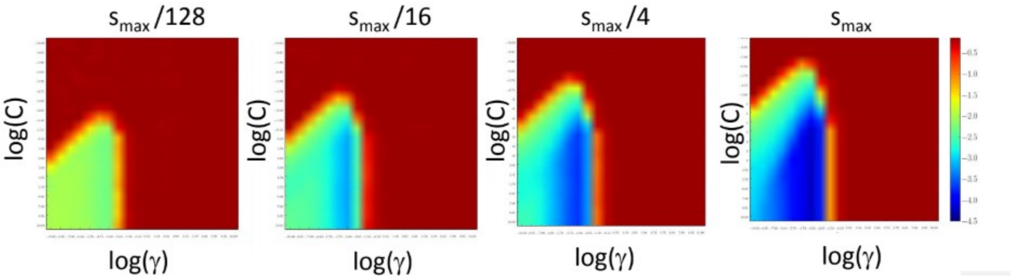
Speedup Techniques for Hyperparameter Optimization

Multi-fidelity Bayesian optimization

Bernd Bischl Frank Hutter Lars Kotthoff
Marius Lindauer Joaquin Vanschoren

Motivating example

- Performance of an SVM on MNIST and subsets of it:



  - Computational cost grows quadratically in dataset size z
  - Error shrinks smoothly with z
- Evaluations on the smallest subset (about 400 data points) cost $10\,000\times$ less than on the full data set

Idea of Multi-fidelity Bayesian optimization [Kandasamy et al. 2017; Klein et al. 2016]

- Recall: standard Bayesian optimization uses a model $\hat{c}(\boldsymbol{\lambda}) \approx y$ to select the next $\boldsymbol{\lambda}$

Idea of Multi-fidelity Bayesian optimization [Kandasamy et al. 2017; Klein et al. 2016]

- Recall: standard Bayesian optimization uses a model $\hat{c}(\boldsymbol{\lambda}) \approx y$ to select the next $\boldsymbol{\lambda}$
- Multi-fidelity Bayesian optimization uses a model $\hat{c}(\boldsymbol{\lambda}, z) \approx y$ to select the next $(\boldsymbol{\lambda}, z)$
  - Here, $z \in \mathcal{Z}$ is the fidelity; $\mathcal{Z}$ is the fidelity space, e.g., $\mathcal{Z} = [1, N_\bullet] \times [1, T_\bullet]$

Idea of Multi-fidelity Bayesian optimization [Kandasamy et al. 2017; Klein et al. 2016]

- Recall: standard Bayesian optimization uses a model $\hat{c}(\boldsymbol{\lambda}) \approx y$ to select the next $\boldsymbol{\lambda}$
- Multi-fidelity Bayesian optimization uses a model $\hat{c}(\boldsymbol{\lambda}, z) \approx y$ to select the next $(\boldsymbol{\lambda}, z)$
  - Here, $z \in \mathcal{Z}$ is the fidelity; $\mathcal{Z}$ is the fidelity space, e.g., $\mathcal{Z} = [1, N_\bullet] \times [1, T_\bullet]$
- Denoting $z_\bullet$ as the maximum fidelity (e.g., $z_\bullet = [N_\bullet, T_\bullet]$), our goal is to find:

$$\boldsymbol{\lambda}^* = \arg\min_{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}} f(\boldsymbol{\lambda}) = \arg\min_{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}} \hat{c}(\boldsymbol{\lambda}, z_\bullet)$$

Idea of Multi-fidelity Bayesian optimization [Kandasamy et al. 2017; Klein et al. 2016]

- Recall: standard Bayesian optimization uses a model $\hat{c}(\boldsymbol{\lambda}) \approx y$ to select the next $\boldsymbol{\lambda}$
- Multi-fidelity Bayesian optimization uses a model $\hat{c}(\boldsymbol{\lambda}, z) \approx y$ to select the next $(\boldsymbol{\lambda}, z)$
  - Here, $z \in \mathcal{Z}$ is the fidelity; $\mathcal{Z}$ is the fidelity space, e.g., $\mathcal{Z} = [1, N_\bullet] \times [1, T_\bullet]$
- Denoting $z_\bullet$ as the maximum fidelity (e.g., $z_\bullet = [N_\bullet, T_\bullet]$), our goal is to find:

$$\boldsymbol{\lambda}^* = \arg\min_{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}} f(\boldsymbol{\lambda}) = \arg\min_{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}} \hat{c}(\boldsymbol{\lambda}, z_\bullet)$$

