Optimization of Machine Learning Hyperparameters

Dr. Frank Hutter
Head of Emmy Noether Research Group on Learning, Optimization, and Automated Algorithm Design
Computer Science Institute
University of Freiburg, Germany

July 2014
Recap: Gaussian processes

- Prior over functions $P(f)$
- Data $\mathcal{D}_{1:t} = \{x_{1:t}, f(x_{1:t})\}$

\[
P(f|\mathcal{D}_{1:t}) \propto P(\mathcal{D}_{1:t}|f) P(f)
\]
Today’s Learning Goals

• After today’s lecture, you can ...
  – Discuss some limitations of GP-based Bayesian optimization
  – Explain regression trees and random forests
  – Reason about the pros and cons of using a particular hyperparameter optimization method for a particular problem
  – Effectively use modern hyperparameter optimization methods
Outline of Today’s Class

Extensions of Bayesian optimization
  – General types of hyperparameters
  – Random forests
  – Bayesian optimization with random forests & applications
  – Which hyperparameters matter?

• Overview of advanced methods not covered

• Demo of systems you can use for the project
Types of Hyperparameters

• Numerical hyperparameters
  – Continuous
    (e.g., learning rate, momentum, regularization)
  – Integer
    (e.g., #units, #layers, batch size in stochastic gradient descent)

• Categorical hyperparameters
  – Finite domain, unordered
    (e.g., activation function \( \in \{ \text{tanh}, \text{ReLu}, \text{or sigmoid} \} \))
  – Special case: Boolean
    (e.g., use preprocessing or not)
Conditional Hyperparameters

• **Conditional** hyperparameters are only active if certain other hyperparameters take certain values
  – The hyperparameters they depend on are called their parents

• Examples in neural networks
  – All hyperparameters in layer K are only active if the network has at least K layers
  – E.g., “Learning rate in layer 3” is a conditional parameter with parent “network depth”

• Examples in model selection
  – Select between algorithms A and B and their hyperparameters
  – Then A’s hyperparameters are only active if we select A
Details on Conditional Hyperparameters

• Conditional hyperparameters can be parents themselves
  – The hyperparameter space is often tree-structured
  – Sometimes, hyperparameters depend on multiple parents; the space is then structured as a directed acyclic graph (DAG)

• Semantics:
  – If a hyperparameter is not active it does not matter which value we select for it
  – The learning algorithm will not even inspect the hyperparameter
Strengths of GP-based Bayesian optimization

• Few numerical hyperparameters
  – E.g., up to 10 real-valued hyperparameters
  – This covers a broad range of popular algorithms

• Smoothly varying performance
  – No sudden jumps in performance

• Reliable performance
  – E.g., similar performance when using a different random seed
  – Observation noise is the same everywhere: homoscedasticity

• Evaluating performance is expensive
  – The overhead of using GP models is negligible
Weaknesses of GP-based Bayesian optimization

• Complex parameter spaces
  – Many parameters (but low effective dimensionality)
  – Categorical parameters: finite domain, unordered, e.g., activation function $\in \{\text{tanh, ReLu, or sigmoid}\}$
  – Structure: e.g., “Learning rate in layer 3” is a conditional parameter with parent “network depth”

• Noise: sometimes heteroscedastic, large, non-Gaussian

• Robustness (importance of priors)

• Model overhead (budget is runtime, not #function evals)

• A better model in these cases: random forests [Breiman, '01]
  – Adapted to yield uncertainty estimates
Outline of Today’s Class

• Extensions of Bayesian optimization
  – General types of hyperparameters
  Random forests
    – Bayesian optimization with random forests & applications
    – Which hyperparameters matter?

