

# Bayesian Optimization

Partially based on Ryan Adams' Tutorial

<http://tinyurl.com/botutorial>

Recommended reading:

Brochu et al. (2009) [1], Shahriari et al. (2016) [18]

**Marc Deisenroth**

Department of Computing  
Imperial College London

February 20, 2018

# Machine Learning Meta-Challenges

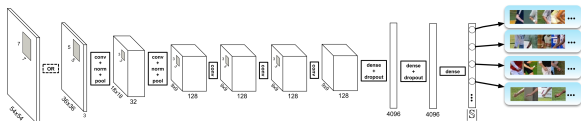
- ▶ Machine learning models are getting more and more complicated
  - ▶▶ Usually more parameters (e.g., deep neural networks)
- ▶ Non-convex optimization methods have many parameters to tune
  - ▶▶ Generally hard to apply modern techniques or reproduce results

# Machine Learning Meta-Challenges

- ▶ Machine learning models are getting more and more complicated
  - ▶▶ Usually more parameters (e.g., deep neural networks)
- ▶ Non-convex optimization methods have many parameters to tune
- ▶▶ Generally hard to apply modern techniques or reproduce results

Automate the selection of critical hyper-parameters  
(see also: [Automated Machine Learning \(AutoML\)](#))

# Example: Deep Neural Networks

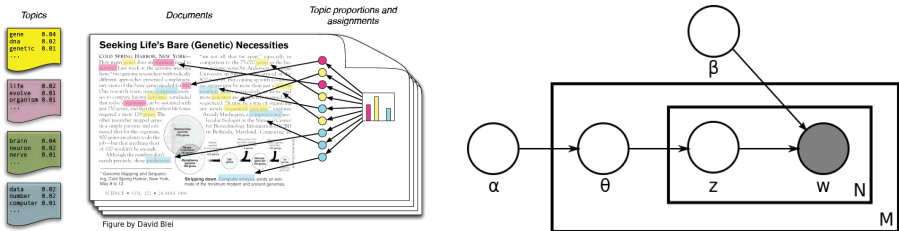


## Huge interest in large neural networks

- ▶ When well-tuned, very successful for visual object identification, speech recognition, computational biology, ...
- ▶ Huge investments by Google, Facebook, Microsoft, etc.
- ▶ **Many choices:** number of layers, weight regularization, layer size, which nonlinearity, batch size, learning rate schedule, stopping conditions

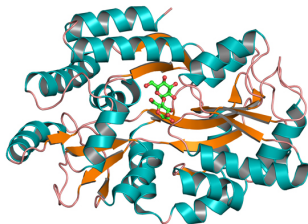
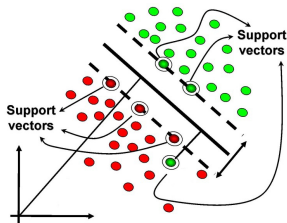


# Example: Online Latent Dirichlet Allocation



- ▶ Hoffman et al. (2010): Approximate inference for **large-scale text analysis with Latent Dirichlet Allocation**
- ▶ Good empirical results when well tuned
- ▶ **Hyper-parameters** tricky to set: Dirichlet parameters, number of topics, learning rate schedule, batch size, vocabulary size, ...

# Example: Classification of DNA Sequences



- ▶ Objective: Predict which DNA sequences will bind with which proteins.
- ▶ Miller et al. (2012): [Latent Structural Support Vector Machine](#)
- ▶ **Hyper-parameters:** margin/slack parameter, entropy parameter, convergence criterion

# Search for Good Hyper-parameters

- ▶ Define an objective function
  - ▶ Usually, we care about generalization performance.
  - ▶ Cross validation to measure parameter quality

# Search for Good Hyper-parameters

- Define an objective function
  - Usually, we care about generalization performance.
  - Cross validation to measure parameter quality
- Standard search procedures:
  - Grid search
  - Random search (very simple, works surprisingly well)
  - Manual tuning
  - Black magic

# Search for Good Hyper-parameters

- ▶ Define an objective function
  - ▶ Usually, we care about generalization performance.
  - ▶ Cross validation to measure parameter quality
- ▶ Standard search procedures:
  - ▶ Grid search
  - ▶ Random search (very simple, works surprisingly well)
  - ▶ Manual tuning
  - ▶ Black magic
- ▶ Painful:
  - ▶ Training may be very expensive (e.g., time or money)
  - ▶ Many training cycles
  - ▶ Possibly noisy

# Alternative Approach: Bayesian Optimization

## Setting

Globally optimize an objective function that is expensive to evaluate (e.g., cross-validation error for a massive neural network)

- ▶ Build a **probabilistic proxy model** for the objective using outcomes of past experiments as training data

# Alternative Approach: Bayesian Optimization

## Setting

Globally optimize an objective function that is expensive to evaluate (e.g., cross-validation error for a massive neural network)

- ▶ Build a **probabilistic proxy model** for the objective using outcomes of past experiments as training data
- ▶ The proxy model is much **cheaper to evaluate** than the original objective

# Alternative Approach: Bayesian Optimization

## Setting

Globally optimize an objective function that is expensive to evaluate (e.g., cross-validation error for a massive neural network)

- ▶ Build a **probabilistic proxy model** for the objective using outcomes of past experiments as training data
- ▶ The proxy model is much **cheaper to evaluate** than the original objective
- ▶ **Optimize cheap proxy** function to determine where to evaluate the true objective next



# Alternative Approach: Bayesian Optimization

## Setting

Globally optimize an objective function that is expensive to evaluate (e.g., cross-validation error for a massive neural network)

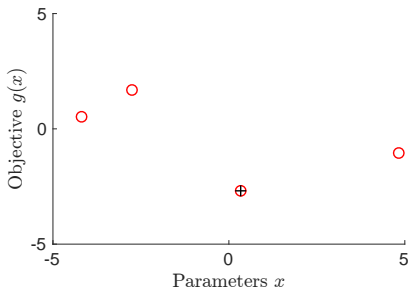
- ▶ Build a **probabilistic proxy model** for the objective using outcomes of past experiments as training data
- ▶ The proxy model is much **cheaper to evaluate** than the original objective
- ▶ **Optimize cheap proxy** function to determine where to evaluate the true objective next
- ▶ Standard proxy: **Gaussian process**

## Setting (2)

- ▶ Objective: Find global minimum of objective function  $g$ :

$$\mathbf{x}_* = \arg \min_x g(\mathbf{x})$$

- ▶ We can evaluate the objective  $g$  pointwise, but do not have an easy functional form or gradients; observations may be noisy
- ▶ **Evaluating  $g$  is costly** (e.g., train a massive deep network)



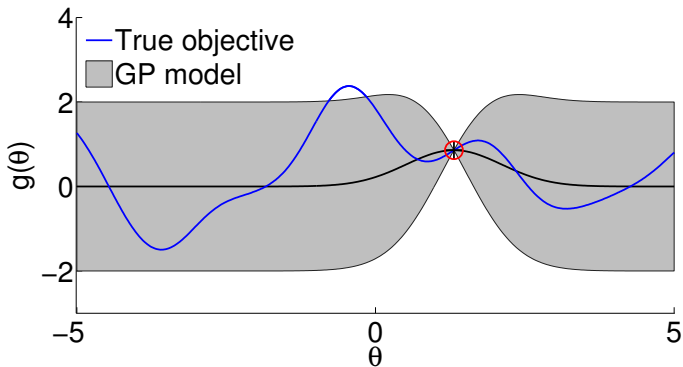
# Key Steps

- ▶ To avoid evaluating  $g$  an excessive number of times, approximate it using a **proxy function**  $\tilde{g}$  (which is cheap to evaluate)
- ▶ Find a **global optimum**  $\tilde{g}(\mathbf{x}_*)$  of **proxy function**  $\tilde{g}$
- ▶ Evaluate true objective  $g$  at  $\mathbf{x}_*$
- ▶ Overall: Evaluate  $g$  only once

