Lecture 6: Bayesian optimization and contextual bandits applications

Hrvoje Stojic

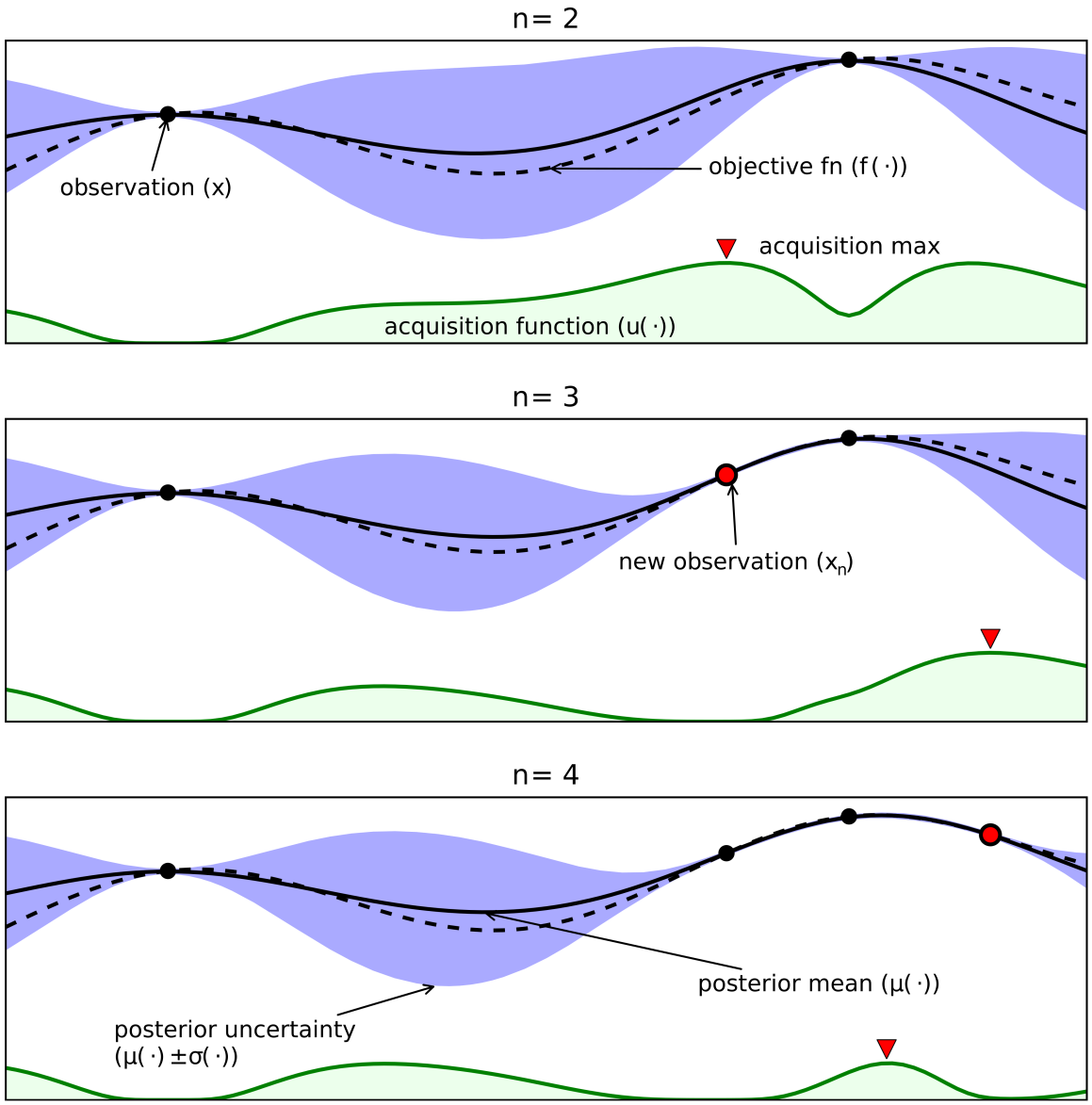
May 17, 2021

Bayesian optimization

Bayesian optimization with GP's

Source: Shahriari et al 2016

Bayesian optimization with GP's



Source: Shahriari et al 2016

Bayesian optimization algorithm

Bayesian optimization algorithm

1. **for** $t = 1, 2, \ldots$ do
2. Choose $\mathbf{x}_t$ by combining attributes of the posterior distribution in an acquisition function α and maximizing

$$\mathbf{x}_t = \text{argmax}_x \alpha(\mathbf{x}; \mathcal{D}_{1:t-1})$$

3. Obtain the reward (i.e. outcome of the objective function)

$$y_t = f(\mathbf{x}_t) + \epsilon_t$$

4. Augment the data $\mathcal{D}_{1:t} = \mathcal{D}_{1:t-1}, (\mathbf{x}_t, y_t)$ and update the GP
5. **end for**

Bayesian optimization algorithm

1. **for** $t = 1, 2, \ldots$ do
2. Choose $\mathbf{x}_t$ by combining attributes of the posterior distribution in an acquisition function α and maximizing

$$\mathbf{x}_t = \text{argmax}_x \alpha(\mathbf{x}; \mathcal{D}_{1:t-1})$$

3. Obtain the reward (i.e. outcome of the objective function)

$$y_t = f(\mathbf{x}_t) + \epsilon_t$$

4. Augment the data $\mathcal{D}_{1:t} = \mathcal{D}_{1:t-1}, (\mathbf{x}_t, y_t)$ and update the GP
5. **end for**

- Some acquisition functions α: UCB, Thompson sampling, Expected improvement, Probability of improvement, Entropy search, Predictive entropy search, Portfolios of acquisition functions (Hedge, Entropy search portfolio)

Bayesian optimization algorithm

1. **for** $t = 1, 2, \ldots$ do
2. Choose $\mathbf{x}_t$ by combining attributes of the posterior distribution in an acquisition function α and maximizing

$$\mathbf{x}_t = \text{argmax}_x \alpha(\mathbf{x}; \mathcal{D}_{1:t-1})$$

3. Obtain the reward (i.e. outcome of the objective function)

$$y_t = f(\mathbf{x}_t) + \epsilon_t$$

4. Augment the data $\mathcal{D}_{1:t} = \mathcal{D}_{1:t-1}, (\mathbf{x}_t, y_t)$ and update the GP
5. **end for**

- Some acquisition functions α: UCB, Thompson sampling, Expected improvement, Probability of improvement, Entropy search, Predictive entropy search, Portfolios of acquisition functions (Hedge, Entropy search portfolio)
- A bit of history: searching for gold with "kriging"

Upper confidence bound (UCB)

Upper confidence bound (UCB)

- Recall the expressions for GP prediction

$$P(y_{t+1}|\mathcal{D}_{1:t}, \mathbf{x}_{t+1} = \mathcal{N}(\mu_t(\mathbf{x}_{t+1}), \sigma_t^2(\mathbf{x}_{t+1}) + \sigma_n^2)$$

$$\mu_t(\mathbf{x}_{t+1}) = \mathbf{k}^T[K + \sigma_n^2 I]^{-1}\mathbf{y}_{1:t}$$

$$\sigma_t^2(\mathbf{x}_{t+1}) = k(\mathbf{x}_{t+1}, \mathbf{x}_{t+1}) - \mathbf{k}^T[K + \sigma_n^2 I]^{-1}\mathbf{k}$$

Upper confidence bound (UCB)

- Recall the expressions for GP prediction

$$P(y_{t+1}|\mathcal{D}_{1:t}, \mathbf{x}_{t+1} = \mathcal{N}(\mu_t(\mathbf{x}_{t+1}), \sigma_t^2(\mathbf{x}_{t+1}) + \sigma_n^2)$$

$$\mu_t(\mathbf{x}_{t+1}) = \mathbf{k}^T[K + \sigma_n^2 I]^{-1}\mathbf{y}_{1:t}$$

$$\sigma_t^2(\mathbf{x}_{t+1}) = k(\mathbf{x}_{t+1}, \mathbf{x}_{t+1}) - \mathbf{k}^T[K + \sigma_n^2 I]^{-1}\mathbf{k}$$

- UCB acquisition function: $\mu_t(\mathbf{x}_t) + \beta_t \sigma_t(\mathbf{x}_t)$

Upper confidence bound (UCB)

- Recall the expressions for GP prediction

$$P(y_{t+1}|\mathcal{D}_{1:t}, \mathbf{x}_{t+1} = \mathcal{N}(\mu_t(\mathbf{x}_{t+1}), \sigma_t^2(\mathbf{x}_{t+1}) + \sigma_n^2)$$

$$\mu_t(\mathbf{x}_{t+1}) = \mathbf{k}^T[K + \sigma_n^2 I]^{-1}\mathbf{y}_{1:t}$$

$$\sigma_t^2(\mathbf{x}_{t+1}) = k(\mathbf{x}_{t+1}, \mathbf{x}_{t+1}) - \mathbf{k}^T[K + \sigma_n^2 I]^{-1}\mathbf{k}$$

- UCB acquisition function: $\mu_t(\mathbf{x}_t) + \beta_t \sigma_t(\mathbf{x}_t)$
- Regret bound for RBF kernel: $\sqrt{T(\log T)^{d+1}}$ (Srinivas et al., 2010)

Probability of Improvement (PI)

Source: de Freitas lectures, 2013

Probability of Improvement (PI)

- Favours points that are likely to improve on best so far
- Defined as

$$\mathrm{PI}(\mathbf{x}) = P(f(\mathbf{x}) \geq \mu^+ + \epsilon) = \Phi\left(\frac{\mu_t(\mathbf{x}) - \mu^+ - \epsilon}{\sigma_t(\mathbf{x})}\right)$$

  - where μ^+ is the best observed value so far and ϵ is a free parameter

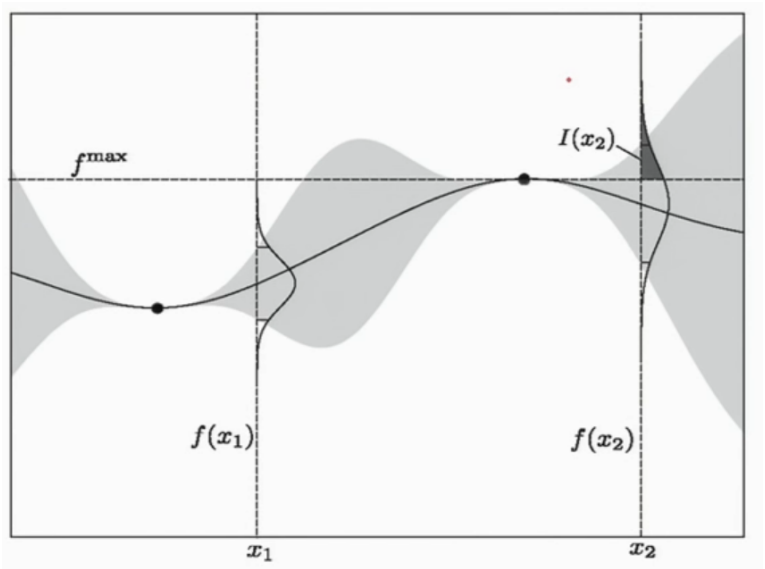
Source: de Freitas lectures, 2013

Probability of Improvement (PI)

- Favours points that are likely to improve on best so far
- Defined as

$$\mathrm{PI}(\mathbf{x}) = P(f(\mathbf{x}) \geq \mu^+ + \epsilon) = \Phi\left(\frac{\mu_t(\mathbf{x}) - \mu^+ - \epsilon}{\sigma_t(\mathbf{x})}\right)$$

