Speedup Techniques for Hyperparameter Optimization

Overview of Multi-Fidelity Optimization

Bernd Bischl   Frank Hutter   Lars Kotthoff
Marius Lindauer   Joaquin Vanschoren

Motivating Example

- One possible cheap approximation of an expensive function: use a data subset
  - Many cheap evaluations on small subsets
  - Few expensive evaluations on the full data

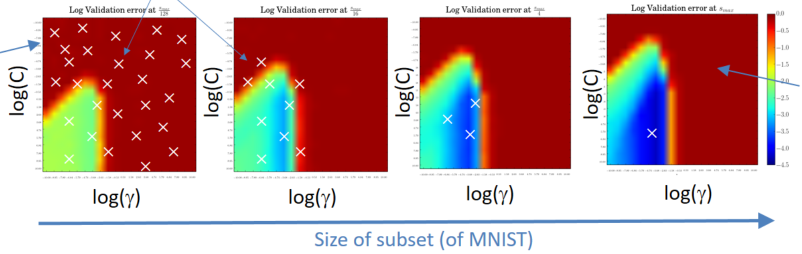
Motivating Example

- One possible cheap approximation of an expensive function: use a data subset
  - Many cheap evaluations on small subsets
  - Few expensive evaluations on the full data

- E.g.: Support Vector Machines (SVM) on MNIST dataset (hyperparameters: C, γ)

× denotes evaluated hyperparameter settings

Log validation error (red:bad; green:OK; blue:good) when trained on subset of 390 data points (takes few seconds to compute)

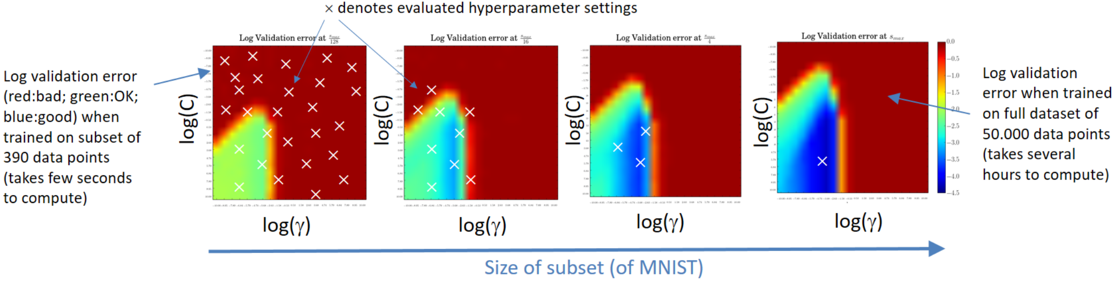


Log validation error when trained on full dataset of 50.000 data points (takes several hours to compute)

Motivating Example

- One possible cheap approximation of an expensive function: use a data subset
  - Many cheap evaluations on small subsets
  - Few expensive evaluations on the full data

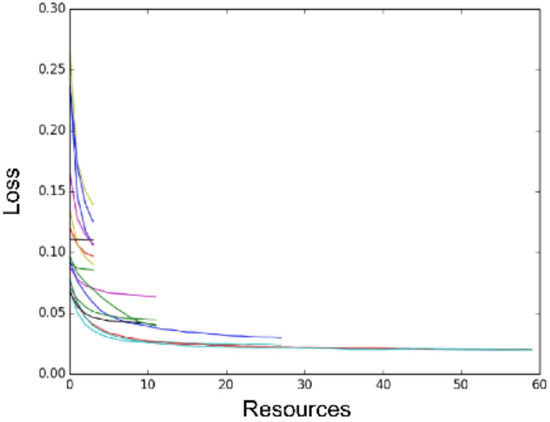
- E.g.: Support Vector Machines (SVM) on MNIST dataset (hyperparameters: C, γ)



$\rightarrow$ up to 1000x speedups over blackbox optimization on full data [Klein et al, AISTATS 2017]