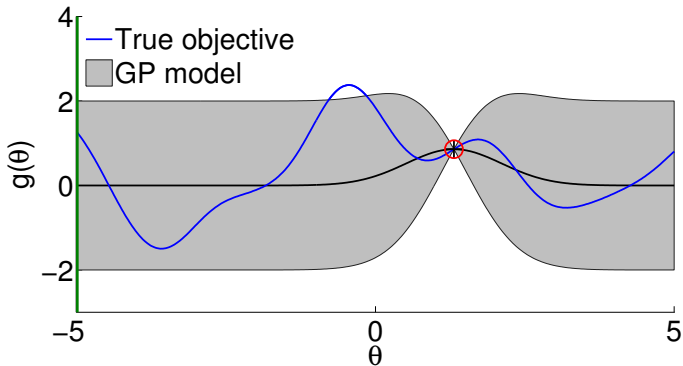
# Key Steps

- ▶ To avoid evaluating  $g$  an excessive number of times, approximate it using a **proxy function**  $\tilde{g}$  (which is cheap to evaluate)
- ▶ Find a **global optimum**  $\tilde{g}(\mathbf{x}_*)$  of **proxy function**  $\tilde{g}$
- ▶ Evaluate true objective  $g$  at  $\mathbf{x}_*$
- ▶ Overall: Evaluate  $g$  only once
- ▶ Works well if  $\tilde{g} \approx g$ .
- ▶ Usually not the case ►► Repeat this cycle and keep updating  $\tilde{g}$