- Implications for Bayesian optimization
  - Model $\hat{c}$ needs to be good at extrapolating from small to large z
  - Acquisition function now also needs to select z (i.e., take into account cost of evaluations)

Entropy Search: Reminder

- Define the p_{min} distribution given data $\mathcal{D}$:

$$p_{\text{min}}(\boldsymbol{\lambda}^* \mid \mathcal{D}) := p(\boldsymbol{\lambda}^* \in \arg\min_{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}} f(\boldsymbol{\lambda}) \mid \mathcal{D})$$

- Entropy search aims to minimize the entropy $\mathcal{H}[p_{\text{min}}]$ [Hennig and Schuler. 2012]
  - It aims to be maximally certain about the location of $\boldsymbol{\lambda}^*$

Entropy Search: Reminder

- Define the p_{min} distribution given data $\mathcal{D}$:

$$p_{\text{min}}(\boldsymbol{\lambda}^* \mid \mathcal{D}) := p(\boldsymbol{\lambda}^* \in \arg\min_{\boldsymbol{\lambda}\in\boldsymbol{\Lambda}} f(\boldsymbol{\lambda}) \mid \mathcal{D})$$

- Entropy search aims to minimize the entropy $\mathcal{H}[p_{\text{min}}]$ [Hennig and Schuler. 2012]
  - It aims to be maximally certain about the location of $\boldsymbol{\lambda}^*$

- In a nutshell: select $\boldsymbol{\lambda}$ that maximizes the following acquisition function:

$$u_{ES}(\boldsymbol{\lambda}) := \mathcal{H}[p_{\text{min}}(\cdot \mid \mathcal{D})] - \mathbb{E}_{p(\tilde{c}|\boldsymbol{\lambda},\mathcal{D})}\left[\mathcal{H}[p_{\text{min}}(\cdot \mid \mathcal{D} \cup \langle \boldsymbol{\lambda}, \tilde{c}\rangle)]\right]$$

Entropy Search for Multi-Fidelity Optimization [Klein et al. 2017]

- We now care about the p_{min} distribution for the maximal budget $z_\bullet$:

$$p_{\text{min}}(\boldsymbol{\lambda}^* \mid \mathcal{D}) := p(\boldsymbol{\lambda}^* \in \arg\min_{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}} f(\boldsymbol{\lambda}, z_\bullet) \mid \mathcal{D})$$

- We still want to minimize the entropy $\mathcal{H}[p_{\text{min}}]$

Entropy Search for Multi-Fidelity Optimization [Klein et al. 2017]

- We now care about the p_{min} distribution for the maximal budget $z_\bullet$:

$$p_{\mathsf{min}}(\boldsymbol{\lambda}^* \mid \mathcal{D}) := p(\boldsymbol{\lambda}^* \in \arg\min_{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}} f(\boldsymbol{\lambda}, z_\bullet) \mid \mathcal{D})$$

- We still want to minimize the entropy $\mathcal{H}[p_{\mathsf{min}}]$
- Now we aim for the biggest reduction in entropy per time spent
  - Now we don't model only f, but also the cost $c(\boldsymbol{\lambda}, z)$
  - We choose the next $(\boldsymbol{\lambda}, z)$ by maximizing:

$$u_{ES}(\boldsymbol{\lambda}, z \mid \mathcal{D}) := \mathbb{E}_{p(\tilde{c} \mid (\boldsymbol{\lambda}, z), \mathcal{D})} \left[\frac{\mathcal{H}[p_{\mathsf{min}}(\cdot \mid \mathcal{D})] - \mathcal{H}[p_{\mathsf{min}}(\cdot \mid \mathcal{D} \cup \langle (\boldsymbol{\lambda}, z), \tilde{c} \rangle]}{c(\boldsymbol{\lambda}, z)} \right]$$

Entropy Search for Multi-Fidelity Optimization [Klein et al. 2017]

