AutoML: Bayesian Optimization for HPO

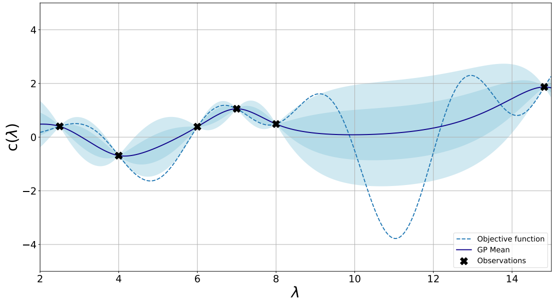
Surrogate Models

Bernd Bischl Frank Hutter Lars Kotthoff
Marius Lindauer Joaquin Vanschoren

Desiderata for Surrogate Models in Bayesian Optimization

In all cases

- Regression model with uncertainty estimates
- Accurate predictions



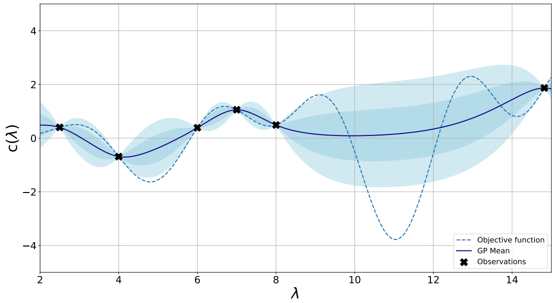
Desiderata for Surrogate Models in Bayesian Optimization

In all cases

- Regression model with uncertainty estimates
- Accurate predictions

Depending on the application

- Is cheap to train
- Scales well in the number of data points
- Scales well in the number of dimensions
- Can handle different types of inputs (categorical and continuous)



Overview of the Surrogate Models We'll Discuss

- Gaussian Processes
- Random Forests
- Bayesian Neural Networks

Gaussian Processes (GPs): Reminder of Pros and Cons

Advantages

- Smooth and reliable uncertainty estimates
- Strong sample efficiency
- We can encode expert knowledge about the design space in the kernel

Gaussian Processes (GPs): Reminder of Pros and Cons

Advantages

- Smooth and reliable uncertainty estimates
- Strong sample efficiency
- We can encode expert knowledge about the design space in the kernel

These advantages make GPs the most commonly-used model in Bayesian optimization

Gaussian Processes (GPs): Reminder of Pros and Cons

Advantages

- Smooth and reliable uncertainty estimates
- Strong sample efficiency
- We can encode expert knowledge about the design space in the kernel

These advantages make GPs the most commonly-used model in Bayesian optimization

Disadvantages

- Performance can be quite sensitive to the choice of kernel
- Cost scales cubically with the number of observations
- Weak performance for high dimensionality
- Not easily applicable in discrete or conditional spaces
- Sensitive to its own hyperparameters

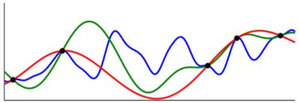
Gaussian Processes (GPs): Kernel Hyperparameters

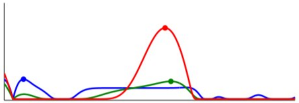
- We could optimize GP hyperparameters (maximum likelihood, MLE, or maximum a posteriori, MAP)
- But sampling GP hyperparameters from the posterior distribution performs better; e.g., via Markov-Chain Monte-Carlo (MCMC)

Gaussian Processes (GPs): Kernel Hyperparameters

- We could optimize GP hyperparameters (maximum likelihood, MLE, or maximum a posteriori, MAP)
- But sampling GP hyperparameters from the posterior distribution performs better; e.g., via Markov-Chain Monte-Carlo (MCMC)
- Marginalize over GP hyperparameters θ and compute an integrated acquisition function:

$$\bar{u}(\boldsymbol{\lambda}) = \int u(\boldsymbol{\lambda}, \hat{c}_\theta) p(\theta) d\theta$$


(a) Posterior samples under varying hyperparameters


(b) Expected improvement under varying hyperparameters

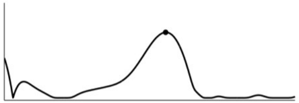

(c) Integrated expected improvement

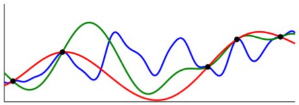
Image source: [Snoek et al. 2015]

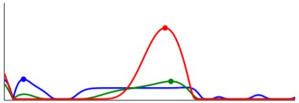
Gaussian Processes (GPs): Kernel Hyperparameters

- We could optimize GP hyperparameters (maximum likelihood, MLE, or maximum a posteriori, MAP)
- But sampling GP hyperparameters from the posterior distribution performs better; e.g., via Markov-Chain Monte-Carlo (MCMC)
- Marginalize over GP hyperparameters θ and compute an integrated acquisition function:

$$\bar{u}(\boldsymbol{\lambda}) = \int u(\boldsymbol{\lambda}, \hat{c}_{\theta}) p(\theta) d\theta$$

- Downside: computational expense
  - MCMC is computationally expensive
  - Acquisition function now has to be calculated for each sample