Motivating Example 2: Shorter Runs of Anytime Algorithms

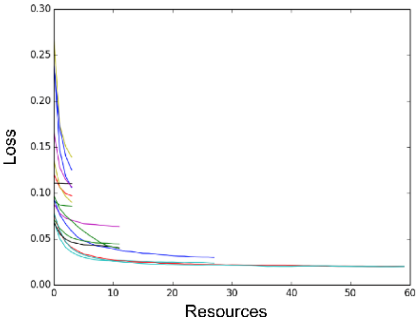
- Performance with shorter runs of an anytime algorithm (such as SGD):



Multi-Fidelity Optimization In General

Exploit cheap approximations of an expensive blackbox function $\rightarrow$ afford more configurations

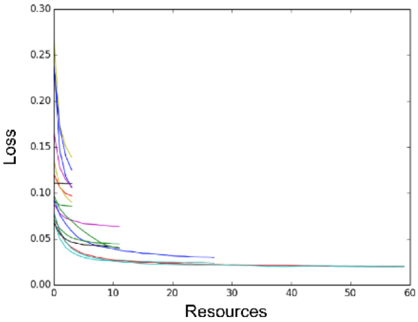
- Idea: eliminate poor configurations early, allocate more resources to promising ones.



Multi-Fidelity Optimization In General

Exploit cheap approximations of an expensive blackbox function → afford more configurations

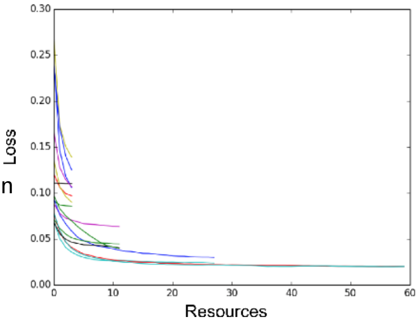
- Idea: eliminate poor configurations early, allocate more resources to promising ones.
- Possible Resources:
  - Data subset size
  - Runtime / # epochs / # iterations



Multi-Fidelity Optimization In General

Exploit cheap approximations of an expensive blackbox function → afford more configurations

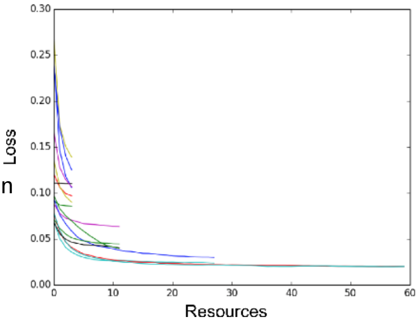
- Idea: eliminate poor configurations early, allocate more resources to promising ones.
- Possible Resources:
  - Data subset size
  - Runtime / # epochs / # iterations
  - Downsampled size of images in object recognition
  - Depth / width of neural networks



Multi-Fidelity Optimization In General

Exploit cheap approximations of an expensive blackbox function $\rightarrow$ afford more configurations

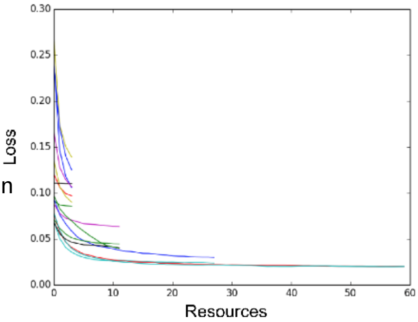
- Idea: eliminate poor configurations early, allocate more resources to promising ones.
- Possible Resources:
  - Data subset size
  - Runtime / # epochs / # iterations
  - Downsampled size of images in object recognition
  - Depth / width of neural networks
  - Number of trees
  - Number of features
  - Number of cross validation folds



Multi-Fidelity Optimization In General

Exploit cheap approximations of an expensive blackbox function → afford more configurations