  - where μ^+ is the best observed value so far and ϵ is a free parameter



Source: de Freitas lectures, 2013

Expected Improvement (EI)

Expected Improvement (EI)

- Improves over PI by incorporating the amount of improvement
- Expected utility approach

$$\mathbf{x}_{t+1} = \text{argmin}_{\mathbf{x}} \int \|f_{t+1}(\mathbf{x}) - f(\mathbf{x}^*)\| P(f_{t+1}|\mathcal{D}_t) df_{t+1}$$

Expected Improvement (EI)

- Improves over PI by incorporating the amount of improvement
- Expected utility approach

$$\mathbf{x}_{t+1} = \text{argmin}_{\mathbf{x}} \int \|f_{t+1}(\mathbf{x}) - f(\mathbf{x}^*)\| P(f_{t+1}|\mathcal{D}_t) df_{t+1}$$

- True objective at the max is not available, Mockus (1991) proposed the expected improvement approximation
  $\text{EI}(\mathbf{x}) = \mathbb{E}\left[\max\left\{0, f(\mathbf{x}) - \mu^+ - \epsilon\right\} | \mathcal{D}_t\right]$

Expected Improvement (EI)

- Improves over PI by incorporating the amount of improvement
- Expected utility approach

$$\mathbf{x}_{t+1} = \text{argmin}_{\mathbf{x}} \int \|f_{t+1}(\mathbf{x}) - f(\mathbf{x}^*)\| P(f_{t+1}|\mathcal{D}_t) df_{t+1}$$

- True objective at the max is not available, Mockus (1991) proposed the expected improvement approximation
  $\text{EI}(\mathbf{x}) = \mathbb{E}\left[\max\left\{0, f(\mathbf{x}) - \mu^+ - \epsilon\right\}|\mathcal{D}_t\right]$
- We can obtain an analytical expression with the GP

$$\text{EI}(\mathbf{x}) = (\mu_t(\mathbf{x}) - \mu^+ - \epsilon)\Phi(Z) + \sigma_t(\mathbf{x})\phi(Z)$$

  - when $\sigma_t > 0$, 0 if $\sigma_t = 0$
  - where $Z = \frac{\mu_t(\mathbf{x}) - \mu^+ - \epsilon}{\sigma_t(\mathbf{x})}$

Expected Improvement (EI)

- Improves over PI by incorporating the amount of improvement
- Expected utility approach

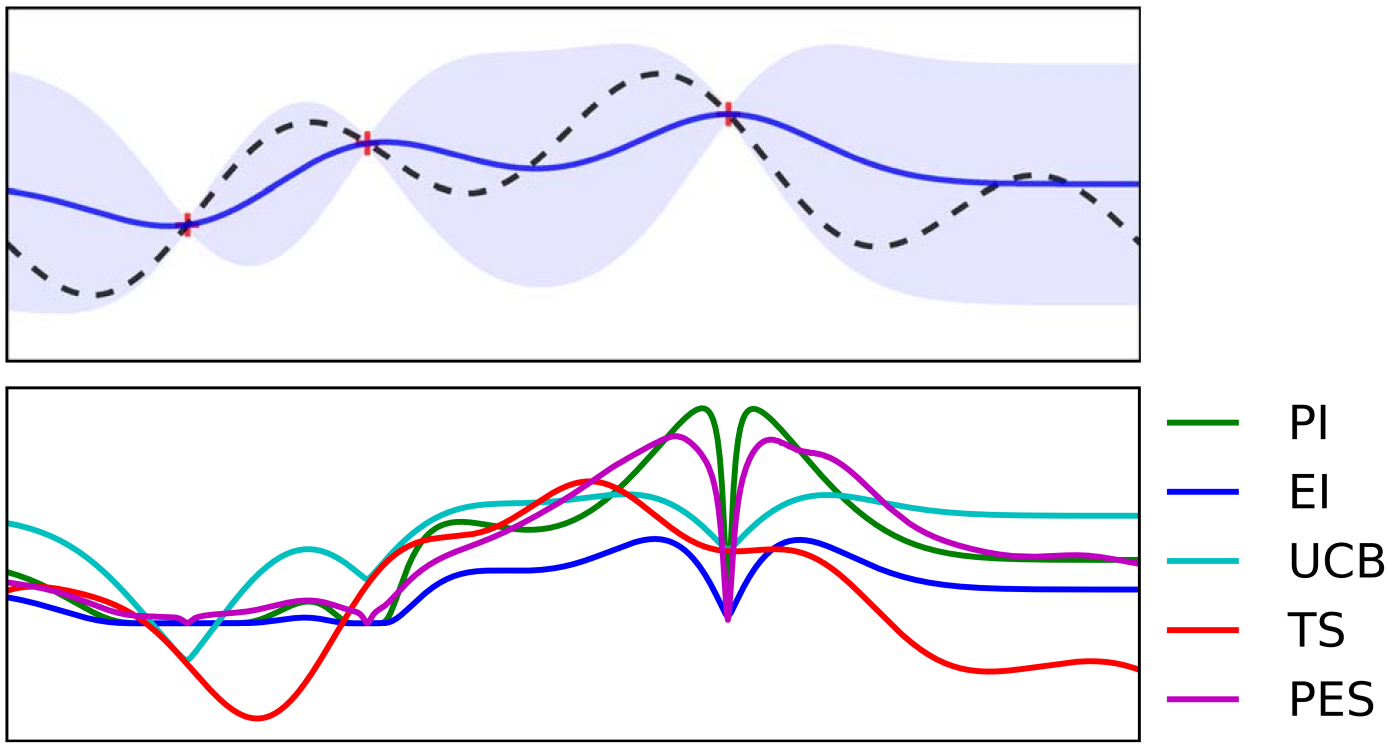
$$\mathbf{x}_{t+1} = \text{argmin}_{\mathbf{x}} \int \|f_{t+1}(\mathbf{x}) - f(\mathbf{x}^*)\| P(f_{t+1}|\mathcal{D}_t) df_{t+1}$$

- True objective at the max is not available, Mockus (1991) proposed the expected improvement approximation
  $\text{EI}(\mathbf{x}) = \mathbb{E}\left[\max\left\{0, f(\mathbf{x}) - \mu^+ - \epsilon\right\}|\mathcal{D}_t\right]$
- We can obtain an analytical expression with the GP

$$\text{EI}(\mathbf{x}) = (\mu_t(\mathbf{x}) - \mu^+ - \epsilon)\Phi(Z) + \sigma_t(\mathbf{x})\phi(Z)$$

  - when $\sigma_t > 0$, 0 if $\sigma_t = 0$
  - where $Z = \frac{\mu_t(\mathbf{x}) - \mu^+ - \epsilon}{\sigma_t(\mathbf{x})}$
- EI is high when the (posterior) expected value $\mu_t(\mathbf{x})$ is higher than the current best value μ^+; or when the uncertainty $\sigma_t(\mathbf{x})$ around the point $\mathbf{x}$ is high.