(a) Posterior samples under varying hyperparameters


(b) Expected improvement under varying hyperparameters

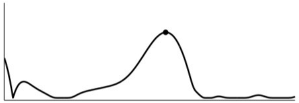
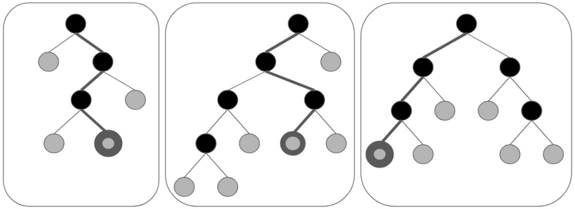

(c) Integrated expected improvement

Image source: [Snoek et al. 2015]

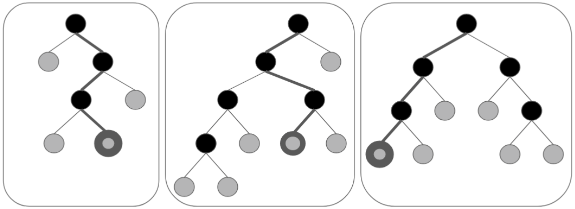
Random Forests (RFs): Reminder & How To Compute Uncertainties



RF Training

- Fit a set of randomized regression trees
- Randomization via bootstrapping & random selection of split variables / split points
- Each tree yields a possible explanation for the observations

Random Forests (RFs): Reminder & How To Compute Uncertainties



RF Training

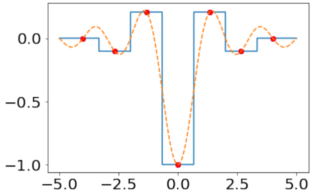
- Fit a set of randomized regression trees
- Randomization via bootstrapping & random selection of split variables / split points
- Each tree yields a possible explanation for the observations

RF Prediction

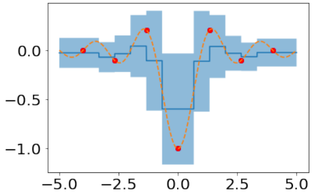
- Predict with each tree
- Aggregate predictions (e.g., average)
- Uncertainty estimate: empirical variance across tree predictions

Random Forests (RFs): Impact of Basic Model Choices

(a) no bootstrapping,
no random splits

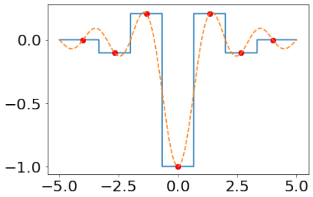


(b) with bootstrapping,
no random splits

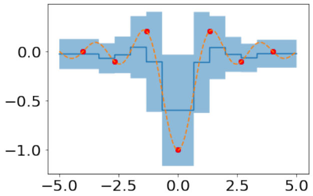


Random Forests (RFs): Impact of Basic Model Choices

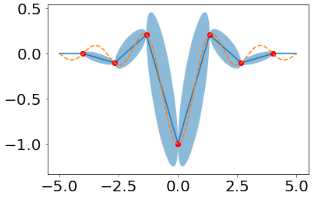
(a) no bootstrapping, no random splits



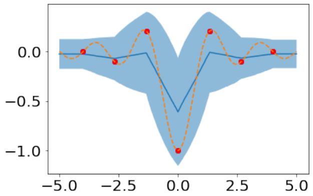
(b) with bootstrapping, no random splits



(c) no bootstrapping, with random splits



(d) with bootstrapping, with random splits



Random Forests (RFs): Overview of Pros and Cons

Advantages

- Cheap to train
- Scales well with #observations n:
  - Fitting: $O(n \log n)$
  - Prediction: $O(\log n)$
- Scales well with #dimensions
- Training can be parallelized
- Can easily handle conditional, categorical, continuous and discrete spaces
- Quite robust against its own hyperparameters

Random Forests (RFs): Overview of Pros and Cons

Advantages

- Cheap to train
- Scales well with #observations n:
  - Fitting: $O(n \log n)$
  - Prediction: $O(\log n)$
- Scales well with #dimensions
- Training can be parallelized
- Can easily handle conditional, categorical, continuous and discrete spaces
- Quite robust against its own hyperparameters

Disadvantages

- Poor uncertainty estimates
- Poor extrapolation (constant)
- Priors cannot be incorporated easily

Random Forests (RFs): Overview of Pros and Cons

Advantages

- Cheap to train
- Scales well with #observations n:
  - Fitting: $O(n \log n)$
  - Prediction: $O(\log n)$
- Scales well with #dimensions
- Training can be parallelized
- Can easily handle conditional, categorical, continuous and discrete spaces
- Quite robust against its own hyperparameters

Disadvantages

- Poor uncertainty estimates
- Poor extrapolation (constant)
- Priors cannot be incorporated easily

These qualities make RFs a robust option for Bayesian optimization in high dimensions, for categorical spaces, or when function evaluations are quite fast

Bayesian Neural Networks: Overview

- Neural networks are more flexible & scalable than Gaussian processes
- But for use in Bayesian optimization, neural networks need to be made probabilistic