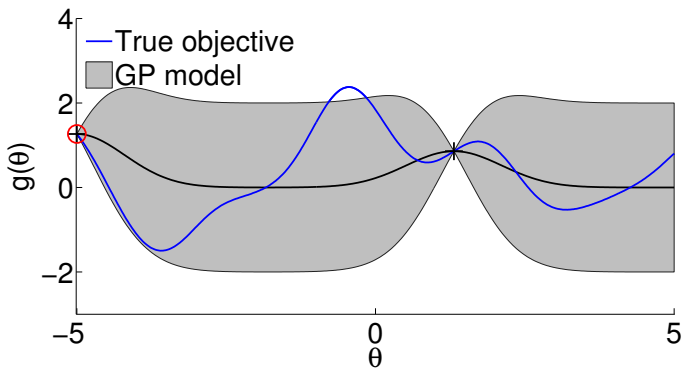
# Bayesian Optimization: Illustration



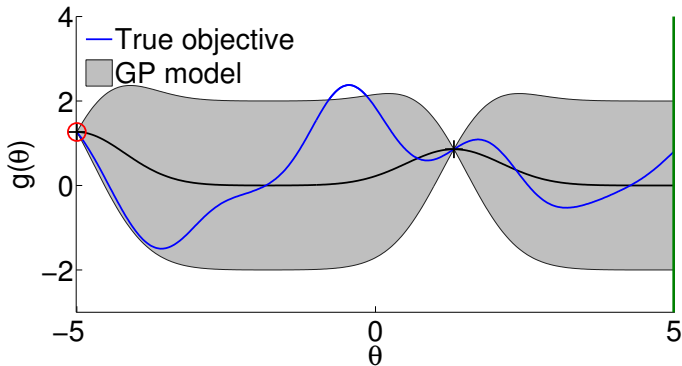
# Bayesian Optimization: Illustration



# Bayesian Optimization: Illustration

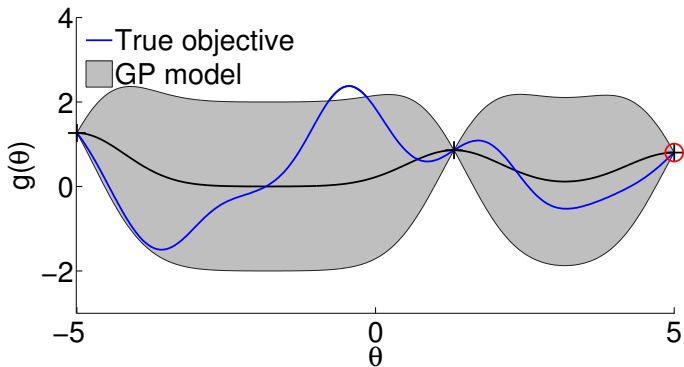


# Bayesian Optimization: Illustration

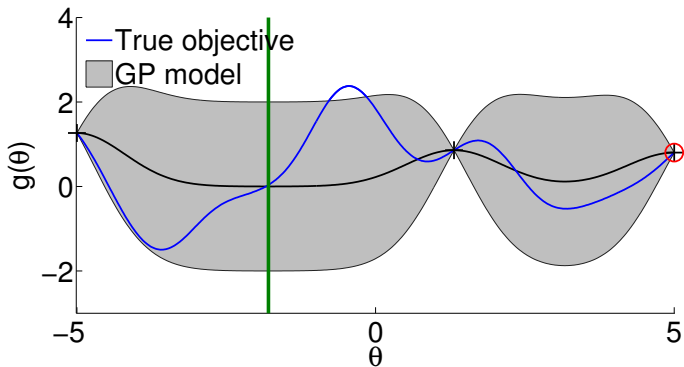




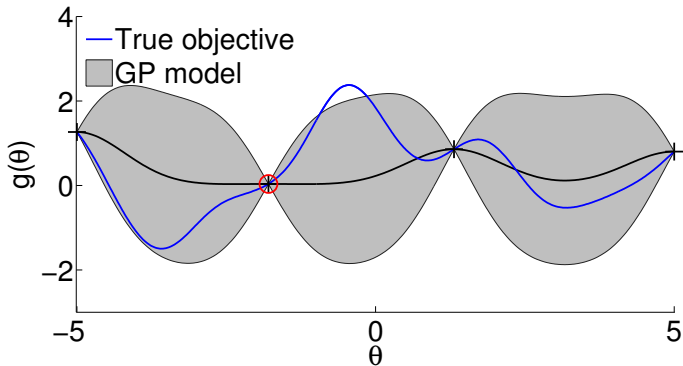
# Bayesian Optimization: Illustration



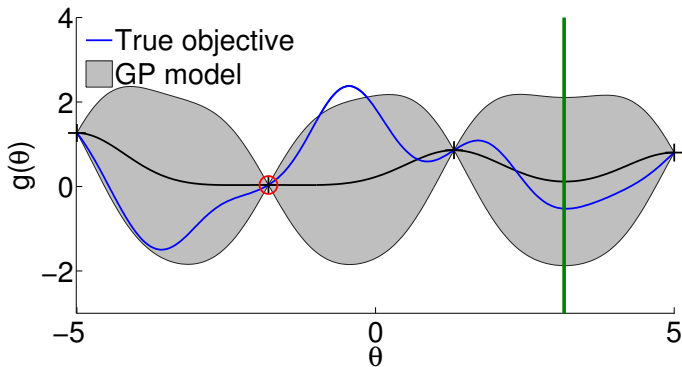
# Bayesian Optimization: Illustration



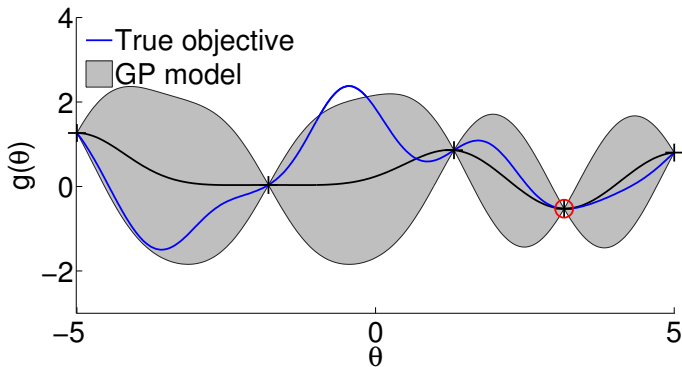
# Bayesian Optimization: Illustration



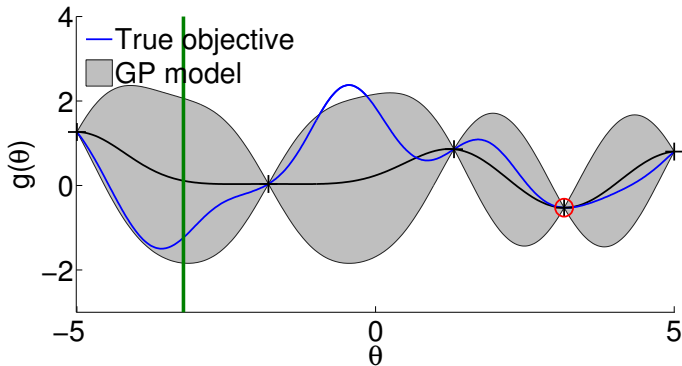
# Bayesian Optimization: Illustration



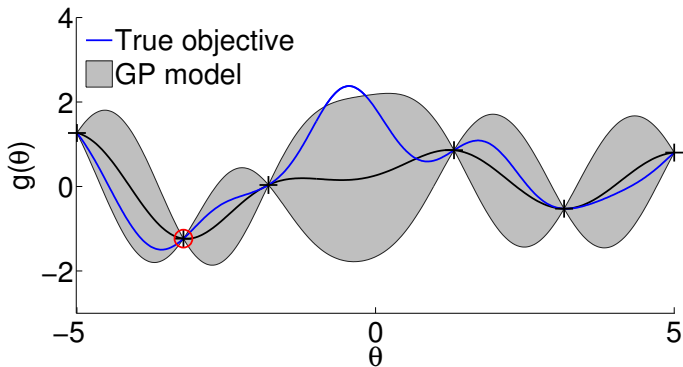
# Bayesian Optimization: Illustration



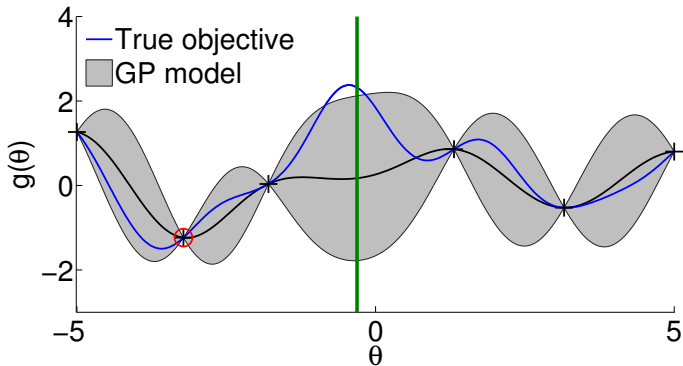
# Bayesian Optimization: Illustration



# Bayesian Optimization: Illustration

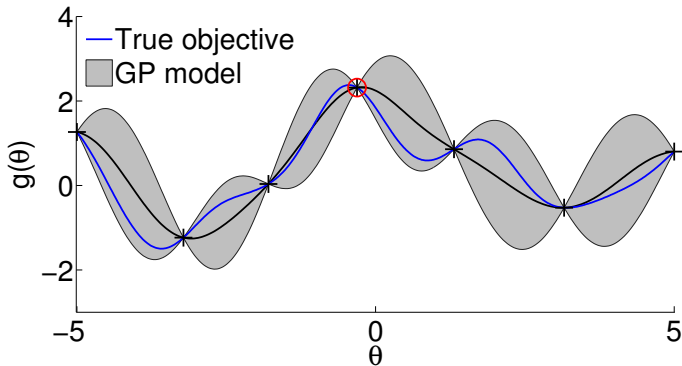


# Bayesian Optimization: Illustration

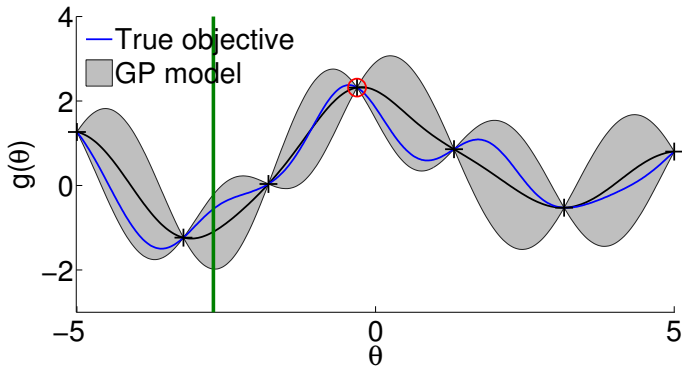




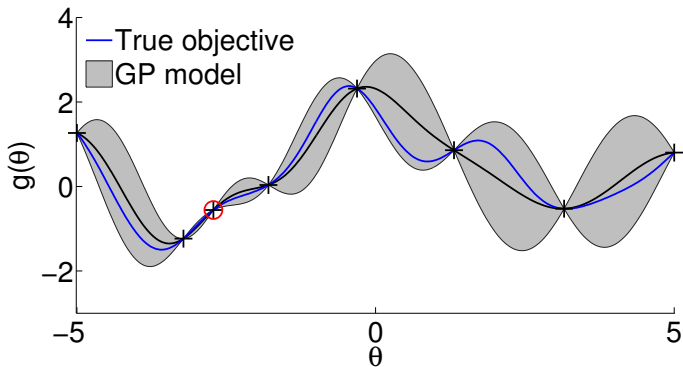
# Bayesian Optimization: Illustration



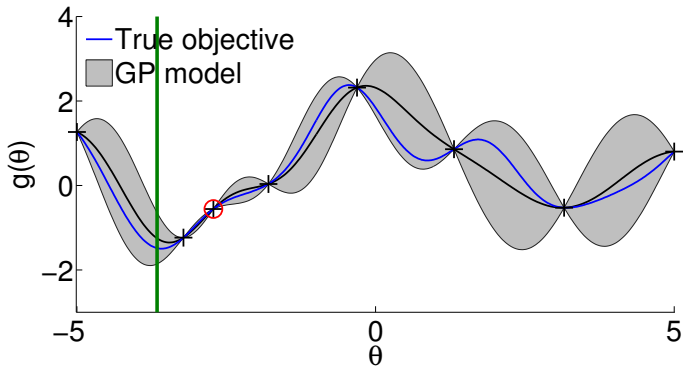
# Bayesian Optimization: Illustration



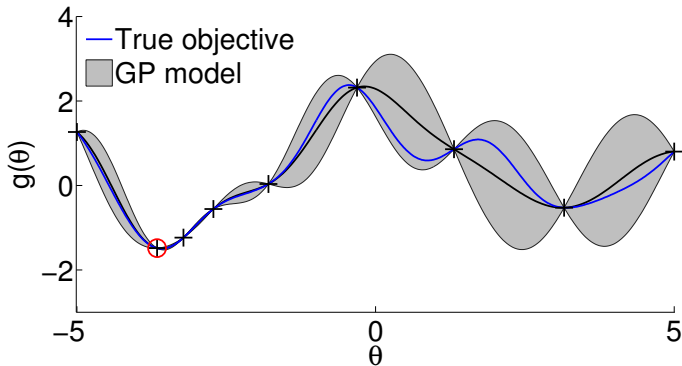
# Bayesian Optimization: Illustration



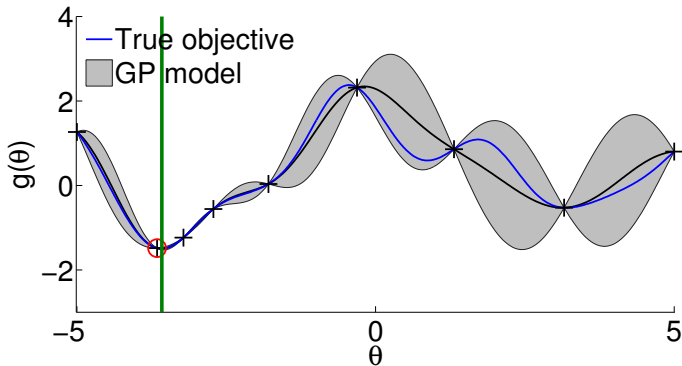
# Bayesian Optimization: Illustration



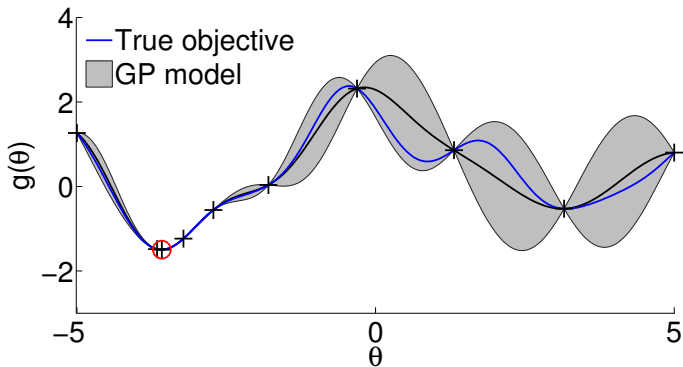
# Bayesian Optimization: Illustration



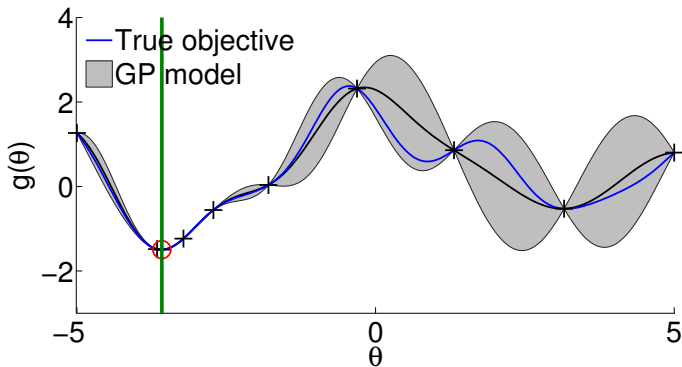
# Bayesian Optimization: Illustration



# Bayesian Optimization: Illustration

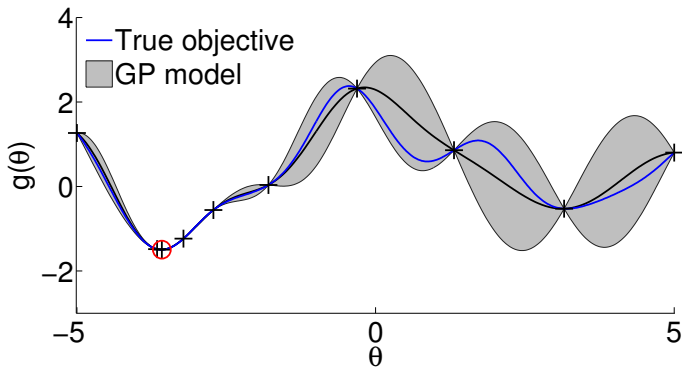


# Bayesian Optimization: Illustration

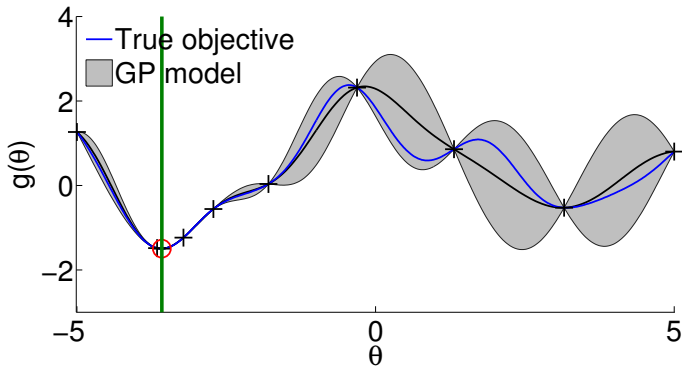




# Bayesian Optimization: Illustration

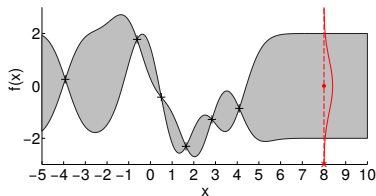


# Bayesian Optimization: Illustration



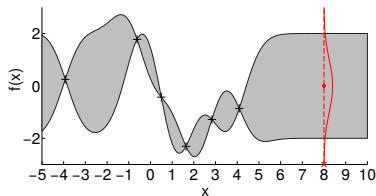