Acquisition function illustration

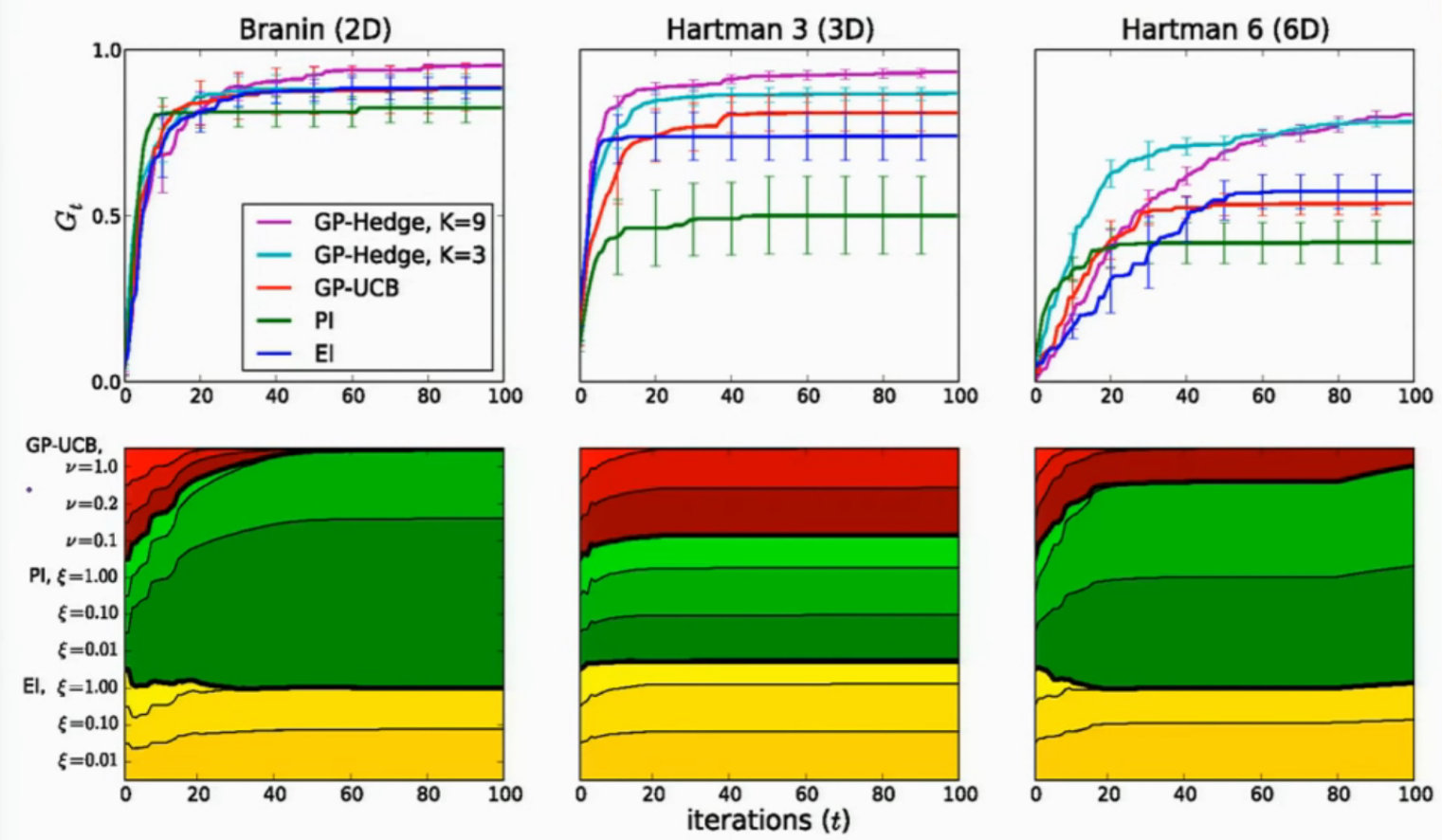


Source: Shahriari et al (2016) Figure 5

Portfolio of acquisition functions

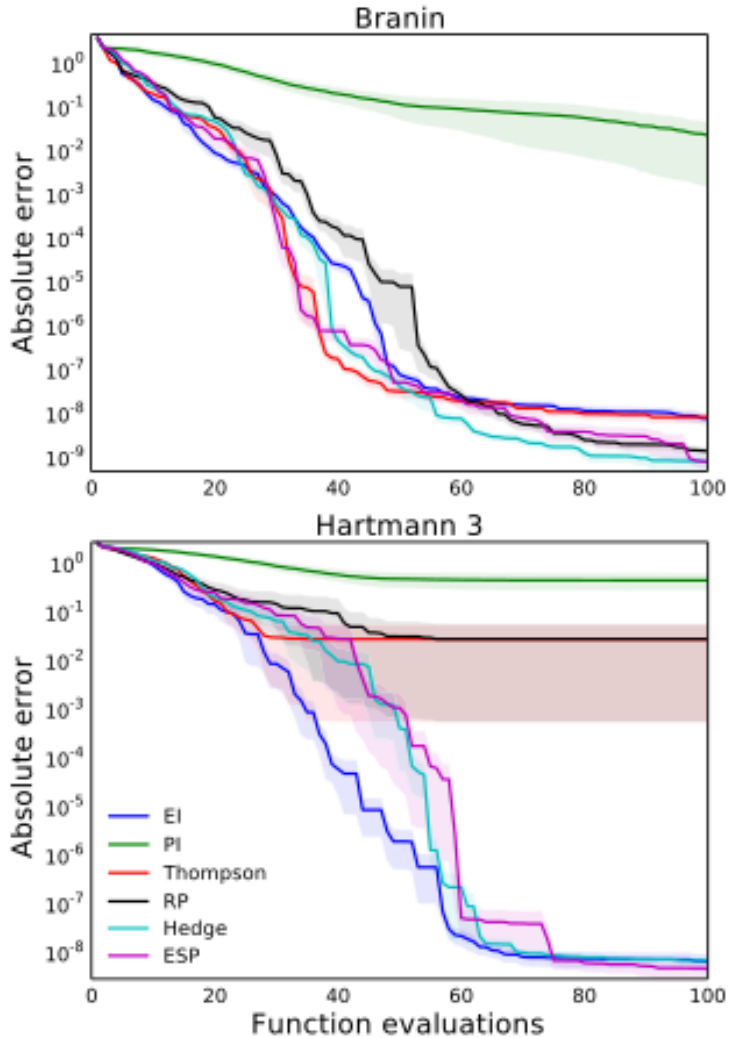
Source: Hoffman, Brochu, de Freitas (2011); Shahriari et al (2014)

Portfolio of acquisition functions



Source: Hoffman, Brochu, de Freitas (2011); Shahriari et al (2014)

Portfolios might be the best approach



Source: Shahriari et al (2016) Figure 6

Optimization: handling GP hyperparameters

Optimization: handling GP hyperparameters

- In BO typically GP is updated in each iteration and hyperparameters are optimized again as well
- This optimization most often involves multistarted quasi-Newton hill climbers using GP marginal likelihood

Optimization: handling GP hyperparameters

- In BO typically GP is updated in each iteration and hyperparameters are optimized again as well
- This optimization most often involves multistarted quasi-Newton hill climbers using GP marginal likelihood
- *Fully*-Bayesian treatment: marginalizing out the hyperparameters

$$\alpha_t(\mathbf{x}) = \mathbb{E}_{\theta|\mathcal{D}_t}[\alpha(\mathbf{x};\theta)] = \int \alpha(\mathbf{x};\theta) P(\theta|\mathcal{D}_t) d\theta$$

Optimization: handling GP hyperparameters

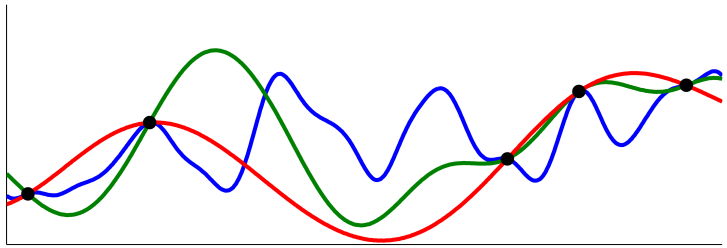
- In BO typically GP is updated in each iteration and hyperparameters are optimized again as well
- This optimization most often involves multistarted quasi-Newton hill climbers using GP marginal likelihood
- *Fully*-Bayesian treatment: marginalizing out the hyperparameters

$$\alpha_t(\mathbf{x}) = \mathbb{E}_{\theta|\mathcal{D}_t}[\alpha(\mathbf{x};\theta)] = \int \alpha(\mathbf{x};\theta) P(\theta|\mathcal{D}_t) d\theta$$

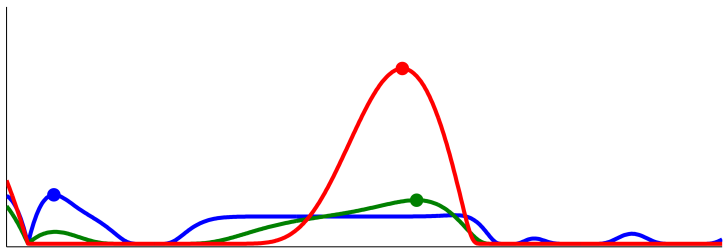
- Takes into account uncertainty about GP's hyperparameters and tends to improve uncertainty estimates of the function
- This can be done using either quadrature or Monte Carlo estimate (e.g. via slice sampling - Murray & Adams, 2010)

Optimization: handling GP hyperparameters

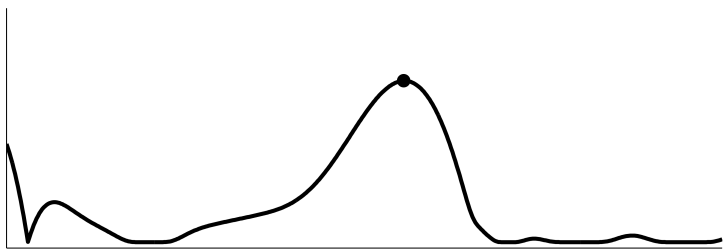
- Example of marginalizing out the hyperparameters with EI



(a) Posterior samples under varying hyperparameters



(b) Expected improvement under varying hyperparameters



(c) Integrated expected improvement

Source: Snoek et al (2012) Figure 1

Optimization: acquisition functions

Optimization: acquisition functions

- Acquisition functions are often multi-modal
- In continuous action spaces optimization is then not trivial
- Only useful if cheap relative to evaluating objective f

Optimization: acquisition functions

- Acquisition functions are often multi-modal
- In continuous action spaces optimization is then not trivial
- Only useful if cheap relative to evaluating objective f

- In practice:
  - Discretization and grid search (e.g. Snoek et al 2012)
  - Adaptive grids (Badernet, Kegl, 2010)
  - If gradients available (or can be approximated cheaply), then multi-started quasi-Newton hill climbing approach or multi-started local search
  - Difficult to asses the performance/convergence and its not clear whether assumptions for theoretical guarantees are met

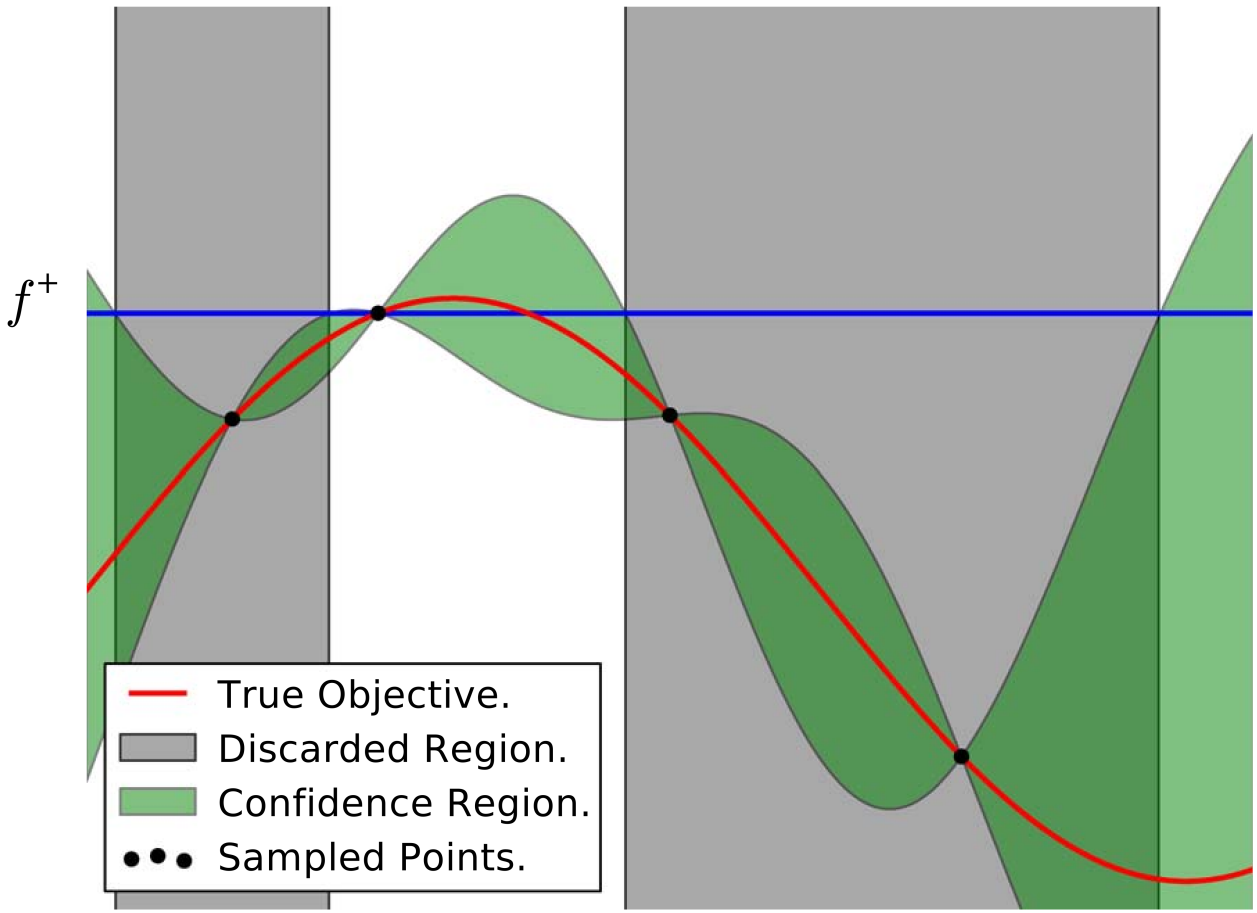
Optimization: acquisition functions

- Acquisition functions are often multi-modal
- In continuous action spaces optimization is then not trivial
- Only useful if cheap relative to evaluating objective f

- In practice:
  - Discretization and grid search (e.g. Snoek et al 2012)
  - Adaptive grids (Badernet, Kegl, 2010)
  - If gradients available (or can be approximated cheaply), then multi-started quasi-Newton hill climbing approach or multi-started local search
  - Difficult to asses the performance/convergence and its not clear whether assumptions for theoretical guarantees are met

- Recent developments
  - Optimistic optimization: Use the same optimism in the face of uncertainty on acquisition function optimization level as well (de Freitas, Smola and Zoghi, 2012))
  - BamSOO: shrinks the region in every iteration to the most promising ones (Wang, Shakibi, Jin and de Freitas, 2014)

Optimizing acquisition functions with optimistic optimization



Source: de Freitas, Smola, Zoghi (2012)

Going further: Taking into account evaluation costs

Going further: Taking into account evaluation costs

- In realistic settings, evaluating objective function might entail different costs for different $\mathbf{x}$ and you might have access to it
  - E.g. training a neural network with 100 vs. 10000 nodes requires much more memory and would take linger time - you could use training duration as a cost, or cost of renting a larger cloud instance for a longer time
- If there is a limited budget, then the search should be biased toward low-cost areas