- The entire algorithm iterates the following 2 steps until time is up:
  1. Select $(\boldsymbol{\lambda}, z)$ by maximizing:

$$u_{ES}(\boldsymbol{\lambda}, z \mid \mathcal{D}) := \mathbb{E}_{p(\tilde{c} \mid (\boldsymbol{\lambda}, z), \mathcal{D})} \left[\frac{\mathcal{H}[p_{\text{min}}(\cdot \mid \mathcal{D})] - \mathcal{H}[p_{\text{min}}(\cdot \mid \mathcal{D} \cup \langle (\boldsymbol{\lambda}, z), \tilde{c} \rangle}{c(\boldsymbol{\lambda}, z)} \right]$$

  2. Observe performance $f(\boldsymbol{\lambda}, z)$ and cost $c(\boldsymbol{\lambda}, z)$ and update models for f and c

Entropy Search for Multi-Fidelity Optimization [Klein et al. 2017]

- The entire algorithm iterates the following 2 steps until time is up:
  1. Select $(\boldsymbol{\lambda}, z)$ by maximizing:

$$u_{ES}(\boldsymbol{\lambda}, z \mid \mathcal{D}) := \mathbb{E}_{p(\tilde{c}|(\boldsymbol{\lambda},z),\mathcal{D})}\left[\frac{\mathcal{H}[p_{\min}(\cdot \mid \mathcal{D})] - \mathcal{H}[p_{\min}(\cdot \mid \mathcal{D} \cup \langle(\boldsymbol{\lambda}, z), \tilde{c}\rangle}{c(\boldsymbol{\lambda}, z)}\right]$$

  2. Observe performance $f(\boldsymbol{\lambda}, z)$ and cost $c(\boldsymbol{\lambda}, z)$ and update models for f and c
- The algorithm originally focussed on data subsets and is therefore dubbed Fabolas: "Fast Bayesian Optimization on Large Datasets"

Entropy Search for Multi-Fidelity Optimization [Klein et al. 2017]

- The entire algorithm iterates the following 2 steps until time is up:
  1. Select $(\boldsymbol{\lambda}, z)$ by maximizing:

$$u_{ES}(\boldsymbol{\lambda}, z \mid \mathcal{D}) := \mathbb{E}_{p(\tilde{c} \mid (\boldsymbol{\lambda}, z), \mathcal{D})} \left[\frac{\mathcal{H}[p_{\min}(\cdot \mid \mathcal{D})] - \mathcal{H}[p_{\min}(\cdot \mid \mathcal{D} \cup \langle (\boldsymbol{\lambda}, z), \tilde{c} \rangle}{c(\boldsymbol{\lambda}, z)} \right]$$

  2. Observe performance $f(\boldsymbol{\lambda}, z)$ and cost $c(\boldsymbol{\lambda}, z)$ and update models for f and c
- The algorithm originally focussed on data subsets and is therefore dubbed Fabolas: "Fast Bayesian Optimization on Large Datasets"

Advantages

- Conceptually beautiful
- 1 000-fold speedups for optimizing SVMs on MNIST

Entropy Search for Multi-Fidelity Optimization [Klein et al. 2017]

- The entire algorithm iterates the following 2 steps until time is up:
  1. Select $(\boldsymbol{\lambda}, z)$ by maximizing:

$$u_{ES}(\boldsymbol{\lambda}, z \mid \mathcal{D}) := \mathbb{E}_{p(\tilde{c} \mid (\boldsymbol{\lambda}, z), \mathcal{D})} \left[\frac{\mathcal{H}[p_{\min}(\cdot \mid \mathcal{D})] - \mathcal{H}[p_{\min}(\cdot \mid \mathcal{D} \cup \langle (\boldsymbol{\lambda}, z), \tilde{c} \rangle]}{c(\boldsymbol{\lambda}, z)} \right]$$

  2. Observe performance $f(\boldsymbol{\lambda}, z)$ and cost $c(\boldsymbol{\lambda}, z)$ and update models for f and c
- The algorithm originally focussed on data subsets and is therefore dubbed Fabolas: "Fast Bayesian Optimization on Large Datasets"

Advantages

- Conceptually beautiful
- 1 000-fold speedups for optimizing SVMs on MNIST

Disadvantages

- Scalability of GPs is a big problem (limits size of initial design)
- Limited applicability of Gaussian processes

Questions to Answer for Yourself / Discuss with Friends

- Discussion. What kind of cost model would you use in Fabolas?
- Discussion. Could one use an acquisition function other than entropy search for Fabolas?