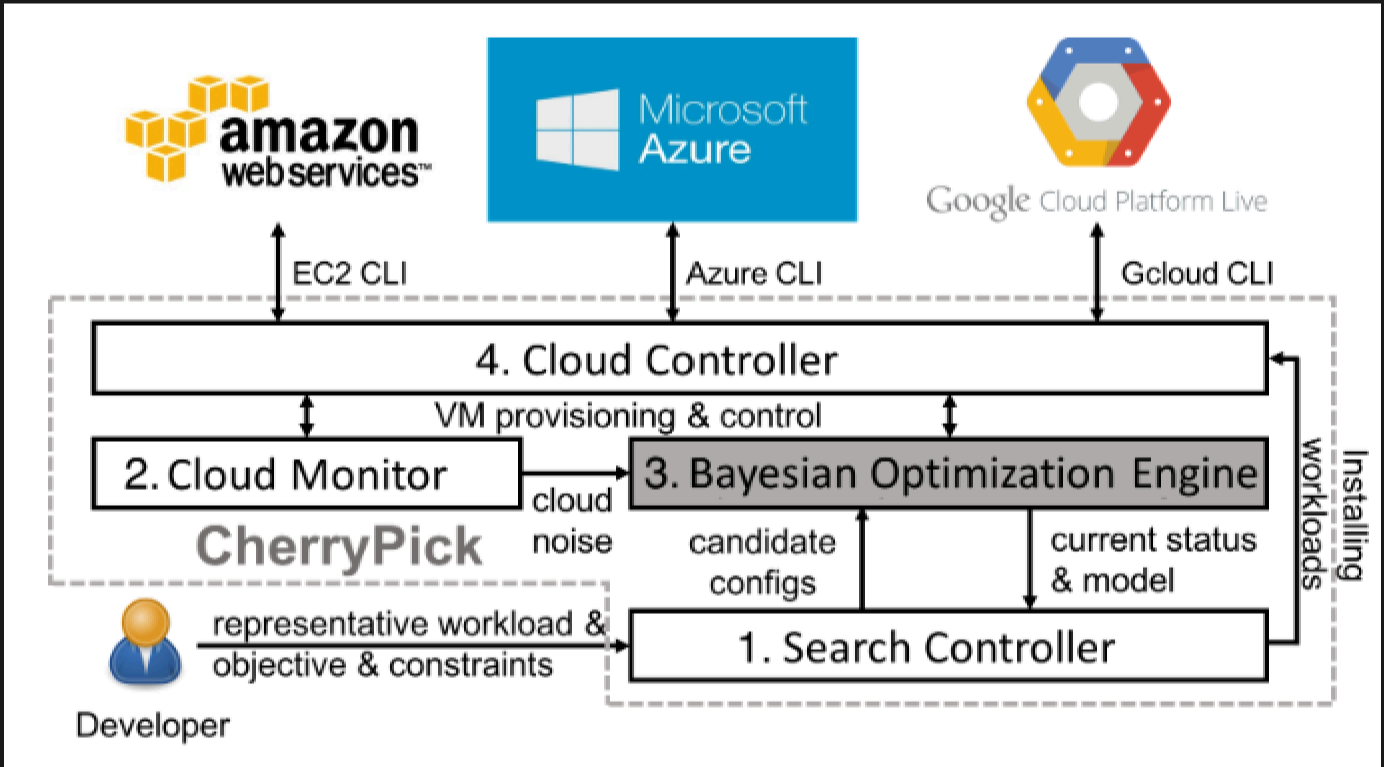
DEALING WITH NOISE

- Multiplicative noise
  - Observation: $C(\vec{x})(1 + \epsilon)$
  - $\log C(\vec{x}) + log(1 + \epsilon) \approx \log C(\vec{x}) + \epsilon$
- Presumably, added as $k(x, x) + \sigma_\epsilon^2 I$ in BO engine



True error $\log(1+\varepsilon)$
Approximated error
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