# Choosing the Next Point to Evaluate the True Objective: Acquisition Functions

# Using Uncertainty in Global Optimization



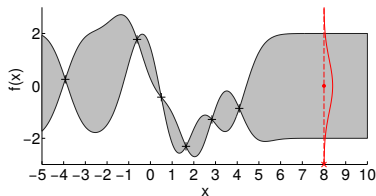
- ▶ Find a good (global) optimum
  - ▶▶ Need to get out of local optima

# Using Uncertainty in Global Optimization



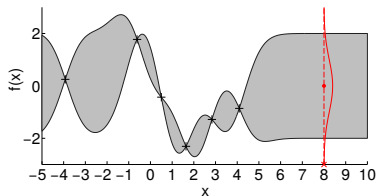
- ▶ Find a good (global) optimum
  - ▶▶ Need to get out of local optima
- ▶ Extrapolate from collected knowledge

# Using Uncertainty in Global Optimization



- ▶ Find a good (global) optimum
  - ▶▶ Need to get out of local optima
- ▶ Extrapolate from collected knowledge
- ▶ GP gives us closed-form means and variances
  - ▶▶ Trade off exploration and exploitation
    - ▶ **Exploration:** Seek places with high variance
    - ▶ **Exploitation:** Seek places with low mean

# Using Uncertainty in Global Optimization



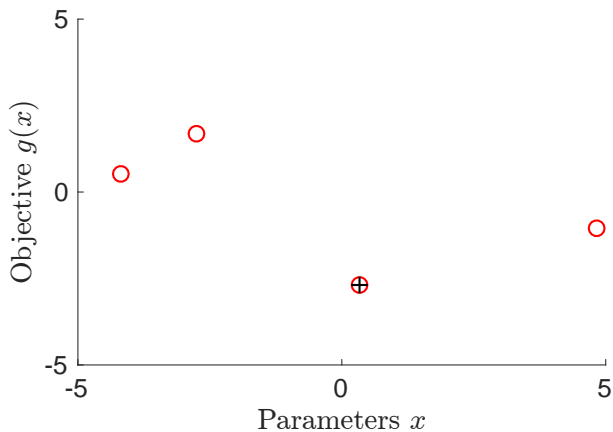
- ▶ Find a good (global) optimum
  - ▶▶ Need to get out of local optima
- ▶ Extrapolate from collected knowledge
- ▶ GP gives us closed-form means and variances
  - ▶▶ Trade off exploration and exploitation
    - ▶ **Exploration:** Seek places with high variance
    - ▶ **Exploitation:** Seek places with low mean
- ▶ **Acquisition function  $\alpha$  trades off exploration and exploitation** for our proxy optimization

## Key Steps (Pseudo-Code)

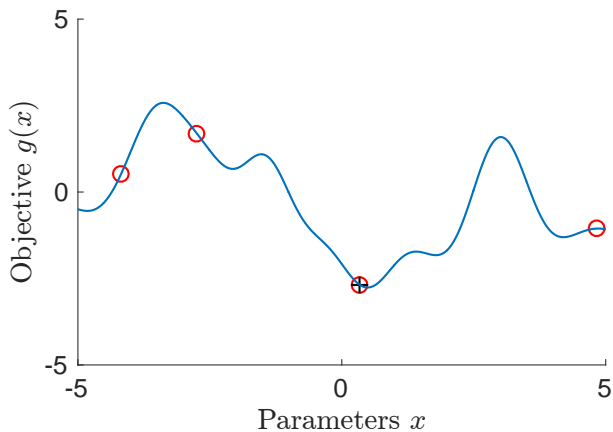
- 1: **Init:** Data set  $\mathcal{D}_0 = \{\mathbf{X}_0, \mathbf{y}_0\}$
- 2: **for** iterations  $t = 1, 2, \dots$  **do**
- 3:     **Update GP** using data  $\mathcal{D}_{t-1}$
- 4:     Select  $\mathbf{x}_t = \arg \max_x \alpha(\mathbf{x})$  by **optimizing acquisition function**
- 5:     Query true objective  $g$  at  $\mathbf{x}_t$
- 6:     Augment data set  $\mathcal{D}_t = \mathcal{D}_{t-1} \cup \{(\mathbf{x}_t, y_t)\}$
- 7: **end for**
- 8: **Return** best input in data set:  $\mathbf{x}^* = \arg \min_x y(\mathbf{x})$