• Overview of advanced methods not covered

• Demo of systems you can use for the project
Random Forests (RFs)

- RFs for classification = bagging + decision trees

- You learned about decision trees before
- You also heard about bagging before
  - Take T bootstrap samples of the data (sample with repetitions)
  - For each bootstrap sample, fit an independent model
  - Average the T predictions
  - Effective method to reduce over-fitting

- What’s another method you learned for avoid over-fitting of decision trees?
  - Pruned decision trees; RFs typically perform better
Random Forests (RFs)

- Bagging is **better with less correlated models**
  - Trees are a great model for bagging: individual trees overfit to different characteristics of the bootstrap samples, and are thus only weakly correlated

- Idea: make trees even less correlated using randomness beyond the bootstrap sampling
  - E.g., slightly perturb data in each tree
  - E.g., in each split only allow splits on a random subset of variables (only evaluate splitting criterion on those)
  - E.g., in each split only allow a random subset of split points for each split variable
Random Forests (RFs)

• Invented by Leo Breiman in 2001

• Properties
  – More trees ≠ more overfitting (in contrast, boosting will push training error to zero)
  – RFs are one of the most robust off-the-shelf supervised ML methods
  – Many ML competitions have been won with RFs
Random Forests (RFs)

- Came from statistics literature, initially not widely used in ML (not kernel-based, not neural networks)
- Now one of the most frequently cited papers in ML
Regression Trees

• Recall decision trees for classification
  – Prediction: majority class label in selected leaf
  – Splitting criterion: information gain

• Regression trees: the analogue for regression
  – Prediction: mean numerical value in selected leaf
  – Splitting criterion: use the partition \((L,R)\) of data points \((x_1, y_1), \ldots, (x_t, y_t)\) into left and right children \(L\) and \(R\) that minimizes:

\[
l(L, R) = \sum_{x_i \in L} (y_i - \mu_L)^2 + \sum_{x_i \in R} (y_i - \mu_R)^2
\]

where \(\mu_L = \text{mean}(\sum_{x_i \in L} (y_i))\) and \(\mu_R = \text{mean}(\sum_{x_i \in R} (y_i))\).
Regression Tree Training

<table>
<thead>
<tr>
<th>param 1</th>
<th>feature 2</th>
<th>param 3</th>
<th>runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>2</td>
<td>red</td>
<td>3.7</td>
</tr>
<tr>
<td>false</td>
<td>2.5</td>
<td>blue</td>
<td>20</td>
</tr>
<tr>
<td>true</td>
<td>5.5</td>
<td>red</td>
<td>2.1</td>
</tr>
<tr>
<td>false</td>
<td>5.5</td>
<td>blue</td>
<td>25</td>
</tr>
<tr>
<td>false</td>
<td>5</td>
<td>red</td>
<td>1.2</td>
</tr>
<tr>
<td>true</td>
<td>4.5</td>
<td>green</td>
<td>19</td>
</tr>
<tr>
<td>true</td>
<td>4</td>
<td>blue</td>
<td>12</td>
</tr>
<tr>
<td>true</td>
<td>3.5</td>
<td>green</td>
<td>17</td>
</tr>
</tbody>
</table>

param\(_3\) ∈ \{red\}

<table>
<thead>
<tr>
<th>param 1</th>
<th>feature 2</th>
<th>param 3</th>
<th>runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>2</td>
<td>red</td>
<td>3.7</td>
</tr>
<tr>
<td>true</td>
<td>5.5</td>
<td>red</td>
<td>2.1</td>
</tr>
<tr>
<td>false</td>
<td>5</td>
<td>red</td>
<td>1.2</td>
</tr>
</tbody>
</table>

param\(_3\) ∈ \{blue, green\}

feature\(_2\) ≤ 3.5

<table>
<thead>
<tr>
<th>param 1</th>
<th>feature 2</th>
<th>param 3</th>
<th>runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>2</td>
<td>red</td>
<td>3.7</td>
</tr>
<tr>
<td>false</td>
<td>5.5</td>
<td>red</td>
<td>2.1</td>
</tr>
<tr>
<td>false</td>
<td>5</td>
<td>red</td>
<td>1.2</td>
</tr>
</tbody>
</table>

feature\(_2\) > 3.5

<table>
<thead>
<tr>
<th>param 1</th>
<th>feature 2</th>
<th>param 3</th>
<th>runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>5.5</td>
<td>red</td>
<td>2.1</td>
</tr>
<tr>
<td>true</td>
<td>5</td>
<td>red</td>
<td>1.2</td>
</tr>
</tbody>
</table>
Regression Tree Training