Going further: Taking into account evaluation costs

- In realistic settings, evaluating objective function might entail different costs for different $\mathbf{x}$ and you might have access to it
  - E.g. training a neural network with 100 vs. 10000 nodes requires much more memory and would take linger time - you could use training duration as a cost, or cost of renting a larger cloud instance for a longer time
- If there is a limited budget, then the search should be biased toward low-cost areas
- Snoek et al (2012): *expected improvement per second*
  - Duration function is also not known
  - $c(\mathbf{x}) : \mathcal{X} \rightarrow R^{+}$
  - We can use another GP model to estimate $c()$
  - Combine the expected improvement with duration, $\mathrm{EI}(\mathbf{x}, \mathcal{D}_t)/c(\mathbf{x})$
  - Biases the search toward good models with fast training times

Expected improvement per second

Source: Snoek et al (2012) Figure 3

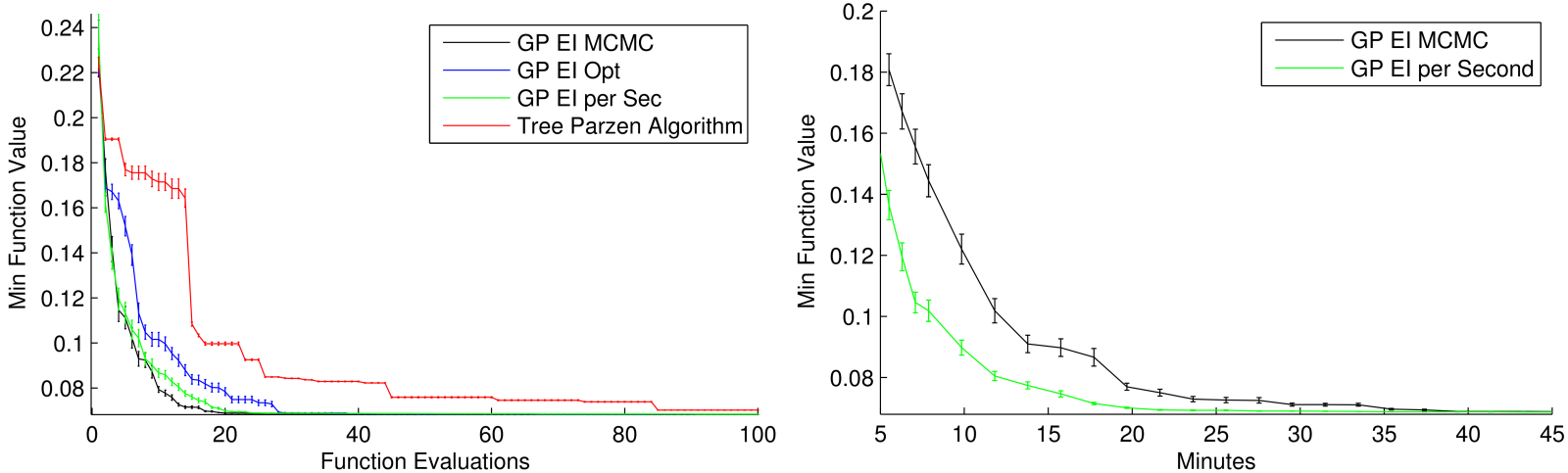
Expected improvement per second

- Optimizing hyperparameters for training logistic regression on MNIST dataset

Source: Snoek et al (2012) Figure 3

Expected improvement per second

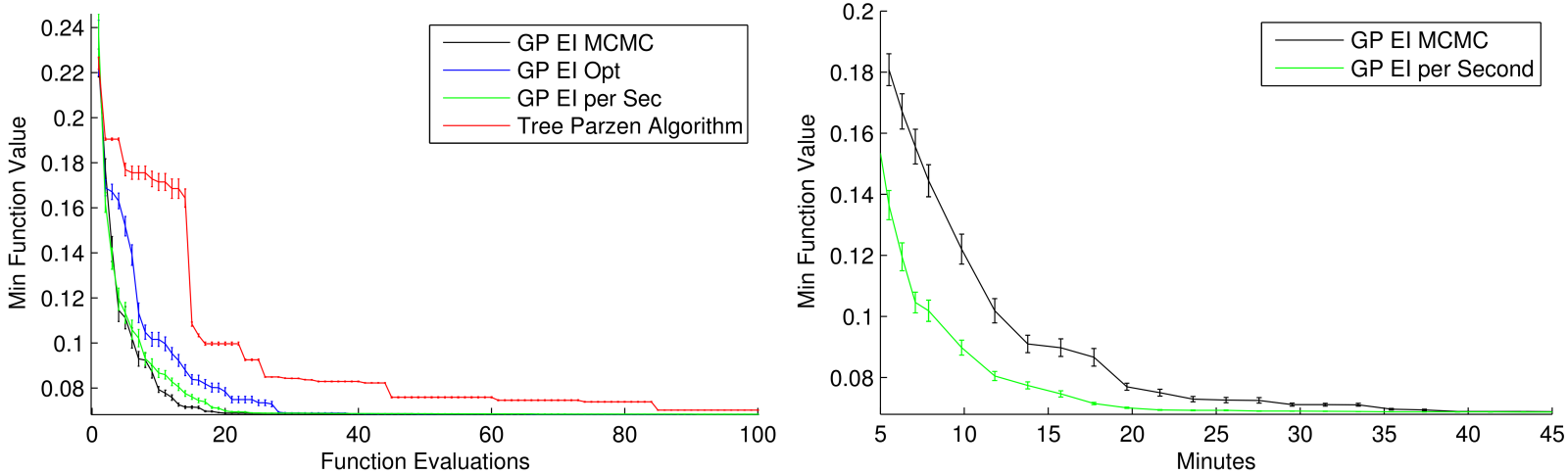
- Optimizing hyperparameters for training logistic regression on MNIST dataset



Source: Snoek et al (2012) Figure 3

Expected improvement per second

- Optimizing hyperparameters for training logistic regression on MNIST dataset



- Little difference in performance (approaching min) between EI MCMC and EI per second, but there is a large difference in amount of time it takes

Source: Snoek et al (2012) Figure 3

Going further: Safe exploration and risk-averse BO

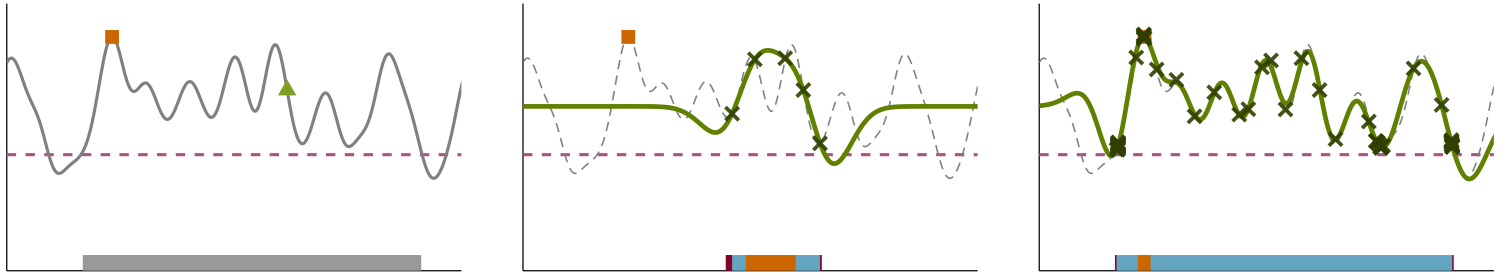
Going further: Safe exploration and risk-averse BO

- In many applications there are outputs that you do not wish to experience

Going further: Safe exploration and risk-averse BO

- In many applications there are outputs that you do not wish to experience

- We can leverage lower confidence bounds to avoid harm (SafeOpt algorithm; Sui et al., 2015)

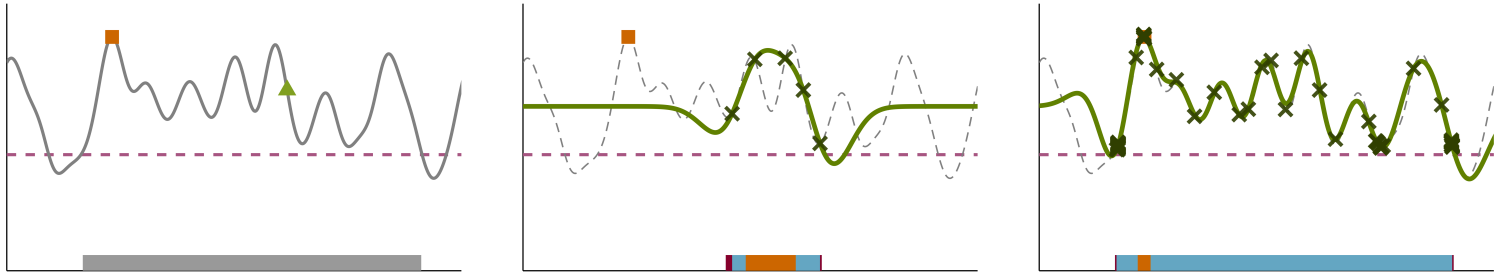


(a) True function and seed set (b) $t = 5$ (c) $t = 100$

Going further: Safe exploration and risk-averse BO

- In many applications there are outputs that you do not wish to experience

- We can leverage lower confidence bounds to avoid harm (SafeOpt algorithm; Sui et al., 2015)



(a) True function and seed set (b) $t = 5$ (c) $t = 100$

- Related to this is risk-averse BO - accounting for the distribution tails (see Torossian, Picheny and Durrande, 2020)

Going further: Beyond low-dimensional problems with REMBO

Source: Shahriari et al (2016) Figure 10

Going further: Beyond low-dimensional problems with REMBO

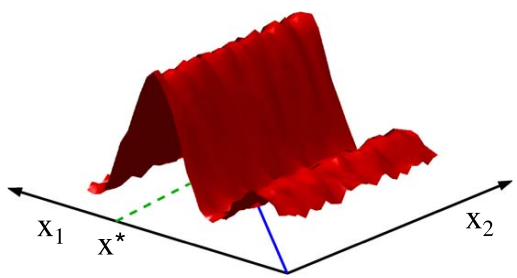
- In continuous action space, Bayesian optimization is restricted to problems of moderate dimension (approx 20)
- In practice, some dimension reduction can be done first
- Random forests scale more easily (SMAC algorithm)

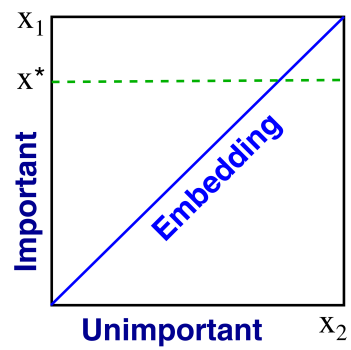
Source: Shahriari et al (2016) Figure 10

Going further: Beyond low-dimensional problems with REMBO

- In continuous action space, Bayesian optimization is restricted to problems of moderate dimension (approx 20)
- In practice, some dimension reduction can be done first
- Random forests scale more easily (SMAC algorithm)

- New approaches combining BO with random search
  - Many problems have low effective dimensionality
  - Rationale why random search sometimes performs well (Bergstra, Bengio, 2012)
  - Bayesian optimization with random embedding (REMBO; Wang et al. 2013)





Source: Shahriari et al (2016) Figure 10

Going further: Parallelization

Going further: Parallelization

- If we are concerned with wallclock time, there are several ways to speed up evaluation

Going further: Parallelization

- If we are concerned with wallclock time, there are several ways to speed up evaluation

1. At decision time
   - When using portfolios
   - We could optimize in parallel many acquisition functions (or a single one with multiple exploration values)

