Model-Based Algorithm Configuration

Guest lecture in CPSC 536H - Empirical Algorithmics

Frank Hutter
Postdoctoral fellow, UBC CS

March 18, 2010

Motivation 1: Algorithm Configuration

Most algorithms have parameters

- Decisions that are left open during algorithm design
  - numerical parameters (*e.g.*, real-valued thresholds)
  - categorical parameters (*e.g.*, which heuristic to use)

Motivation 1: Algorithm Configuration

Most algorithms have parameters

- Decisions that are left open during algorithm design
  - numerical parameters (*e.g.*, real-valued thresholds)
  - categorical parameters (*e.g.*, which heuristic to use)
- Provide flexibility
- Instantiate to optimize empirical performance

Motivation 1: Algorithm Configuration

Most algorithms have parameters

- Decisions that are left open during algorithm design
  - numerical parameters (*e.g.*, real-valued thresholds)
  - categorical parameters (*e.g.*, which heuristic to use)
- Provide flexibility
- Instantiate to optimize empirical performance

Automated Approaches for Parameter Optimization

- Eliminate most tedious part of algorithm design and end use
- Can **generate custom algorithms** for different problem types
- Save development time & improve performance

Motivation 2: Model-Based Approaches

Model-free techniques are limited

- Only return a good parameter setting
- Do not provide additional information
  - How important is each of the parameters?
  - Which parameters interact?
  - For which types of instances is a parameter setting good?

Motivation 2: Model-Based Approaches

Model-free techniques are limited

- Only return a good parameter setting
- Do not provide additional information
  - How important is each of the parameters?
  - Which parameters interact?
  - For which types of instances is a parameter setting good?

Model-based approaches can help

- Construct *response surface model*
  - predictive model of algorithm performance

Motivation 2: Model-Based Approaches

Model-free techniques are limited

- Only return a good parameter setting
- Do not provide additional information
  - How important is each of the parameters?
  - Which parameters interact?
  - For which types of instances is a parameter setting good?

Model-based approaches can help

- Construct *response surface model*
  - predictive model of algorithm performance
- Use model to answer the questions above
  - $\rightsquigarrow$ Inform algorithm designer

Motivation 2: Model-Based Approaches

Model-free techniques are limited

- Only return a good parameter setting
- Do not provide additional information
  - How important is each of the parameters?
  - Which parameters interact?
  - For which types of instances is a parameter setting good?

Model-based approaches can help

- Construct *response surface model*
  - predictive model of algorithm performance
- Use model to answer the questions above
  - $\rightsquigarrow$ Inform algorithm designer
- Use model for algorithm configuration

Outline

1. Predictive Models of Algorithm Performance
2. Sequential Model-Based Optimization
3. Summary

Outline

1. Predictive Models of Algorithm Performance
2. Sequential Model-Based Optimization
3. Summary

Models of algorithm performance: basics

Data: algorithm performance in previous algorithm runs

- Parameter settings $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n$, $\boldsymbol{\theta}_i \in \boldsymbol{\Theta}$
- Observed algorithm performances $y_1, \ldots, y_n$, $y_i \in \mathbb{R}$
- For now: assume just a single instance

Models of algorithm performance: basics

Data: algorithm performance in previous algorithm runs

- Parameter settings $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n$, $\boldsymbol{\theta}_i \in \boldsymbol{\Theta}$
- Observed algorithm performances $y_1, \ldots, y_n$, $y_i \in \mathbb{R}$
- For now: assume just a single instance

Offline model training

- Learn a function $f : \boldsymbol{\Theta} \rightarrow \mathbb{R}$
- To minimize a loss function, such as $\sum_{i=1}^{n}(y_i - f(\boldsymbol{\theta}_i))^2$

Models of algorithm performance: basics

Data: algorithm performance in previous algorithm runs

- Parameter settings $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n$, $\boldsymbol{\theta}_i \in \boldsymbol{\Theta}$
- Observed algorithm performances $y_1, \ldots, y_n$, $y_i \in \mathbb{R}$
- For now: assume just a single instance

Offline model training

- Learn a function $f : \boldsymbol{\Theta} \rightarrow \mathbb{R}$
- To minimize a loss function, such as $\sum_{i=1}^{n} (y_i - f(\boldsymbol{\theta}_i))^2$

Performance prediction for new algorithm run

- Given a new configuration $\boldsymbol{\theta}_{i+1}$
- Predict performance as $f(\boldsymbol{\theta}_{i+1})$

Models of algorithm performance: which machine learning model to use?

Typical types of models used

- Linear regression
- Gaussian process (GP) regression
- Regression trees
- Random forests (forests of regression trees)

Models of algorithm performance: which machine learning model to use?