- In each internal node: only store split criterion used
- In each leaf: store mean of runtimes
Regression Tree Predictions

E.g. $x_{n+1} = (\text{true}, 4.7, \text{red})$

- Walk down tree, return mean runtime stored in leaf $\Rightarrow 1.65$

```
param_3 \in \{\text{red}\}

param_3 \in \{\text{blue, green}\}

\text{feature}_2 \leq 3.5

3.7

\text{feature}_2 > 3.5

1.65

\ldots\ldots\ldots\ldots
```
Random Forests: Sets of Regression Trees

• **Training**
  – Draw B bootstrap samples of the data
  – For each bootstrap sample, fit a randomized regression tree

• **Prediction**
  – Predict with each of the B trees
  – Return empirical mean and variance across these B predictions

• **Complexity** for t data points and B trees
  – Training: $O(B t \log^2 t)$
  – Prediction: $O(B \log t)$

• One of the best off-the-shelf learning methods available
Outline of Today’s Class

• Extensions of Bayesian optimization
  – General types of hyperparameters
  – Random forests
  Bayesian optimization with random forests & applications
  – Which hyperparameters matter?

• Overview of advanced methods not covered

• Demo of systems you can use for the project
SMAC in a Nutshell

SMAC: Sequential Model-Based Algorithm Configuration

repeat
  construct RF model to predict performance
  use that model to select promising configurations
  compare each selected configuration against the best known
until time budget exhausted

• Distributed SMAC
  – Maintain queue of promising configurations
  – Compare these to $\theta^*$ on distributed worker cores

[Hutter, Hoos, Leyton-Brown, 2009-2014]
**WEKA** [Witten et al, 1999-current]

- most widely used off-the-shelf machine learning package
- over 20,000 citations on Google scholar

Java implementation of a **broad range of methods**

- 27 base classifiers (with up to 10 parameters each)
- 10 meta-methods
- 2 ensemble methods

Different methods work best on different data sets

- Want a **true off-the-shelf solution**: Learn
WEKA’s configuration space

Base classifiers
- 27 choices, each with subparameters

Hierarchical structure on top of base classifiers
- In total: 768 parameters, $10^{47}$ configurations
- Optimize cross-validation performance over this space using SMAC
Auto-WEKA: Results

Auto-WEKA performs better than best base classifier
- Even when “best classifier” uses an oracle
- Especially on the 8 largest datasets
- In 6/21 cases more than 10% reductions in relative error
- Time requirements: 30h on 4 cores

Comparison to full grid search
- Union of grids over parameters of all 27 base classifiers
- **Auto-WEKA is 100 times faster**
- **Auto-WEKA has better generalization performance in 15/21 cases**

Auto-WEKA based on SMAC vs. TPE [Bergstra et al, 2011]
- SMAC yielded **better CV performance in 19/21 cases**
- SMAC yielded **better generalization performance in 14/21 cases**
- Differences usually small, in 3 cases substantial (SMAC better)
Auto-WEKA Discussion

• Off-the-shelf machine learning tools are now available
  – Expert understanding of ML techniques not required to use them
  – Users still need to provide good features

• Auto-WEKA is available online: automl.org/autoweka

• Ongoing work
  – Wrappers from several programming languages
  – Reason across datasets to jump-start Auto-WEKA
  – Do the same for sk-learn: Auto-sklearn
  – Do the same for deep neural networks: Auto-Deep
• **Standard hyperparameters**
  – Learning rate, momentum, ...

• **Architectural hyperparameters**
  – How many layers?
  – Use convolutions? With which stride?