Going further: Parallelization

- If we are concerned with wallclock time, there are several ways to speed up evaluation

1. At decision time
   - When using portfolios
   - We could optimize in parallel many acquisition functions (or a single one with multiple exploration values)
2. While waiting for evaluation of the objective function
   - We could consider what $\mathbf{x}$ should be evaluated next
   - Snoek et al (2012) propose to compute MC estimates of the acquisition function under different possible results from pending function evaluations
   - With function like EI we can leverage Gaussian integration property

Addendum: Random forests as an alternative to GP

Source: Shahriari et al (2016) Figure 4

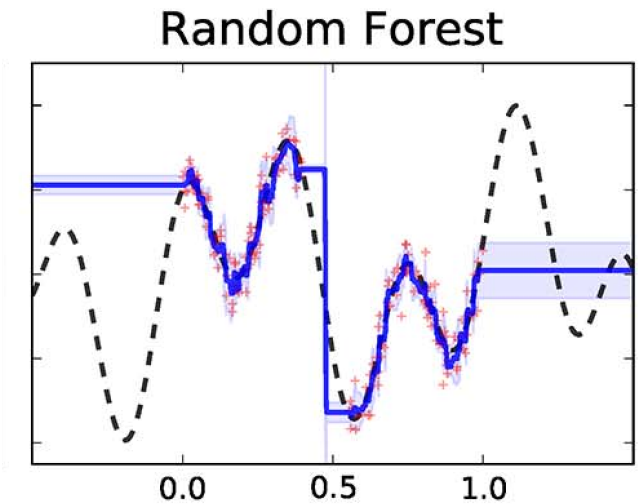
Addendum: Random forests as an alternative to GP

- Scale much better and can deal with categorical data
- Variance in predictions can be used as uncertainty estimate

Source: Shahriari et al (2016) Figure 4

Addendum: Random forests as an alternative to GP

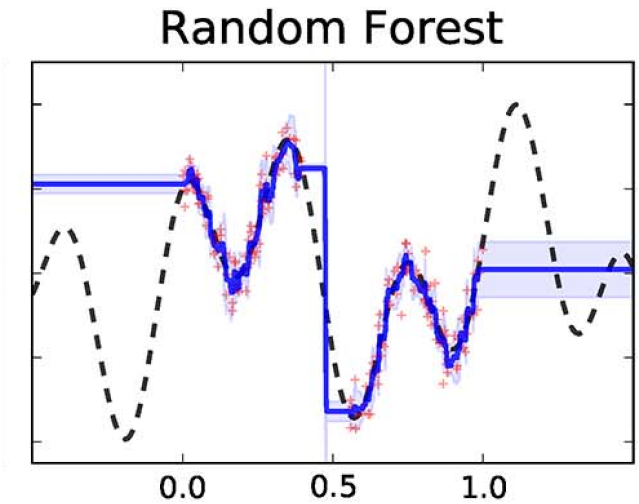
- Scale much better and can deal with categorical data
- Variance in predictions can be used as uncertainty estimate



Source: Shahriari et al (2016) Figure 4

Addendum: Random forests as an alternative to GP

- Scale much better and can deal with categorical data
- Variance in predictions can be used as uncertainty estimate



- Poor extrapolation behaviour, but in practice seems to work well (Hutter, Hoos, Leyton-Brown, 2011 - SMAC algorithm)

Source: Shahriari et al (2016) Figure 4

Addendum: Ensemble of neural networks

Source: Pearce et al (2019)

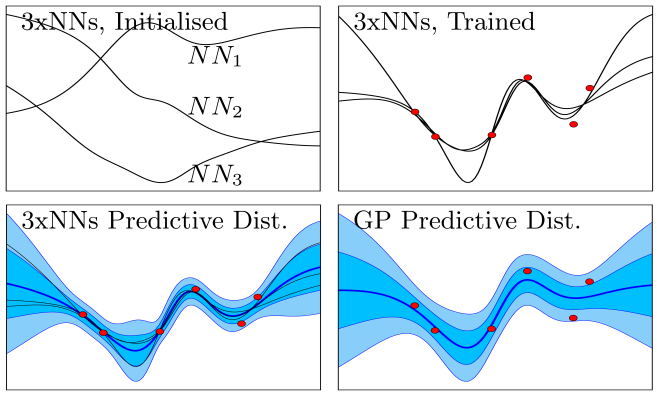
Addendum: Ensemble of neural networks

- A collection of a finite number of neural networks is trained for the same task (Hansen & Salamon, 1990)
- Variance in predictions can again be used as uncertainty estimate
- Recent work shows ensembles can represent uncertainty well, approaching GP's (Lakshminarayanan, Pritzel, Blundell, 2017; Pearce et al, 2019)

Source: Pearce et al (2019)

Addendum: Ensemble of neural networks

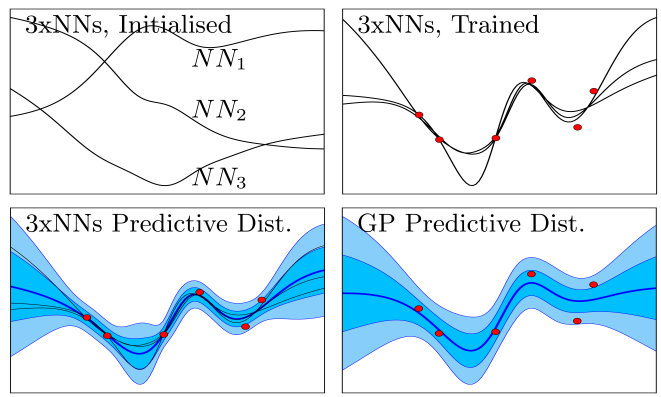
- A collection of a finite number of neural networks is trained for the same task (Hansen & Salamon, 1990)
- Variance in predictions can again be used as uncertainty estimate
- Recent work shows ensembles can represent uncertainty well, approaching GP's (Lakshminarayanan, Pritzel, Blundell, 2017; Pearce et al, 2019)



Source: Pearce et al (2019)

Addendum: Ensemble of neural networks

- A collection of a finite number of neural networks is trained for the same task (Hansen & Salamon, 1990)
- Variance in predictions can again be used as uncertainty estimate
- Recent work shows ensembles can represent uncertainty well, approaching GP's (Lakshminarayanan, Pritzel, Blundell, 2017; Pearce et al, 2019)



- Can be a costly method and not there yet in terms of representing uncertainty, but very promising

Source: Pearce et al (2019)

Application: Optimizing hyperparameters

The problem

The problem

- What are the hyperparameters and how do we optimize them?

The problem

- What are the hyperparameters and how do we optimize them?
- Some examples:

The problem

- What are the hyperparameters and how do we optimize them?
- Some examples:
  - SVM: regularisation term C, kernel parameters

The problem

- What are the hyperparameters and how do we optimize them?
- Some examples:
  - SVM: regularisation term C, kernel parameters
  - Linear regression on big data: SGD learning rate, regularization parameter, mini batch size, number of epochs

The problem

- What are the hyperparameters and how do we optimize them?
- Some examples:
  - SVM: regularisation term C, kernel parameters
  - Linear regression on big data: SGD learning rate, regularization parameter, mini batch size, number of epochs
  - Three-layer convolutional neural network: SGD learning rate, number of epochs, 4 x weight costs (layers and softmax), width, scale and power (normalization on the pooling layers)

The problem

- What are the hyperparameters and how do we optimize them?
- Some examples:
  - SVM: regularisation term C, kernel parameters
  - Linear regression on big data: SGD learning rate, regularization parameter, mini batch size, number of epochs
  - Three-layer convolutional neural network: SGD learning rate, number of epochs, 4 x weight costs (layers and softmax), width, scale and power (normalization on the pooling layers)

The problem

- What are the hyperparameters and how do we optimize them?
- Some examples:
  - SVM: regularisation term C, kernel parameters
  - Linear regression on big data: SGD learning rate, regularization parameter, mini batch size, number of epochs
  - Three-layer convolutional neural network: SGD learning rate, number of epochs, 4 x weight costs (layers and softmax), width, scale and power (normalization on the pooling layers)
- Standard procedures

The problem

- What are the hyperparameters and how do we optimize them?
- Some examples:
  - SVM: regularisation term C, kernel parameters
  - Linear regression on big data: SGD learning rate, regularization parameter, mini batch size, number of epochs
  - Three-layer convolutional neural network: SGD learning rate, number of epochs, 4 x weight costs (layers and softmax), width, scale and power (normalization on the pooling layers)
- Standard procedures
  - Grid search

The problem

- What are the hyperparameters and how do we optimize them?
- Some examples:
  - SVM: regularisation term C, kernel parameters
  - Linear regression on big data: SGD learning rate, regularization parameter, mini batch size, number of epochs
  - Three-layer convolutional neural network: SGD learning rate, number of epochs, 4 x weight costs (layers and softmax), width, scale and power (normalization on the pooling layers)
- Standard procedures
  - Grid search
  - Random Search

The problem

- What are the hyperparameters and how do we optimize them?
- Some examples:
  - SVM: regularisation term C, kernel parameters
  - Linear regression on big data: SGD learning rate, regularization parameter, mini batch size, number of epochs
  - Three-layer convolutional neural network: SGD learning rate, number of epochs, 4 x weight costs (layers and softmax), width, scale and power (normalization on the pooling layers)
- Standard procedures
  - Grid search
  - Random Search

The problem

- What are the hyperparameters and how do we optimize them?
- Some examples:
  - SVM: regularisation term C, kernel parameters
  - Linear regression on big data: SGD learning rate, regularization parameter, mini batch size, number of epochs
  - Three-layer convolutional neural network: SGD learning rate, number of epochs, 4 x weight costs (layers and softmax), width, scale and power (normalization on the pooling layers)
- Standard procedures
  - Grid search
  - Random Search
- Bayesian optimization as an alternative

Example 1: Tuning the SVM hyperparameters

- 20 dimensional problem, where the predictors are independent Gaussian random variables with mean zero and a variance of 9 (Sapp et al. 2014)

Example 1: Tuning the SVM hyperparameters

- 20 dimensional problem, where the predictors are independent Gaussian random variables with mean zero and a variance of 9 (Sapp et al. 2014)
- training set: 250 data points

Example 1: Tuning the SVM hyperparameters

- 20 dimensional problem, where the predictors are independent Gaussian random variables with mean zero and a variance of 9 (Sapp et al. 2014)
- training set: 250 data points
- radial basis SVM to model the data

Example 1: Tuning the SVM hyperparameters

- 20 dimensional problem, where the predictors are independent Gaussian random variables with mean zero and a variance of 9 (Sapp et al. 2014)
- training set: 250 data points
- radial basis SVM to model the data
  - two hyperparameters: regularization cost and radial basis parameter σ