Typical types of models used

- Linear regression
- Gaussian process (GP) regression
- Regression trees
- Random forests (forests of regression trees)

Requirements in the context of algorithm configuration

- Handle many data points
- Handle mixed continuous/discrete parameters
- Quantify uncertainty of predictions

Outline

Picking the next configuration with the model

Balance *exploration* and *exploitation*

- High predicted variance is good (exploration)
- Low predicted mean is good (exploitation)

Picking the next configuration with the model

Balance *exploration* and *exploitation*

- High predicted variance is good (exploration)
- Low predicted mean is good (exploitation)

E.g. probability of improvement

- Pr($\boldsymbol{\theta}$ is better than incumbent)
- Closed form expression for Gaussian predictive distribution

Picking the next configuration with the model

Balance *exploration* and *exploitation*

- High predicted variance is good (exploration)
- Low predicted mean is good (exploitation)

E.g. probability of improvement

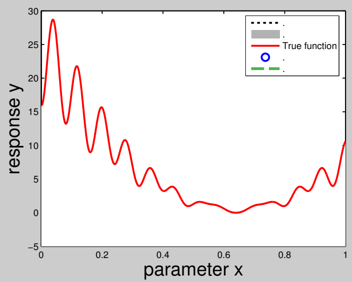
- $\Pr(\boldsymbol{\theta}$ is better than incumbent$)$
- Closed form expression for Gaussian predictive distribution

E.g. expected improvement

- $\mathbb{E}_{cost(\theta)}[max(0, cost(incumbent) - cost(\theta))]$
- Closed form expression for Gaussian predictive distribution
- Also for Gaussian predictive distribution in log space

Sequential Model-Based Optimization (Vanilla)

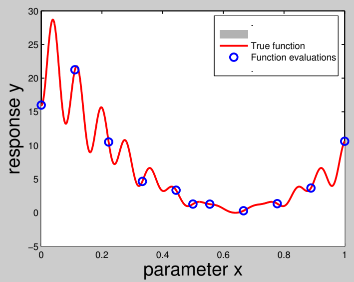
Blackbox function optimization; function = algo. performance



Sequential Model-Based Optimization (Vanilla)

Blackbox function optimization; function = algo. performance

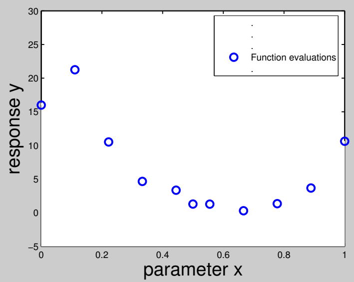
0. Run algorithm with initial parameter settings



Sequential Model-Based Optimization (Vanilla)

Blackbox function optimization; function = algo. performance

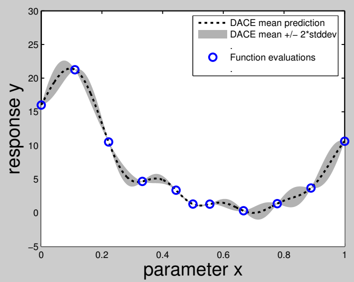
0. Run algorithm with initial parameter settings



Sequential Model-Based Optimization (Vanilla)

Blackbox function optimization; function = algo. performance

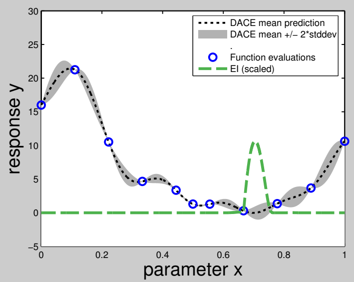
0. Run algorithm with initial parameter settings
1. Fit a model to the data



Sequential Model-Based Optimization (Vanilla)

Blackbox function optimization; function = algo. performance

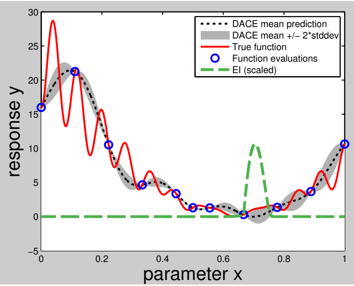
0. Run algorithm with initial parameter settings
1. Fit a model to the data
2. Use model to pick promising parameter setting (EIC)



Sequential Model-Based Optimization (Vanilla)

Blackbox function optimization; function = algo. performance

0. Run algorithm with initial parameter settings
1. Fit a model to the data
2. Use model to pick promising parameter setting (EIC)
3. Perform an algorithm run with that parameter setting