IMPLEMENTATION

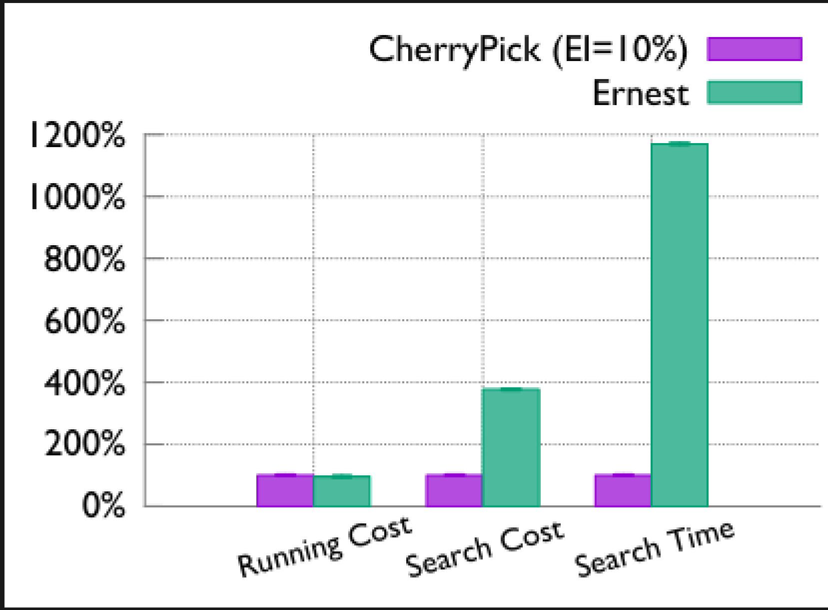


RESULTS

"CherryPick has a 45-90% chance to find optimal configurations"

CONTRAST W/ ERNEST

Big wins in search time:



Random search is surprisingly good.

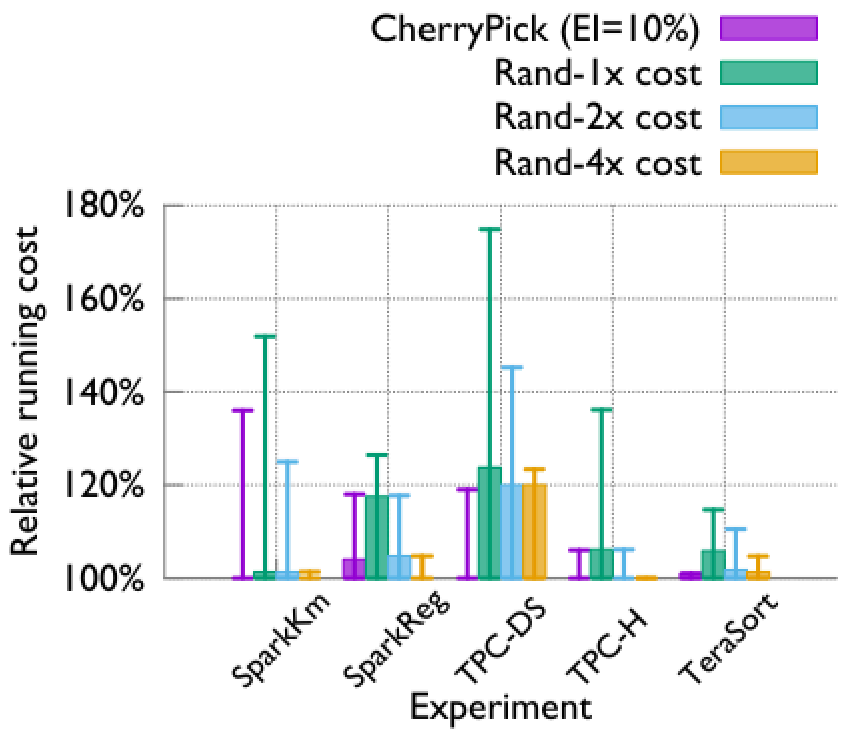


Figure 8: Running cost of configurations by *CherryPick* and random search. The bars show 10th and 90th percentile.

Strengths

- More sample efficient than other approaches
  - Consistently low search cost.
- More adaptable than Ernest
  - More flexible (to VM instance types)
- Tunable option to trade-off search cost and accuracy: EI threshold

Weaknesses

- Assumes a specific workload
- Not significantly better than random search
- Weak/lacking justification of modelling choices
    - No mention of hyperparameter search or fitting, e.g. MLE
    - Multiplicative noise, what if $0.2 < \epsilon < 1.0$?
- Up to 9x less search cost than *exhaustive* search, but is that good enough?

Contrast to:

Practical bayesian optimization of machine learning algorithms [PDF] neurips.cc

J Snoek, H Larochelle… - Advances in neural …, 2012 - proceedings.neurips.cc

The use of machine learning algorithms frequently involves careful tuning of learning parameters and model hyperparameters. Unfortunately, this tuning is often a “black art” requiring expert experience, rules of thumb, or sometimes brute-force search. There is therefore great appeal for automatic approaches that can optimize the performance of any given learning algorithm to the problem at hand. In this work, we consider this problem through the framework of Bayesian optimization, in which a learning algorithm's …

Great discussion on kernel hyperparameter choices.