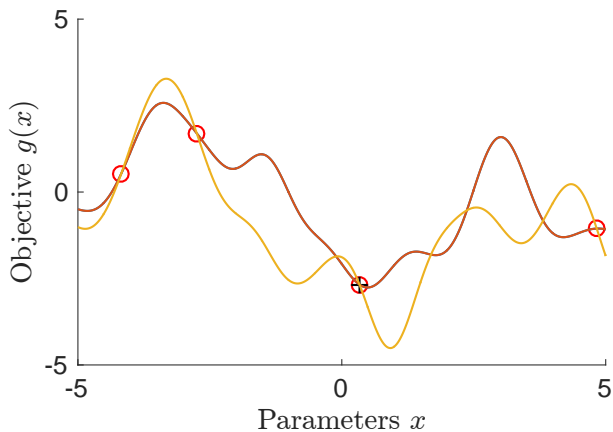
# Where to Evaluate Next?



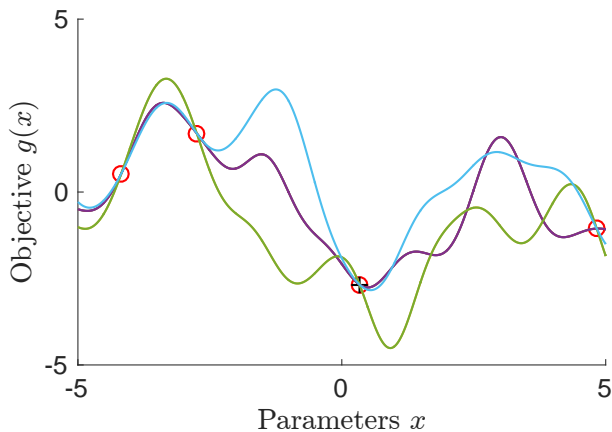
# Where to Evaluate Next?



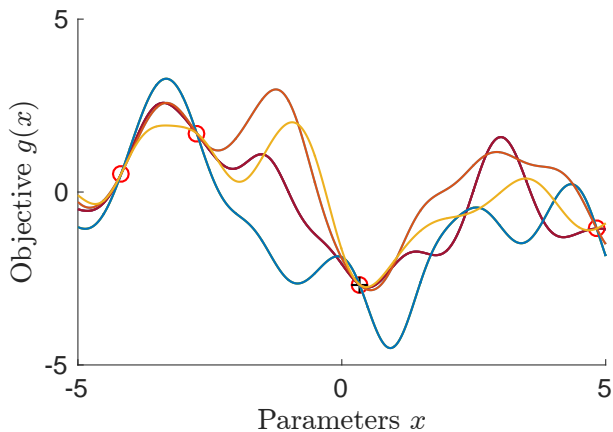
# Where to Evaluate Next?



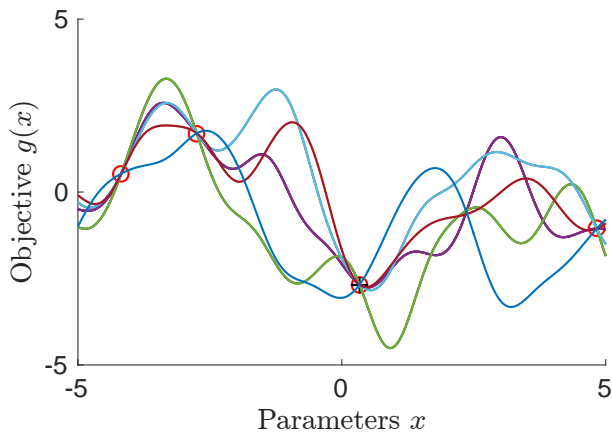
# Where to Evaluate Next?



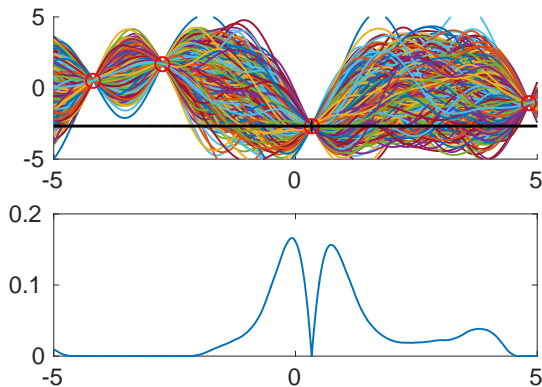
## Where to Evaluate Next?



# Where to Evaluate Next?

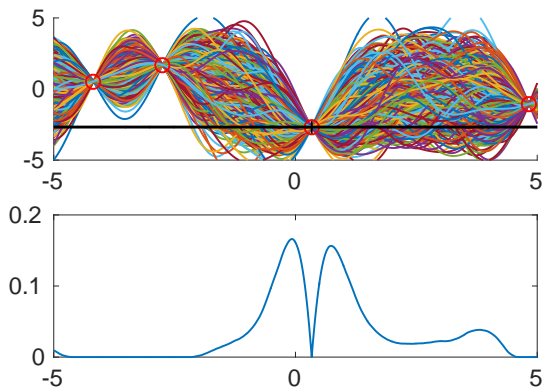


# Where to Evaluate Next to Improve Most?



- Upper panel: Samples from a probabilistic proxy  $\tilde{g}$

## Where to Evaluate Next to Improve Most?



- ▶ Upper panel: Samples from a probabilistic proxy  $\tilde{g}$
- ▶ Lower panel: Corresponding **expected improvement** over the best solution so far (black cross)
  - ▶▶ Evaluate  $g$  at the maximum of the expected improvement



# Closed-Form Acquisition Functions

- ▶ For all  $\mathbf{x} \in \mathbb{R}^D$  the GP posterior gives a predictive mean  $\mu(\mathbf{x})$  variance  $\sigma^2(\mathbf{x})$
- ▶ Define

$$\gamma(\mathbf{x}) = \frac{f(\mathbf{x}_{\text{best}}) - \mu(\mathbf{x})}{\sigma(\mathbf{x})}$$

# Closed-Form Acquisition Functions

- ▶ For all  $\mathbf{x} \in \mathbb{R}^D$  the GP posterior gives a predictive mean  $\mu(\mathbf{x})$  variance  $\sigma^2(\mathbf{x})$

- ▶ Define

$$\gamma(\mathbf{x}) = \frac{f(\mathbf{x}_{\text{best}}) - \mu(\mathbf{x})}{\sigma(\mathbf{x})}$$

- ▶ **Probability of Improvement (Kushner 1964):**

$$\alpha_{\text{PI}}(\mathbf{x}) = \Phi(\gamma(\mathbf{x}))$$

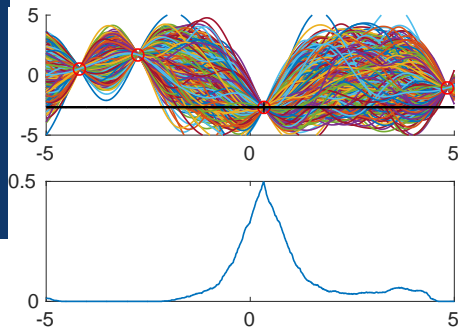
- ▶ **Expected Improvement (Mockus 1978):**

$$\alpha_{\text{EI}}(\mathbf{x}) = \sigma(\mathbf{x}) (\gamma(\mathbf{x}) \Phi(\gamma(\mathbf{x})) + \mathcal{N}(\gamma(\mathbf{x}) | 0, 1))$$