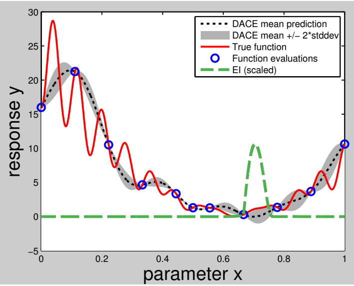


Sequential Model-Based Optimization (Vanilla)

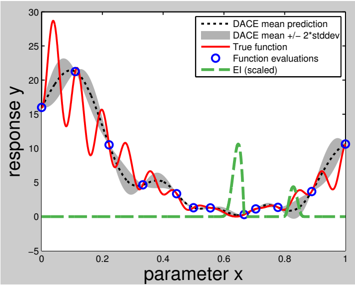
Blackbox function optimization; function = algo. performance

0. Run algorithm with initial parameter settings
1. Fit a model to the data
2. Use model to pick promising parameter setting (EIC)
3. Perform an algorithm run with that parameter setting

- Repeat 1-3 until time is up



First step



Second step

General Algorithm Framework: Sequential Model-Based Optimization

$[\mathbf{R}, \theta_{inc}] \leftarrow$ Initialize$()$

General Algorithm Framework: Sequential Model-Based Optimization

$[\mathbf{R}, \theta_{inc}] \leftarrow \text{Initialize}()$

$\mathcal{M} \leftarrow \text{FitModel}(\mathbf{R})$

General Algorithm Framework: Sequential Model-Based Optimization

$[\mathbf{R}, \theta_{inc}] \leftarrow$ Initialize()

$\mathcal{M} \leftarrow$ FitModel($\mathbf{R}$)
$\vec{\boldsymbol{\Theta}}_{new} \leftarrow$ SelectNewParameterSettings($\mathcal{M}$, θ_{inc})

General Algorithm Framework: Sequential Model-Based Optimization

$[\mathbf{R}, \theta_{inc}] \leftarrow$ Initialize()

$\mathcal{M} \leftarrow$ FitModel($\mathbf{R}$)
$\vec{\boldsymbol{\Theta}}_{new} \leftarrow$ SelectNewParameterSettings($\mathcal{M}$, θ_{inc})
$[\mathbf{R}, \theta_{inc}] \leftarrow$ Intensify($\vec{\boldsymbol{\Theta}}_{new}$, θ_{inc}, $\mathbf{R}$)

General Algorithm Framework: Sequential Model-Based Optimization

$[\mathbf{R}, \theta_{inc}] \leftarrow$ Initialize()

repeat

$\quad \mathcal{M} \leftarrow$ FitModel($\mathbf{R}$)

$\quad \vec{\mathbf{\Theta}}_{new} \leftarrow$ SelectNewParameterSettings($\mathcal{M}$, θ_{inc})

$\quad [\mathbf{R}, \theta_{inc}] \leftarrow$ Intensify($\vec{\mathbf{\Theta}}_{new}$, θ_{inc}, $\mathbf{R}$)

until *time budget exhausted*

General Algorithm Framework: Sequential Model-Based Optimization

$[\mathbf{R},\ \theta_{inc}] \leftarrow$ Initialize()

repeat

$\mathcal{M} \leftarrow$ FitModel($\mathbf{R}$)

$\vec{\boldsymbol{\Theta}}_{new} \leftarrow$ SelectNewParameterSettings($\mathcal{M}$, θ_{inc})

$[\mathbf{R}, \theta_{inc}] \leftarrow$ Intensify($\vec{\boldsymbol{\Theta}}_{new}$, θ_{inc}, $\mathbf{R}$)

until *time budget exhausted*

return θ_{inc}

Sequential Model-Based Optimization: roots

Experimental design literature in statistics

- Expected improvement [Mockus et al., 1978]

Sequential Model-Based Optimization: roots

Experimental design literature in statistics

- Expected improvement [Mockus et al., 1978]
- Efficient Global Optimization (EGO) [Jones et al., 1998]
  - Optimization of expensive blackbox functions without noise
  - Popularized the approach

Sequential Model-Based Optimization: roots

Experimental design literature in statistics

- Expected improvement [Mockus et al., 1978]
- Efficient Global Optimization (EGO) [Jones et al., 1998]
  - Optimization of expensive blackbox functions without noise
  - Popularized the approach
- Sequential Kriging Optimization [Huang et al., 2006]
  - Also allowed noise

Sequential Model-Based Optimization: adaptation for optimizing algorithms

- Sequential Parameter Optimization (SPO)
  [Bartz-Beielstein et al., '05-present]
  - SPO toolbox
  - Set of interactive tools for parameter optimization

Sequential Model-Based Optimization: adaptation for optimizing algorithms

- Sequential Parameter Optimization (SPO)
  [Bartz-Beielstein et al., '05-present]
  - SPO toolbox
  - Set of interactive tools for parameter optimization
- More robust completely automated tool [Hutter et al, GECCO-09]
  - Studied SPO components
  - How many runs to perform for each $\boldsymbol{\theta}$
    - "Intensification mechanism" inspired by FocusedILS