Bayesian Neural Networks: Overview

- Neural networks are more flexible & scalable than Gaussian processes
- But for use in Bayesian optimization, neural networks need to be made probabilistic
- Bayesian deep learning aims to deal with all sources of uncertainty

Bayesian Neural Networks: Overview

- Neural networks are more flexible & scalable than Gaussian processes
- But for use in Bayesian optimization, neural networks need to be made probabilistic
- Bayesian deep learning aims to deal with all sources of uncertainty
  - E.g., we don't have a single weight vector anymore, but a distribution over weights

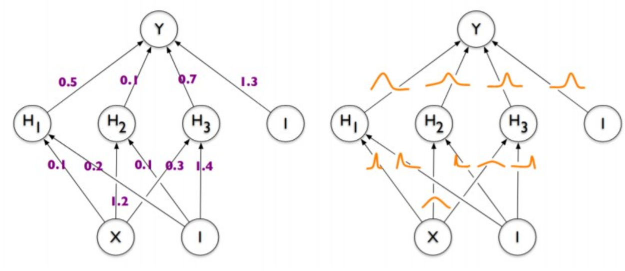


Image source: [Blundell et al. 2015]

Simplest Way of Incorporating Uncertainty in Neural Networks: DNGO

- Fit a standard regression neural network to the data (with a linear output layer)
- Use the representation in the last hidden layer as basis functions $\phi(x)$ of the input x
- Use Bayesian linear regression with these basis functions

Simplest Way of Incorporating Uncertainty in Neural Networks: DNGO

- Fit a standard regression neural network to the data (with a linear output layer)
- Use the representation in the last hidden layer as basis functions $\phi(x)$ of the input x
- Use Bayesian linear regression with these basis functions
  - The last layer is linear in its parameters θ
  - Therefore, the Bayesian linear regression formulas work directly
  - Feasible in closed form, in time $O(Nd^3)$, where N is the number of data points and d is the number of hidden units in the last layer
- Not fully Bayesian yet, but already allows scalable Bayesian optimization [Snoek et al. 2015]

Bayesian Optimization with BNNs: Overview of Existing Approaches

- Scalable Bayesian Optimization Using Deep Neural Networks (DNGO) [Snoek et al. 2015]
- Bayesian Optimization with Robust Bayesian Neural Networks [Springenberg et al. 2016]
- Parallel and Distributed Thompson Sampling for Large-scale Accelerated Exploration of Chemical Space [Hernández-Lobato et al. 2017]

Bayesian Optimization with BNNs: Overview of Existing Approaches

- Scalable Bayesian Optimization Using Deep Neural Networks (DNGO) [Snoek et al. 2015]
- Bayesian Optimization with Robust Bayesian Neural Networks [Springenberg et al. 2016]
- Parallel and Distributed Thompson Sampling for Large-scale Accelerated Exploration of Chemical Space [Hernández-Lobato et al. 2017]

- Hyperparameter Optimization with Factorized Multilayer Perceptrons [Schilling et al. 2015]
- Scalable Hyperparameter Transfer Learning [Perrone et al. 2018]

Bayesian Neural Networks (BNNs): Overview of Pros and Cons

Advantages

- Scales linearly with #observations
- Can obtain nice and smooth uncertainty estimates
- Flexibility: handling of categorical, continuous and discrete spaces

Bayesian Neural Networks (BNNs): Overview of Pros and Cons

Advantages

- Scales linearly with #observations
- Can obtain nice and smooth uncertainty estimates
- Flexibility: handling of categorical, continuous and discrete spaces

Disadvantages

- Usually needs more data than Gaussian processes
- Uncertainty estimates often worse than for Gaussian processes
- Many meta-design decisions
- No robust off-the-shelf implementation

Bayesian Neural Networks (BNNs): Overview of Pros and Cons

Advantages

- Scales linearly with #observations
- Can obtain nice and smooth uncertainty estimates
- Flexibility: handling of categorical, continuous and discrete spaces

Disadvantages

- Usually needs more data than Gaussian processes
- Uncertainty estimates often worse than for Gaussian processes
- Many meta-design decisions
- No robust off-the-shelf implementation

These qualities make BNNs an ever-more promising alternative

Bayesian Neural Networks (BNNs): Further Reading

There is a lot more work on BNNs that hasn't been applied to Bayesian optimization yet:

- Ensembles obtained simply by running SGD several times [Lakshminarayanan et al. 2016]
- Dropout [Gal and Ghahramani. 2015]
- Monte Carlo Batch Normalization [Teye et al. 2018]
- Snapshot Ensembles [Gao Huang et al. 2017]

Questions to Answer for Yourself / Discuss with Friends

- Discussion. For which optimization problems would you rather use a RF than a GP? When would you use a BNN?
- Discussion. Why can DNGO's Bayesian Linear Regression approach only be applied to the last layer of a Deep Neural Network, not to all layers?
- Open Question. All of the surrogate models we saw have pros and cons. Would it be possible to select the best model (and its hyperparameters) dependent on the data at hand, and could this be done effectively? (This is a possible research project.)