- ▶ **GP Lower Confidence Bound (Srinivas et al., 2010):**

$$\alpha_{\text{LCB}}(\mathbf{x}) = -(\mu(\mathbf{x}) - \kappa\sigma(\mathbf{x})), \quad \kappa > 0$$

# Probability of Improvement (1)

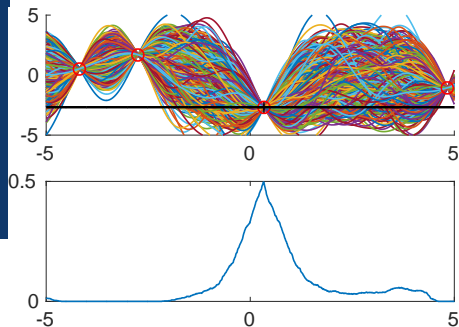


- ▶ **Idea:** Determine the probability that  $\mathbf{x}_*$  leads to a better function value than the currently best one  $f(\mathbf{x}_{\text{best}})$
- ▶ **Sampling-based setting:** Sample  $N$  functions  $f_i$ ; at every input  $\mathbf{x}$  compute a Monte-Carlo estimate

$$\alpha_{\text{PI}}(\mathbf{x}) = p(f(\mathbf{x}) < f(\mathbf{x}_{\text{best}})) \approx \frac{1}{N} \sum_{i=1}^N \delta(f_i(\mathbf{x}) < f(\mathbf{x}_{\text{best}}))$$

▶ Can lead to continued exploitation in an  $\epsilon$ -region around  $\mathbf{x}_{\text{best}}$ .

# Probability of Improvement (1)

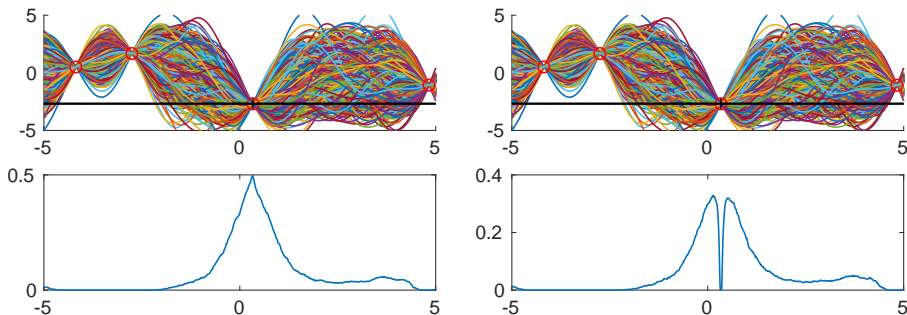


- ▶ **Idea:** Determine the probability that  $\mathbf{x}_*$  leads to a better function value than the currently best one  $f(\mathbf{x}_{\text{best}})$
- ▶ **Sampling-based setting:** Sample  $N$  functions  $f_i$ ; at every input  $\mathbf{x}$  compute a Monte-Carlo estimate

$$\alpha_{\text{PI}}(\mathbf{x}) = p(f(\mathbf{x}) < f(\mathbf{x}_{\text{best}})) \approx \frac{1}{N} \sum_{i=1}^N \delta(f_i(\mathbf{x}) < f(\mathbf{x}_{\text{best}}))$$

- ▶▶ Can lead to continued exploitation in an  $\epsilon$ -region around  $\mathbf{x}_{\text{best}}$ .
- ▶▶ Introduce a “slack variable”  $\zeta$  for more aggressive exploration

## Probability of Improvement (2)



- ▶ Look at a minimum improvement of  $\xi > 0$ :

$$\alpha_{\text{PI}}(\mathbf{x}) = p(f(\mathbf{x}) < f(\mathbf{x}_{\text{best}}) - \xi) \approx \frac{1}{N} \sum_{i=1}^N \delta(f_i(\mathbf{x}) < f(\mathbf{x}_{\text{best}}) - \xi)$$

- ▶ If  $f \sim GP$  and  $p(f(\mathbf{x})) = \mathcal{N}(\mu(\mathbf{x}), \sigma(\mathbf{x}))$ :

$$\alpha_{\text{PI}}(\mathbf{x}) = \Phi(\gamma(\mathbf{x}, \xi)), \quad \gamma(\mathbf{x}, \xi) = \frac{f(\mathbf{x}_{\text{best}}) - \xi - \mu(\mathbf{x})}{\sigma(\mathbf{x})}$$

# Expected Improvement

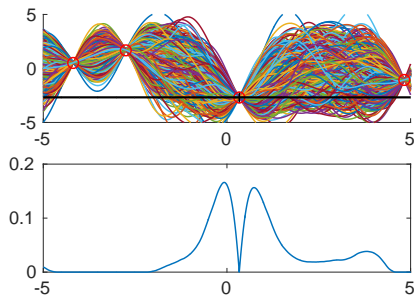
- ▶ **Idea:** Quantify the **amount of improvement**
- ▶ Sampling-based scenario, where  $f_i \sim p(f)$ :

$$\begin{aligned}\alpha_{\text{EI}}(\mathbf{x}) &= \mathbb{E}[\max\{0, f(\mathbf{x}_{\text{best}}) - f(\mathbf{x})\}] \\ &\approx \frac{1}{N} \sum_{i=1}^N \max\{0, f(\mathbf{x}_{\text{best}}) - f_i(\mathbf{x})\}\end{aligned}$$

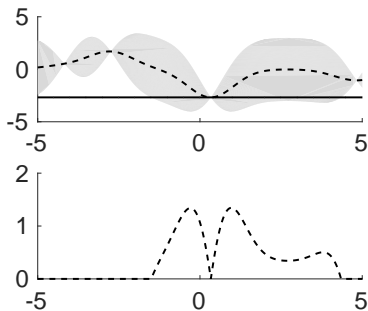
- ▶ If  $f \sim GP$ , we have a closed-form expression:

$$\alpha_{\text{EI}}(\mathbf{x}) = \sigma(\mathbf{x}) \left( \gamma(\mathbf{x}) \Phi(\gamma(\mathbf{x})) + \mathcal{N}(\gamma(\mathbf{x}) \mid 0, 1) \right)$$

- ▶ Slack-variable approach also possible (similar to PI)



## GP-Lower Confidence Bound (1)

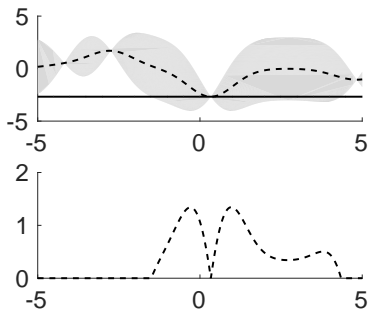


- Use the predictive mean  $\mu(\mathbf{x})$  and variance  $\sigma^2(\mathbf{x})$  of the GP prediction directly for targeted exploration by means of the acquisition function

$$\alpha_{\text{LCB}}(\mathbf{x}_t) = -(\mu(\mathbf{x}_t) - \sqrt{\kappa}\sigma(\mathbf{x}_t))$$

In practice: Use  $\max\{0, f(\mathbf{x}_{\text{best}}) - \alpha_{\text{LCB}}(\mathbf{x}_t)\}$

## GP-Lower Confidence Bound (2)



- ▶ More generally, we can get regret bounds for iteration-dependent  $\kappa$  (Srinivas et al., 2010)

$$\alpha_{\text{LCB}}(\mathbf{x}_t) = -(\mu(\mathbf{x}_t) - \sqrt{\kappa_t}\sigma(\mathbf{x}_t))$$

where  $\kappa_t \in \mathcal{O}(\log t)$  grows with the iteration  $t$

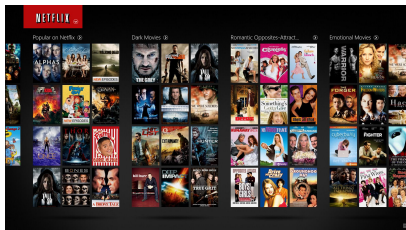


# Optimizing the Acquisition Function

- ▶ Optimizing the acquisition function **requires us to run a global optimizer inside Bayesian optimization**
- ▶ What have we gained?