    $\rightsquigarrow$: SPO^{+}

Sequential Model-Based Optimization: adaptation for optimizing algorithms

- Sequential Parameter Optimization (SPO)
  [Bartz-Beielstein et al., '05-present]
  - SPO toolbox
  - Set of interactive tools for parameter optimization
- More robust completely automated tool [Hutter et al, GECCO-09]
  - Studied SPO components
  - How many runs to perform for each $\boldsymbol{\theta}$
    - "Intensification mechanism" inspired by FocusedILS

    $\rightsquigarrow$: SPO^+
- Time-Bounded SPO [Hutter et al, LION-10]
  - Reduced computational overheads due to the model
  - Removed need for costly initial design

Sequential Model-Based Optimization: algorithm configuration

- Categorical parameters [Hutter, PhD thesis '09; in preparation for CP-10]
  - Different kernel for Gaussian processes
  - Random forest model

Sequential Model-Based Optimization: algorithm configuration

- Categorical parameters [Hutter, PhD thesis '09; in preparation for CP-10]
  - Different kernel for Gaussian processes
  - Random forest model
- Multiple benchmark instances
  [Hutter, PhD thesis '09; in preparation for CP-10]
  - Include *instance features* in the model
  - Predict *marginal performance* across the training instances

Sequential Model-Based Optimization: algorithm configuration

- Categorical parameters [Hutter, PhD thesis '09; in preparation for CP-10]
  - Different kernel for Gaussian processes
  - Random forest model
- Multiple benchmark instances
  [Hutter, PhD thesis '09; in preparation for CP-10]
  - Include *instance features* in the model
  - Predict *marginal performance* across the training instances

$\rightsquigarrow$ ActiveConfigurator 1.0

Sequential Model-Based Optimization: performance

Optimizing algorithms for single instances

- Outperforms FocusedILS in most cases
- Is more robust than FocusedILS
- No need to discretize continuous parameters

Sequential Model-Based Optimization: performance

Optimizing algorithms for single instances

- Outperforms FocusedILS in most cases
- Is more robust than FocusedILS
- No need to discretize continuous parameters

Optimizing algorithms for multiple instances

- Performed somewhat better than FocusedILS
- But need to perform more comparisons

Sequential Model-Based Optimization: issues yet to be addressed

- Capping (as in ParamILS)
  - Tricky for learning: if i-th run capped $\rightarrow$ y_i is only lower bound

Sequential Model-Based Optimization: issues yet to be addressed

- Capping (as in ParamILS)
  - Tricky for learning: if i-th run capped $\rightarrow$ y_i is only lower bound
- Conditional parameters
  - Tricky to exploit in learning

Sequential Model-Based Optimization: issues yet to be addressed

- Capping (as in ParamILS)
  - Tricky for learning: if i-th run capped $\rightarrow$ y_i is only lower bound
- Conditional parameters
  - Tricky to exploit in learning
- Use of model for
  - Active selection of instances
  - Active selection of captimes

Model-free *vs* model-based

- Advantages of model-free approach
  - Conceptual simplicity
  - Simple to integrate adaptive capping
  - Simple to integrate conditional parameters
  - Implementation robustness (less things can break)

Model-free *vs* model-based

- Advantages of model-free approach
  - Conceptual simplicity
  - Simple to integrate adaptive capping
  - Simple to integrate conditional parameters
  - Implementation robustness (less things can break)
- Advantages of model-based approach
  - Can interpolate & extrapolate
  - Can handle continuous parameters
  - Enable future, more sophisticated techniques
    - Active selection of most informative instance
    - Active selection of cutoff time
    - Per-instance approaches

Outline

1. Predictive Models of Algorithm Performance

2. Sequential Model-Based Optimization

3. Summary

Summary

Predictive Models of Algorithm Performance

- Are learned from previously gathered performance data
- Map from a parameter setting to the predicted performance

Summary

Predictive Models of Algorithm Performance

- Are learned from previously gathered performance data
- Map from a parameter setting to the predicted performance

Sequential Model-Based Optimization (SMBO)

- Iteratively selects promising parameter configuration
- Updates the model on the fly

Summary

Predictive Models of Algorithm Performance

- Are learned from previously gathered performance data
- Map from a parameter setting to the predicted performance

Sequential Model-Based Optimization (SMBO)

- Iteratively selects promising parameter configuration
- Updates the model on the fly

Existing Extensions

- Handle noise better: intensification mechanism
- Keep computational overhead at bay
- Outperform ParamILS for single instances