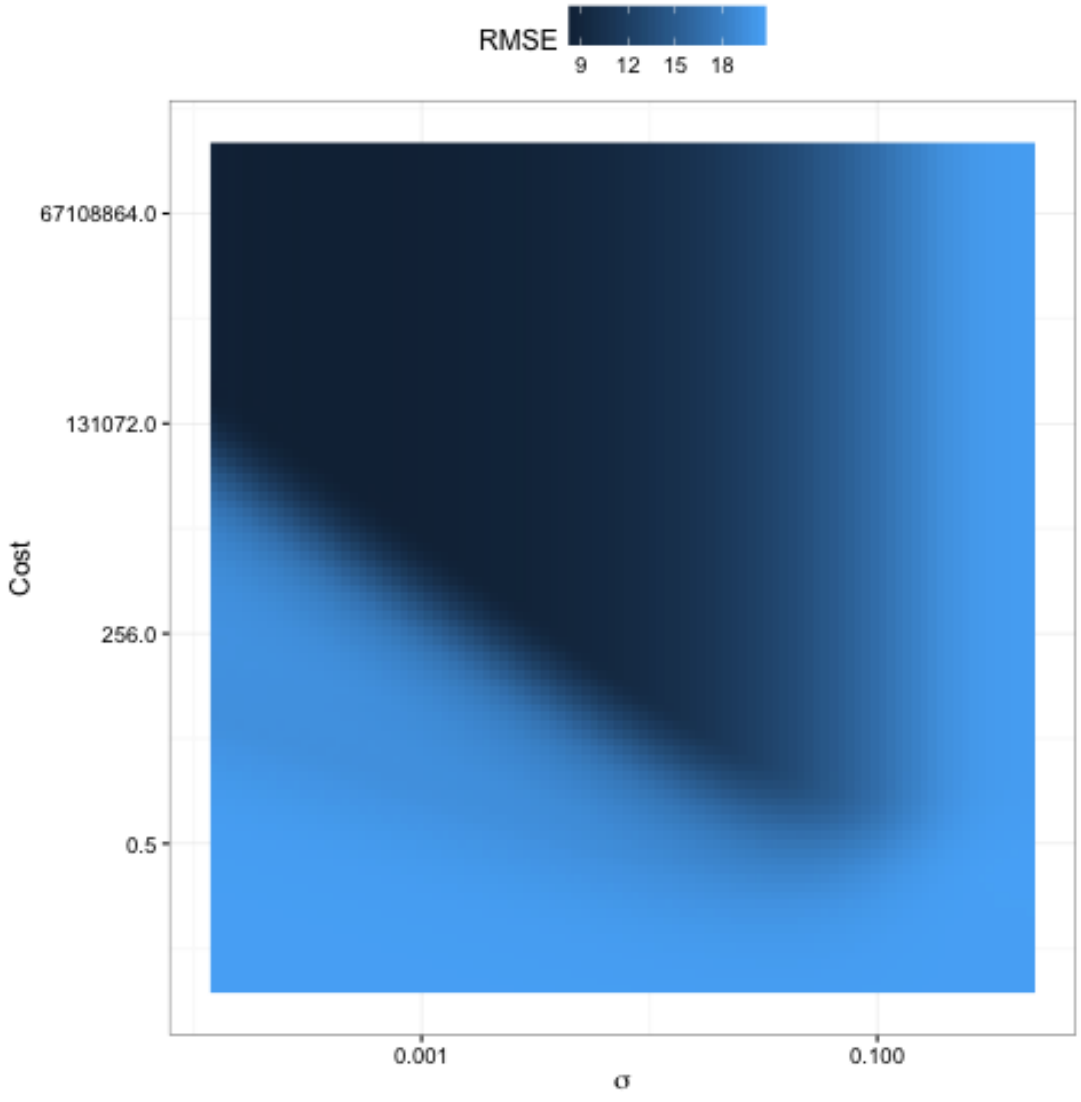
Example 1: Tuning the SVM hyperparameters

- 20 dimensional problem, where the predictors are independent Gaussian random variables with mean zero and a variance of 9 (Sapp et al. 2014)
- training set: 250 data points
- radial basis SVM to model the data
  - two hyperparameters: regularization cost and radial basis parameter σ
- example: Revolutionanalytics.com

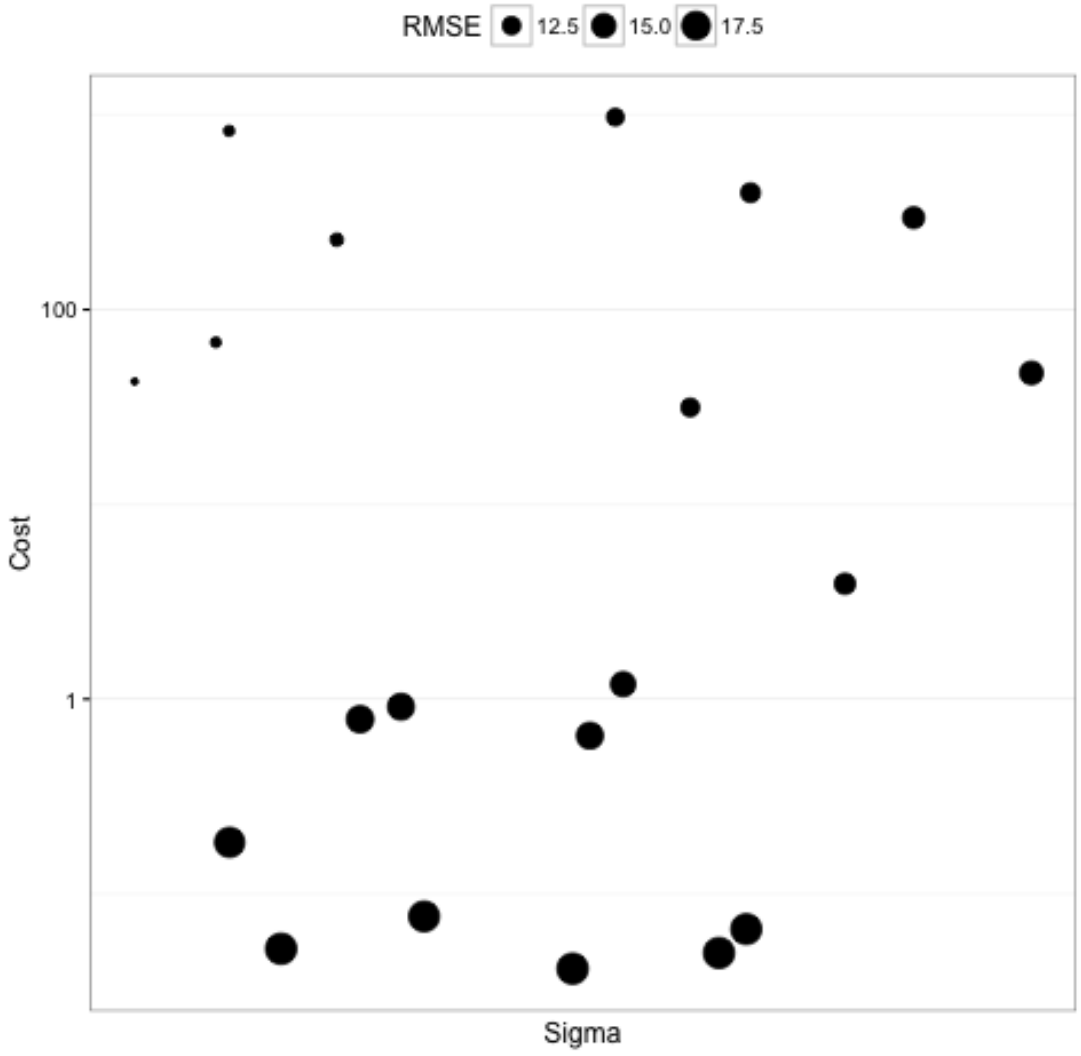
Example 1: Tuning the SVM hyperparameters

- 20 dimensional problem, where the predictors are independent Gaussian random variables with mean zero and a variance of 9 (Sapp et al. 2014)
- training set: 250 data points
- radial basis SVM to model the data
  - two hyperparameters: regularization cost and radial basis parameter σ
- example: Revolutionanalytics.com
  - using kernlab and rBayesianOptimization