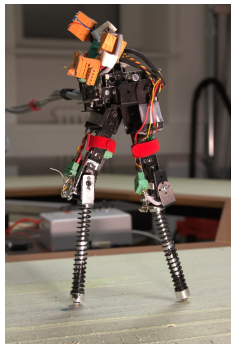
# Optimizing the Acquisition Function

- ▶ Optimizing the acquisition function **requires us to run a global optimizer inside Bayesian optimization**
- ▶ What have we gained?
- ▶ Evaluating the acquisition function is cheap compared to evaluating the true objective
  - ▶▶ We can afford evaluating it many times



# Application Example: Controller Learning in Robotics (Calandra et al., 2016)

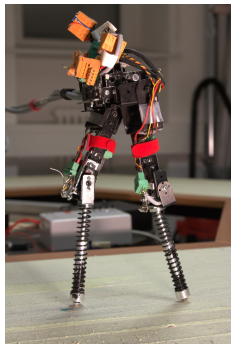
- ▶ Fragile bipedal robot
  - ▶▶ Only few experiments feasible
- ▶ Maximize robustness and walking speed
- ▶ 4 motors:
  - 2 actuated hips + 2 actuated knees
- ▶ Controller implemented as a finite-state-machine (8 parameters)



Calandra et al. (2015)

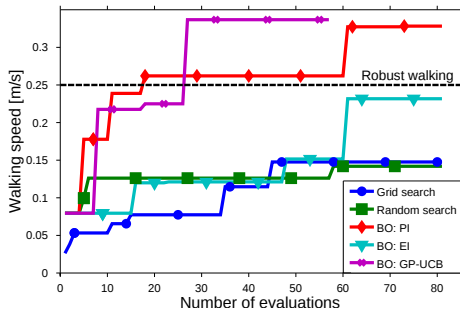
# Application Example: Controller Learning in Robotics (Calandra et al., 2016)

- ▶ Fragile bipedal robot
  - ▶▶ Only few experiments feasible
- ▶ Maximize robustness and walking speed
- ▶ 4 motors:
  - 2 actuated hips + 2 actuated knees
- ▶ Controller implemented as a finite-state-machine (8 parameters)
- ▶ Good parameters found after 80–100 experiments
- ▶ **Substantial speed-up** compared to manual parameter search



Calandra et al. (2015)

# Comparison



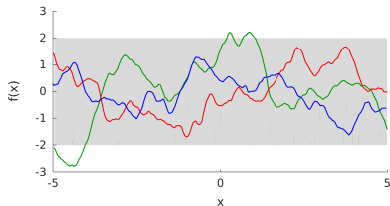
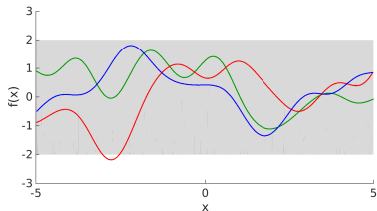
- ▶ Squared exponential covariance function
- ▶ Learned GP hyper-parameters (no MCMC for integrating them out)

# Limitations

- ▶ Getting the function model wrong can be catastrophic
- ▶ Limited scalability in the number of dimensions and/or evaluations of the true objective function

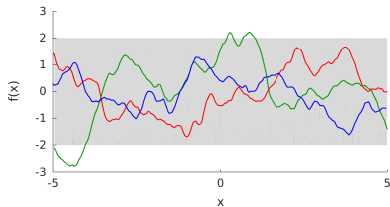
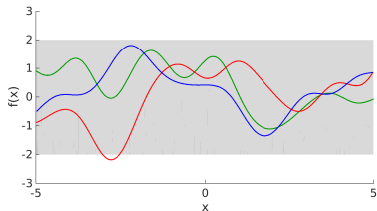
Why?

# Poor Model Choice



- ▶ Covariance function selection is crucial for good performance
  - ▶ Choose a sufficiently flexible and adaptive kernel, e.g., Matérn (but not the squared exponential)

## Poor Model Choice



- ▶ Covariance function selection is crucial for good performance
  - ▶ Choose a sufficiently flexible and adaptive kernel, e.g., Matérn (but not the squared exponential)
- ▶ Nice side-effect of Matérn: Exploration is more encouraged than with the squared exponential kernel



# Choosing Covariance Functions

- ▶ Structured SVM for Protein Motif Finding (Miller et al., 2012)
- ▶ Optimize hyper-parameters of SSVM using BO (Snoek et al., 2012)

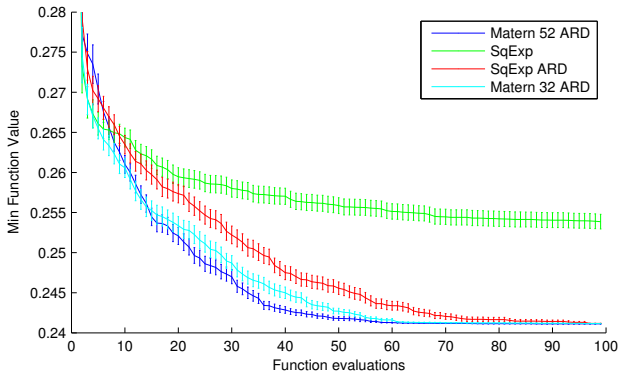


Figure: Figure from Snoek et al. (2012)

# Gaussian Process Hyper-Parameters

- ▶ Empirical Bayes (maximize the marginal likelihood) can fail **horribly**, especially in the early stages of Bayesian optimization when we have only a few data points

# Gaussian Process Hyper-Parameters

- ▶ Empirical Bayes (maximize the marginal likelihood) can fail **horribly**, especially in the early stages of Bayesian optimization when we have only a few data points
- ▶ Solution: Integrate out the GP hyper-parameters  $\theta$  by **Markov Chain Monte Carlo (MCMC)** sampling (e.g., slice sampling)

# Gaussian Process Hyper-Parameters

- ▶ Empirical Bayes (maximize the marginal likelihood) can fail **horribly**, especially in the early stages of Bayesian optimization when we have only a few data points
- ▶ Solution: Integrate out the GP hyper-parameters  $\theta$  by **Markov Chain Monte Carlo (MCMC)** sampling (e.g., slice sampling)
- ▶ Look at **integrated acquisition function**

$$\begin{aligned}\alpha(\mathbf{x}) &= \mathbb{E}_{\theta}[\alpha(\mathbf{x}, \theta)] = \int \alpha(\mathbf{x}, \theta) p(\theta) d\theta \\ &\approx \frac{1}{K} \sum_{k=1}^K \alpha(\mathbf{x}, \theta^{(k)}), \quad \theta^{(k)} \sim \underbrace{p(\theta | \mathbf{X}_n, \mathbf{y}_n)}_{\text{hyper-parameter posterior}}\end{aligned}$$

# Integrating out GP Hyper-parameters

- ▶ Online LDA (Hoffman et al., 2010) for topic modeling
- ▶ Two critical hyper-parameters that control the learning rate learned by BO (Snoek et al., 2012)