- Idea: eliminate poor configurations early, allocate more resources to promising ones.
- Possible Resources:
  - Data subset size
  - Runtime / # epochs / # iterations
  - Downsampled size of images in object recognition
  - Depth / width of neural networks
  - Number of trees
  - Number of features
  - Number of cross validation folds



  - General concept, applicable even in fields outside ML, e.g., fluid simulation:
    - ★ Number of particles
    - ★ Time scale of simulation

General Remarks on Multi-Fidelity Optimization

- Often, we have a choice which resources we use as budget

General Remarks on Multi-Fidelity Optimization

- Often, we have a choice which resources we use as budget
- For multi-fidelity optimization to be helpful, performance with low budgets should be informative about performance with high budgets

General Remarks on Multi-Fidelity Optimization

- Often, we have a choice which resources we use as budget
- For multi-fidelity optimization to be helpful, performance with low budgets should be informative about performance with high budgets
- In the simplest case: good with low resources $\leftrightarrow$ good with high resources.
  - In practice, this is of course not always true

How Useful is the Cheap Approximation? The Rank Correlation

Given:

- A set of configurations $\mathbf{\Lambda} = \{\boldsymbol{\lambda}_1, ..., \boldsymbol{\lambda}_n\}$
- The performances $f(\boldsymbol{\lambda}_1), ..., f(\boldsymbol{\lambda}_n)$ on the expensive black box
- The performances $g(\boldsymbol{\lambda}_1), ..., g(\boldsymbol{\lambda}_n)$ on a cheap approximation of the black box

How Useful is the Cheap Approximation? The Rank Correlation

Given:

- A set of configurations $\mathbf{\Lambda} = \{\boldsymbol{\lambda}_1, ..., \boldsymbol{\lambda}_n\}$
- The performances $f(\boldsymbol{\lambda}_1), ..., f(\boldsymbol{\lambda}_n)$ on the expensive black box
- The performances $g(\boldsymbol{\lambda}_1), ..., g(\boldsymbol{\lambda}_n)$ on a cheap approximation of the black box

We compute the Spearman rank correlation between $[f(\boldsymbol{\lambda}_1), ..., f(\boldsymbol{\lambda}_n)]$ and $[g(\boldsymbol{\lambda}_1), ..., g(\boldsymbol{\lambda}_n)]$

- If this is high (in the extreme: 1), the relative ranking of the configurations is the same on f and g
  - In that case, we can optimize cheaply on g and also obtain an optimum for f
- If it is low (≈ 0), optimizing g does not tell us anything about f

How Useful is the Cheap Approximation? The Rank Correlation

Given:

- A set of configurations $\boldsymbol{\Lambda} = \{\boldsymbol{\lambda}_1, ..., \boldsymbol{\lambda}_n\}$
- The performances $f(\boldsymbol{\lambda}_1), ..., f(\boldsymbol{\lambda}_n)$ on the expensive black box
- The performances $g(\boldsymbol{\lambda}_1), ..., g(\boldsymbol{\lambda}_n)$ on a cheap approximation of the black box

We compute the Spearman rank correlation between $[f(\boldsymbol{\lambda}_1), ..., f(\boldsymbol{\lambda}_n)]$ and $[g(\boldsymbol{\lambda}_1), ..., g(\boldsymbol{\lambda}_n)]$

- If this is high (in the extreme: 1), the relative ranking of the configurations is the same on f and g
  - In that case, we can optimize cheaply on g and also obtain an optimum for f
- If it is low (≈ 0), optimizing g does not tell us anything about f

Goal: find approximations g that are very cheap but have high rank correlations with f

Questions to Answer for Yourself / Discuss with Friends

- Repetition. Which cheap approximation is better in this hypothetical case?
  - Downscaling images (5x cheaper, rank correlation of 0.8)
  - Less epoch of SGD (4x cheaper, rank correlation of 0.75)

- Discussion. Can you think of an application of your interest
  where you would likely have a good multi-fidelity approximation?