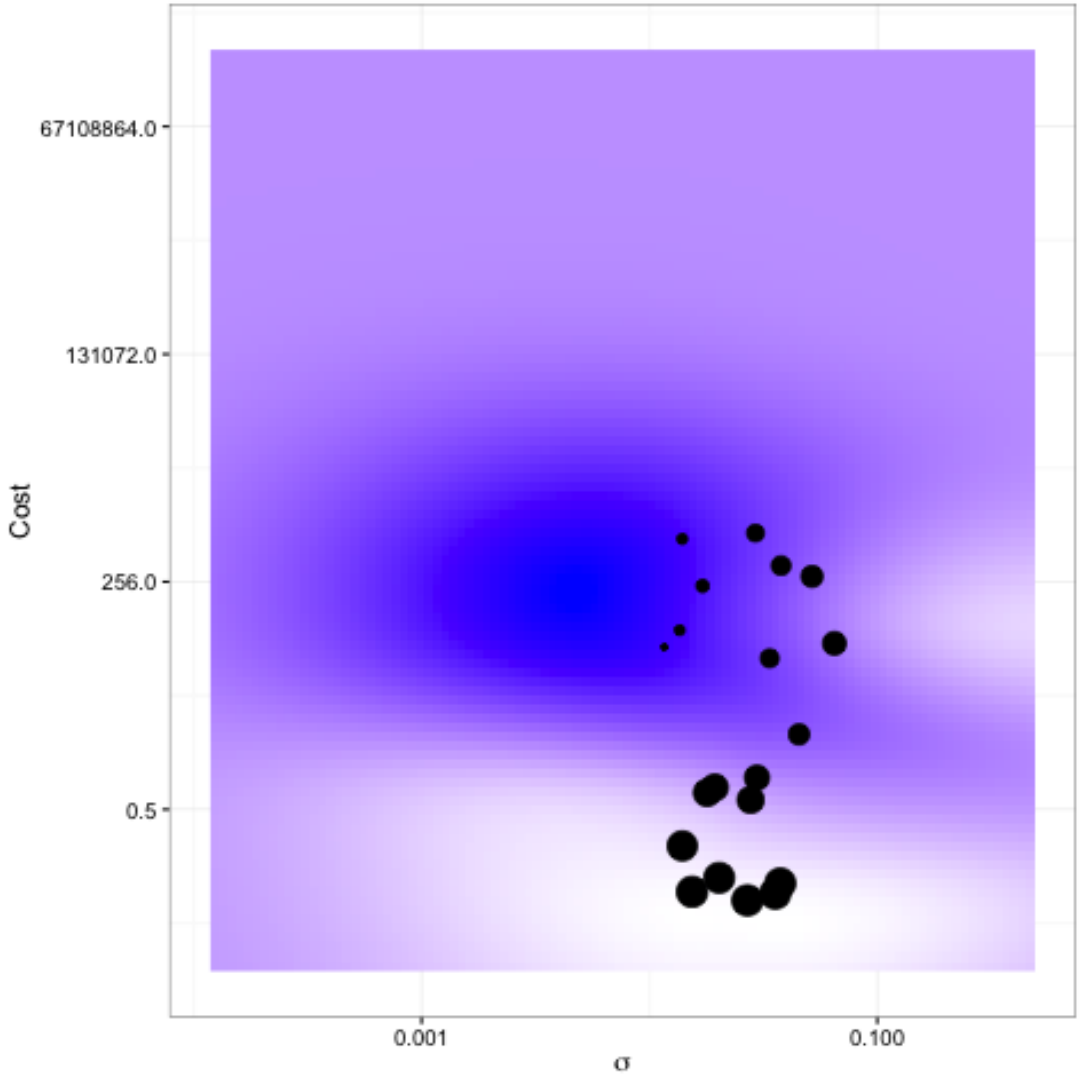
Example 1: RMSE surface



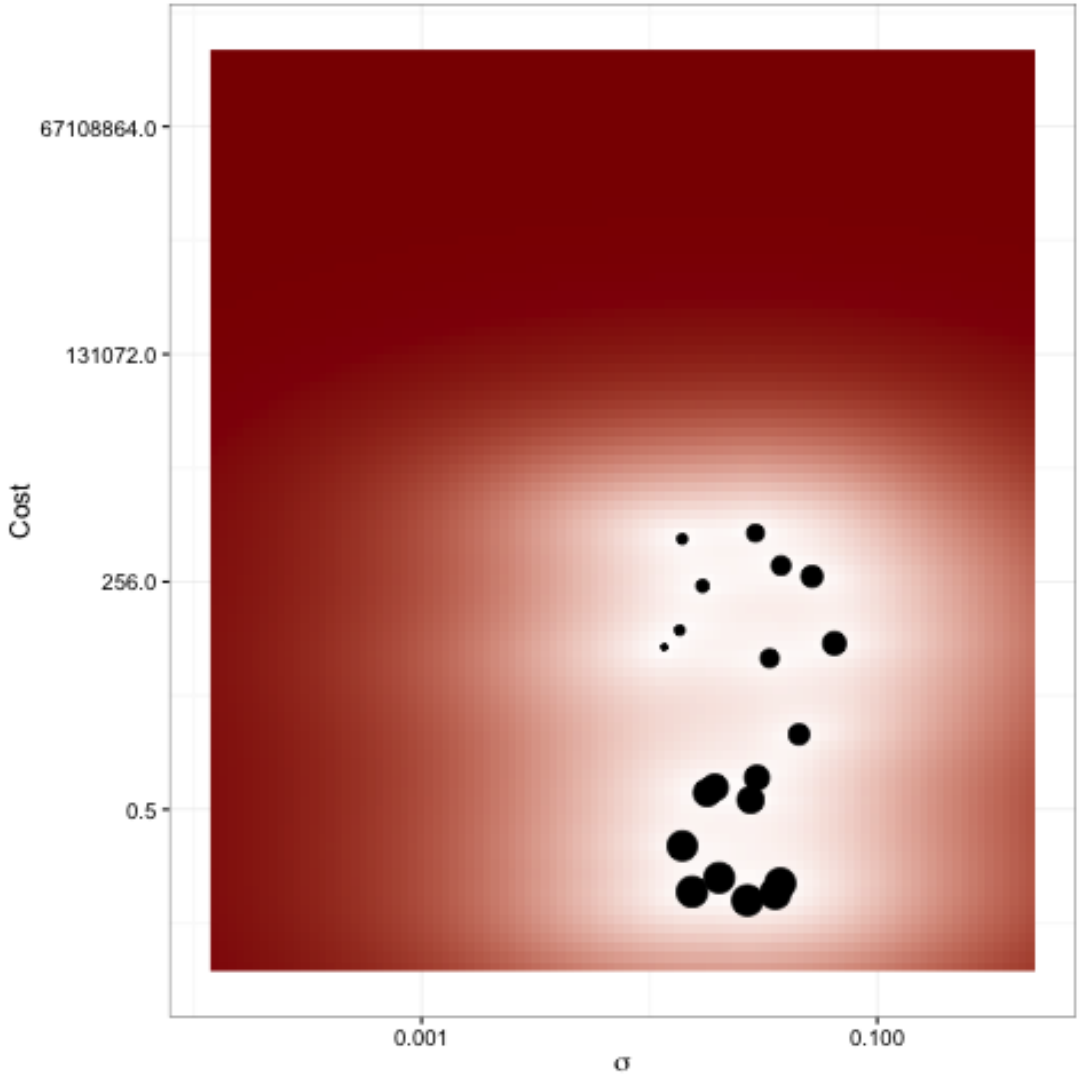
Example 1: Random search



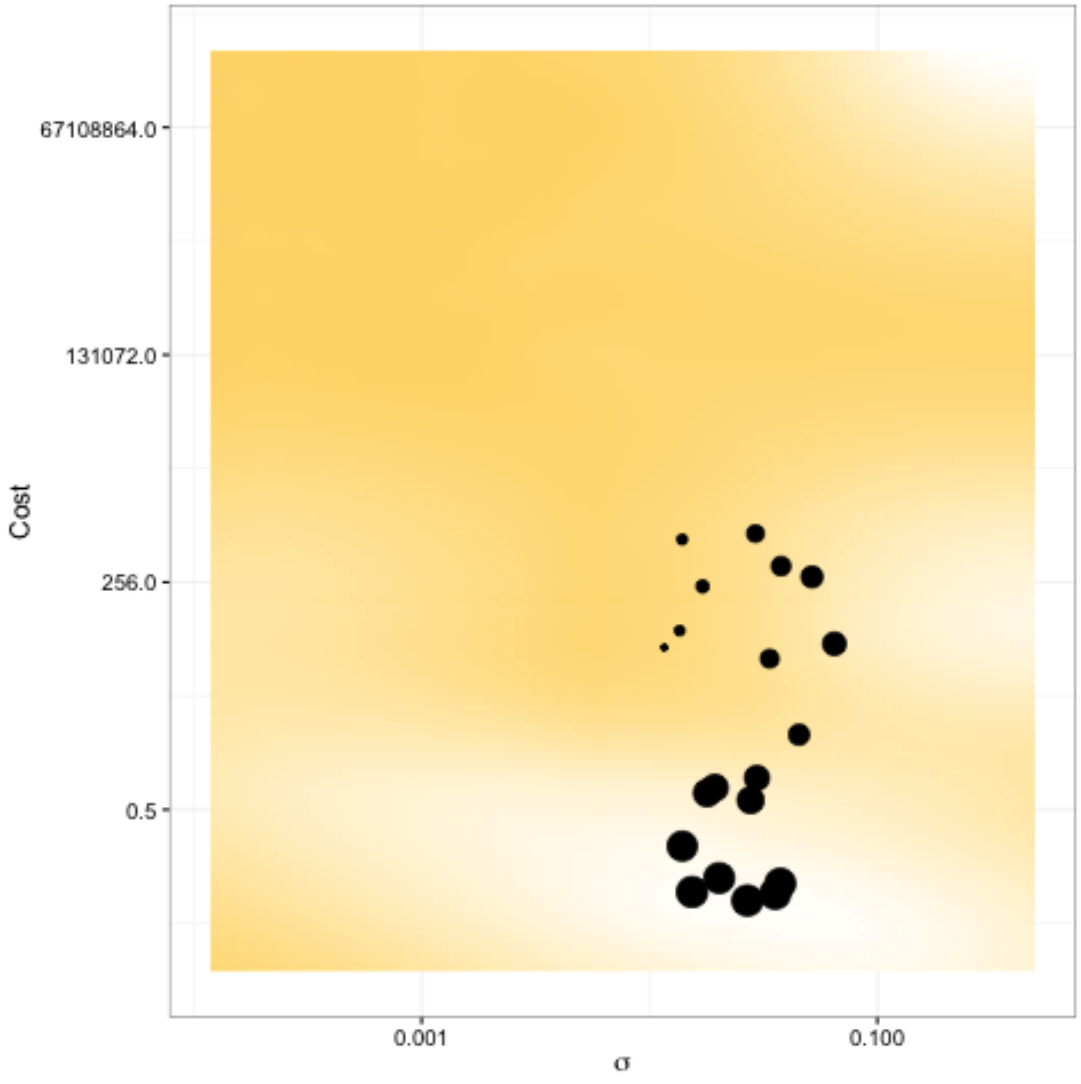
Example 1: GP predictive mean (based on initial random search)



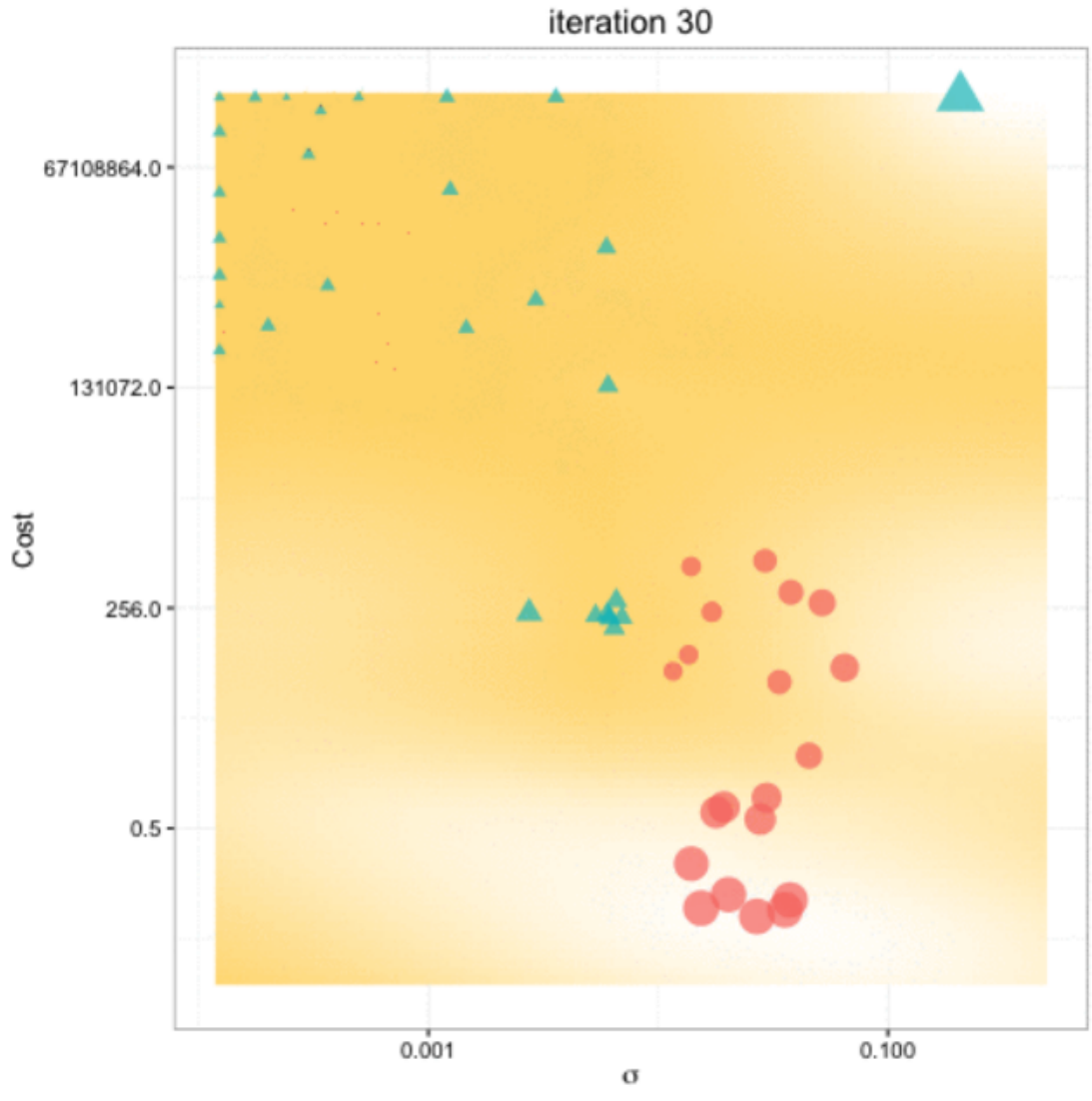
Example 1: GP predictive variance (based on initial random search)



Example 1: GP-UCB (based on initial random search)



Example 1: GP-UCB solution after 30 evaluations



Example 2: Tuning the Random forest hyperparameters

- scikit-learn's moons dataset, two classes and two features (Sapp et al. 2014)

Example 2: Tuning the Random forest hyperparameters

- scikit-learn's moons dataset, two classes and two features (Sapp et al. 2014)
- two hyperparameters:

Example 2: Tuning the Random forest hyperparameters

- scikit-learn's moons dataset, two classes and two features (Sapp et al. 2014)
- two hyperparameters:
  - number of Decision Trees we would like to have,

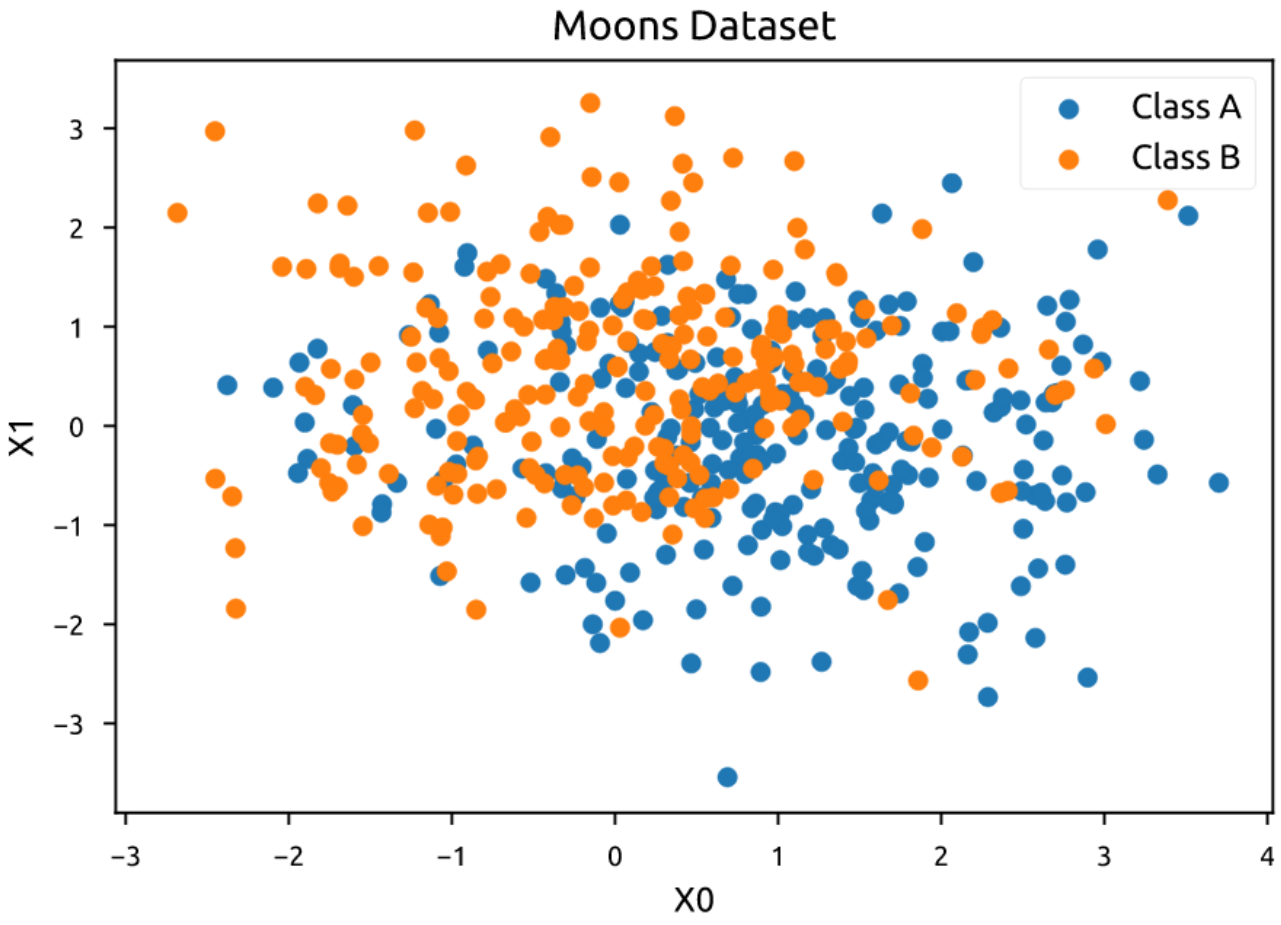
Example 2: Tuning the Random forest hyperparameters

- scikit-learn's moons dataset, two classes and two features (Sapp et al. 2014)
- two hyperparameters:
  - number of Decision Trees we would like to have,
  - the maximum depth for each of those decision trees

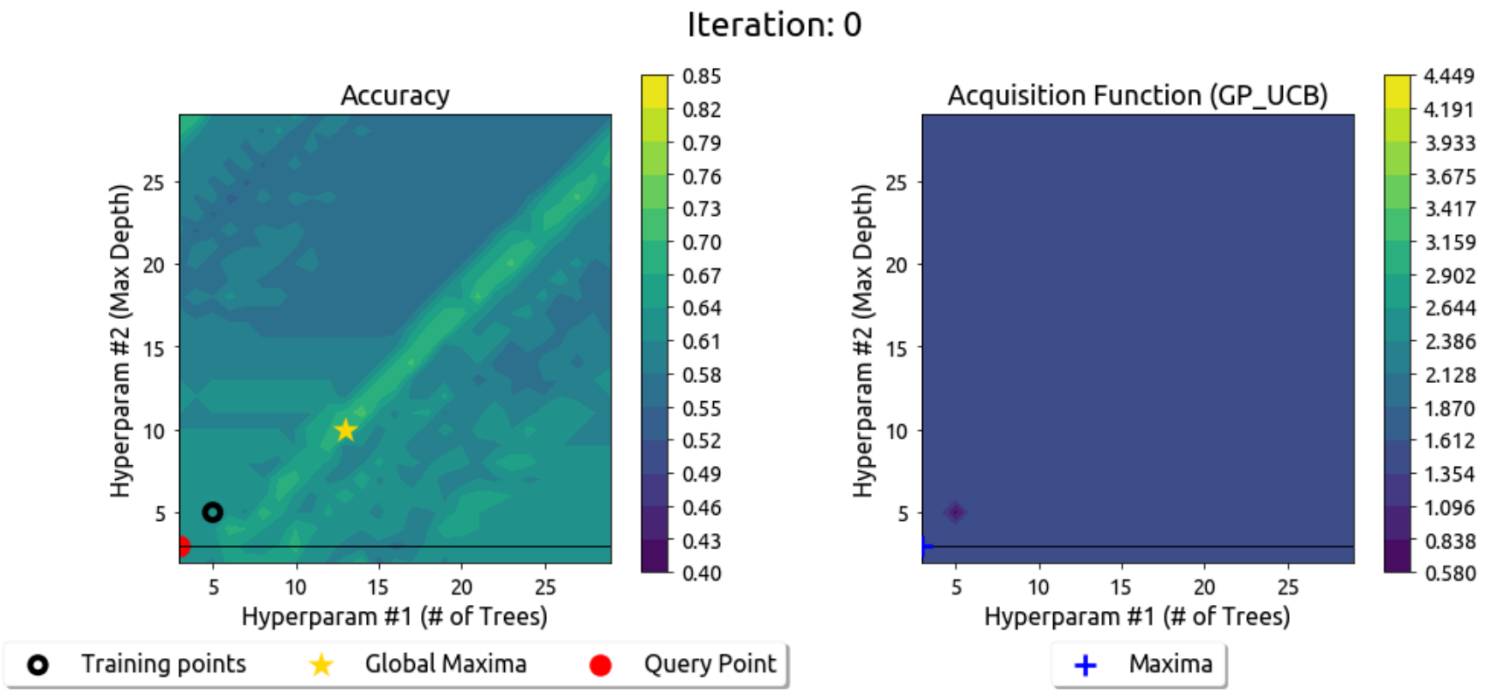
Example 2: Tuning the Random forest hyperparameters

- scikit-learn's moons dataset, two classes and two features (Sapp et al. 2014)
- two hyperparameters:
  - number of Decision Trees we would like to have,
  - the maximum depth for each of those decision trees
- example: Distill

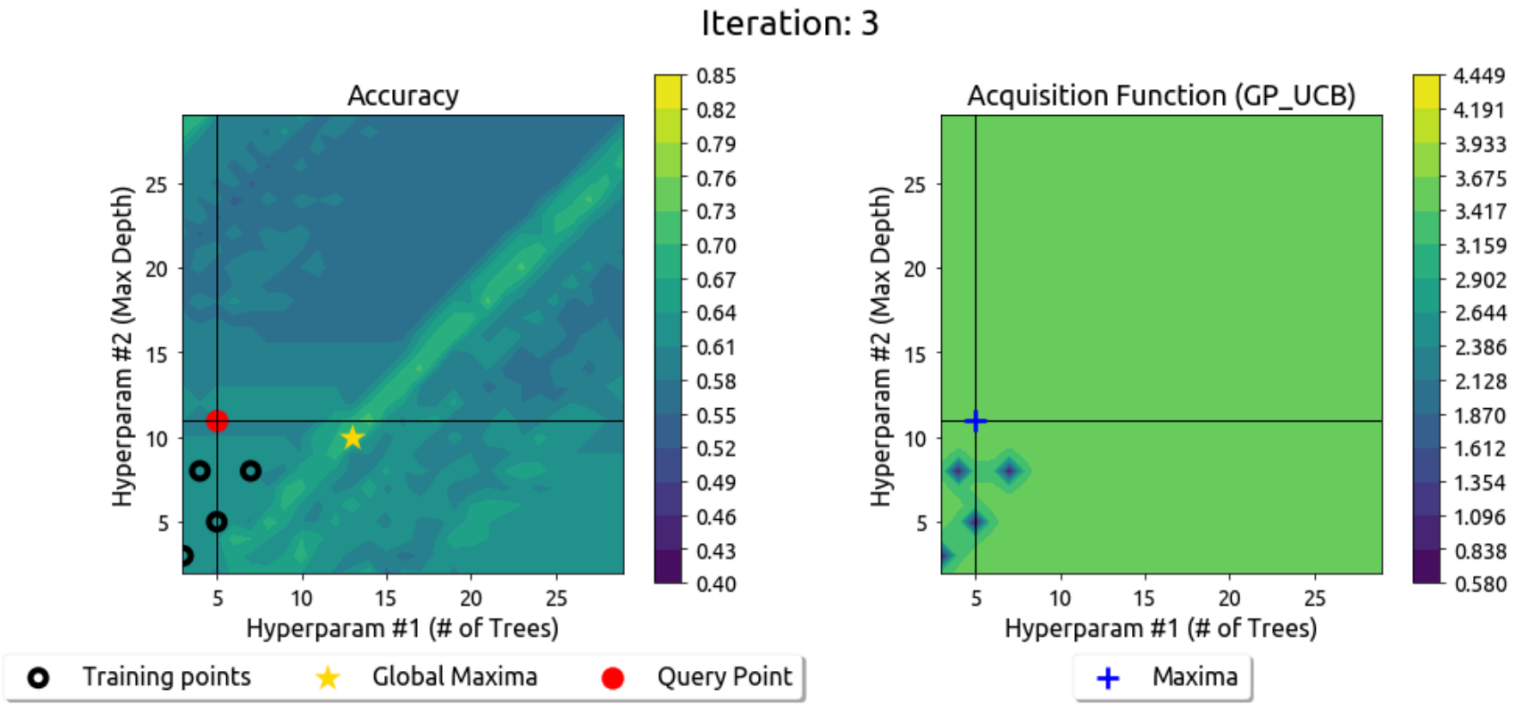
Example 2: Dataset



Example 2: GP-UCB solution after 0 evaluations



Example 2: GP-UCB solution after 3 evaluations



Example 2: GP-UCB solution after 9 evaluations