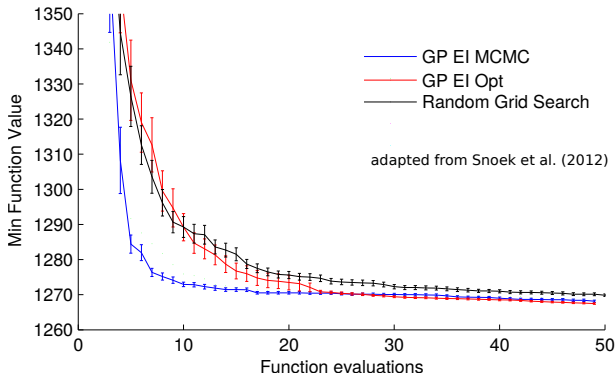
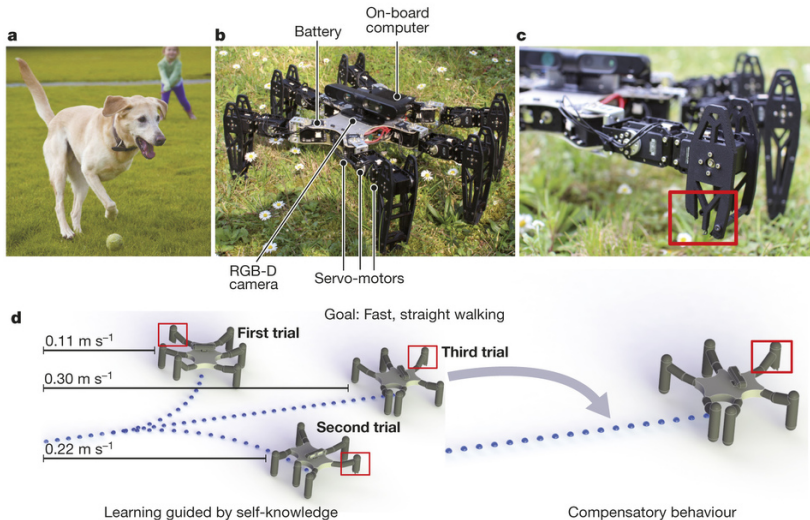


Figure: Figure from Snoek et al. (2012)

# Robots That Learn to Recover from Damage



Cully et al. (2015)

## Further Topics in BO

- ▶ **Entropy-based acquisition functions:** Directly describe the distribution over the best input location (Hennig & Schuler, 2012; Hernández-Lobato et al., 2014)
- ▶ **Non-myopic** Bayesian optimization (Osborne et al., 2009)
- ▶ **High-dimensional** optimization (Wang et al., 2016)
- ▶ **Large-scale** Bayesian optimization (Hutter et al., 2014)
- ▶ **Non-GP** Bayesian optimization (Hutter et al., 2014; Snoek et al., 2015)
- ▶ **Constraints** (e.g., Gelbart et al., 2014)
- ▶ **Automated machine learning** (e.g., Feurer et al., 2015)
- ▶ **Multi-tasking, parallelizing, resource allocation, ...** (e.g., Swersky et al., 2014; Snoek et al., 2012)

# Software

- ▶ **BayesOpt** <https://bitbucket.org/rmcantini/bayesopty/> (Martinez-Cantin, 2014)
- ▶ **Spearmint** <https://github.com/HIPS/Spearmint>
- ▶ **Pybo** <https://github.com/mwhoffman/pybo> (Hoffman & Shariari)
- ▶ **GPyOpt** <https://github.com/SheffieldML/GPyOpt> (Gonzalez et al.)
- ▶ Matlab toolbox (bayesopty)



# References I

- [1] E. Brochu, V. M. Cora, and N. de Freitas. A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning. Technical Report TR-2009-023, Department of Computer Science, University of British Columbia, 2009.
- [2] R. Calandra, A. Seyfarth, J. Peters, and M. P. Deisenroth. Bayesian Optimization for Learning Gaits under Uncertainty. *Annals in Mathematics and Artificial Intelligence*, pages 1–19, June 2015.
- [3] A. Cully, J. Clune, D. Tarapore, and J.-B. Mouret. Robots that can adapt like animals. *Nature*, pages 1–26, 2015.
- [4] M. Feurer, A. Klein, K. Eggenberger, J. Springenberg, M. Blum, and F. Hutter. Efficient and Robust Automated Machine Learning. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, pages 2962–2970. Curran Associates, Inc., 2015.
- [5] M. Gelbart, J. Snoek, and R. P. Adams. Bayesian Optimization with Unknown Constraints. In *International Conference on Uncertainty in Artificial Intelligence*, pages 1–14, 2014.
- [6] P. Hennig and C. J. Schuler. Entropy Search for Information-Efficient Global Optimization. *Journal of Machine Learning Research*, 13:1809–1837, 2012.
- [7] J. M. Hernández-Lobato, M. W. Hoffman, and Z. Ghahramani. Predictive Entropy Search for Efficient Global Optimization of Black-box Functions. In *Advances in Neural Information Processing Systems*, pages 1–9, 2014.
- [8] M. D. Hoffman, D. M. Blei, and F. Bach. Online Learning for Latent Dirichlet Allocation. *Advances in Neural Information Processing Systems*, 23:1–9, 2010.
- [9] F. Hutter, H. Hoos, and K. Leyton-Brown. An Efficient Approach for Assessing Hyperparameter Importance. In *Proceedings of International Conference on Machine Learning*, pages 754–762, June 2014.
- [10] D. R. Jones, M. Schonlau, and W. J. Welch. Efficient Global Optimization of Expensive Black-Box Functions. *Journal of Global Optimization*, 13(4):455–492, Dec. 1998.
- [11] H. J. Kushner. A New Method of Locating the Maximum Point of an Arbitrary Multipeak Curve in the Presence of Noise. *Journal of Basic Engineering*, 86:97, 1964.
- [12] D. Lizotte. *Practical Bayesian Optimization*. PhD thesis, University of Alberta, Edmonton, Alberta, 2008.

## References II

- [13] R. Martinez-Cantin. BayesOpt: A Bayesian Optimization Library for Nonlinear Optimization, Experimental Design and Bandits. *Journal of Machine Learning Research*, 15:3915–3919, 2014.
- [14] R. Martinez-Cantin, N. de Freitas, A. Doucet, and J. Castellanos. Active Policy Learning for Robot Planning and Exploration under Uncertainty. In *Proceedings of Robotics: Science and Systems III*, Atlanta, GA, USA, June 2007.
- [15] K. Miller, M. P. Kumar, B. Packer, D. Goodman, and D. Koller. Max-Margin Min-Entropy Models. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, volume 22, pages 779–787, 2012.
- [16] J. Mockus, V. Tiesis, and A. Zilinska. The Application of Bayesian Methods for Seeking the Extremum. *Towards Global Optimization*, 2:117–129, 1978.
- [17] M. A. Osborne, R. Garnett, and S. J. Roberts. Gaussian Processes for Global Optimization. In *Proceedings of the International Conference on Learning and Intelligent Optimization*, 2009.
- [18] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas. Taking the Human out of the Loop: A Review of Bayesian Optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.
- [19] J. Snoek, H. Larochelle, and R. P. Adams. Practical Bayesian Optimization of Machine Learning Algorithms. In *Advances in Neural Information Processing Systems (NIPS)*, 2012.
- [20] J. Snoek, O. Rippel, K. Swersky, R. Kiros, N. Satish, N. Sundaram, M. M. A. Patwary, Prabhat, and R. P. Adams. Scalable Bayesian Optimization Using Deep Neural Networks. In *Proceedings of the International Conference on Machine Learning*, 2015.
- [21] J. T. Springenberg, A. Klein, S. Falkner, and F. Hutter. Bayesian Optimization with Robust Bayesian Neural Networks. In *Advances in Neural Information Processing Systems 29*, December 2016.
- [22] N. Srinivas, A. Krause, S. Kakade, and M. Seeger. Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design. In *Proceedings of the International Conference on Machine Learning*, 2010.
- [23] K. Swersky, J. Snoek, and R. P. Adams. Freeze-Thaw Bayesian Optimization. Technical report, 2014.
- [24] D. Ulmasov, C. Baroukh, B. Chachuat, M. P. Deisenroth, and R. Misener. Bayesian Optimization with Dimension Scheduling: Application to Biological Systems. In *Proceedings of the European Symposium on Computer Aided Process Engineering*, 2016.
- [25] Z. Wang, F. Hutter, M. Zoghi, D. Matheson, and N. De Freitas. Bayesian Optimization in a Billion Dimensions via Random Embeddings. In *Journal of Artificial Intelligence Research*, volume 55, pages 361–367, 2016.