• **Hyperparameters in each hidden layer k**
  – #hidden units, regularization, weight initialization, ...
  – Conditional on there being at least k layers
Comparing Hyperparameter Optimizers

[Eggensperger, Feurer, Hutter, Bergstra, Snoek, Hoos & Leyton-Brown, 2013]

- Hyperparameter optimization library: automl.org/hpolib
  - Benchmarks
    - Artificial test functions (for quick debugging)
    - Low-dimensional: logistic regression, online LDA, structured SVM
    - Medium-dimensional: neural network, deep network
    - High-dimensional: Auto-WEKA
  - Optimizers
    - SMAC [Hutter et al, '11], based on random forests
    - Spearmint [Snoek et al, '12], based on Gaussian processes
    - TPE [Bergstra et al, '11], based on density estimators (EDA algorithm)
  - Results
    - Spearmint performs best for low-dimensional continuous problems
    - SMAC performs best for high-dimensional structured optimization, e.g. combined architecture search and hyperparameter optimization
Outline of Today’s Class

• Extensions of Bayesian optimization
  – Challenges for Bayesian optimization
  – Random forests
  – Bayesian optimization with random forests & applications
    Which hyperparameters matter?

• Overview of advanced topics we did not cover

• Demo of systems you can use for your project
Which hyperparameters matter?

- Recall: grid search provides intuition
  - Which hyperparameters matter?
  - Which hyperparameters interact?
  - But it’s exponential in the number of hyperparameters

- Recall: better hyperparameter optimization methods exist

- We can learn a model on their performance data
  - That’s exactly what we do in Bayesian optimization
  - With that model we can predict performance at the grid

  - We can also predict **marginals**: how good is a hyperparameter value on average over all possible values of other hyperparameters?
We can predict arbitrary marginals

- Example algorithm with 3 hyperparameters: $\kappa, \tau_0, \text{and } S$
- Model based on 100 random samples
- Full grid has 288 points

Blue circles: ground truth marginals using full grid
We can quantify importance & interactions

- 65% of variance is due to $S$
- Another 18% is due to interaction between $S$ and $\kappa$
- This analysis takes milliseconds given the data: automl.org/fanova
Outline of Today’s Class

• Extensions of Bayesian optimization
  – Challenges for Bayesian optimization
  – Random forests
  – Bayesian optimization with random forests & applications
  – Which hyperparameters matter?

Overview of advanced topics we did not cover

• Demo of systems you can use for your project
The Algorithm Configuration Problem

Definition

– Given:
  • Runnable algorithm $A$ with configuration space $\Theta = \Theta_1 \times \cdots \times \Theta_n$
  • Distribution $D$ over problem instances $\Pi$
  • Performance metric $m : \Theta \times \Pi \rightarrow \mathbb{R}$

– Find:

$$\theta^* \in \arg \min_{\theta \in \Theta} \mathbb{E}_{\pi \sim D}[m(\theta, \pi)]$$

Motivation

Customize versatile algorithms for different application domains

– Fully automated improvements
– Optimize speed, accuracy, memory, energy consumption, ...
Algorithm Performance Prediction

• Given:
  – Configuration space \( \Theta = \Theta_1 \times \cdots \times \Theta_n \)
  – For each problem instance \( i: x_i \), a vector of feature values
  – Observed algorithm runtime data: \( (\theta_1, x_1, y_1), \ldots, (\theta_n, x_n, y_n) \)

• Find: model \( m: [\theta, x] \mapsto y \) predicting A’s performance
  – Regression problem
  – Statistical model \( \rightarrow \) we’re not solving the halting problem

• Applications:
  – Algorithm configuration: pick best fixed \( \theta \)
  – Algorithm selection: given \( x \), pick best algorithm from \( \{A_1, \ldots, A_k\} \)
  – Context-specific algorithm configuration: given \( x \), select \( \theta \)
Automated Machine Learning

• Machine Learning has celebrated **substantial successes**
• But it requires **human experts** to
  – Preprocess the data
  – Perform feature selection
  – Select a model family
  – Optimize hyperparameters
  – ...

• **AutoML**: taking the human expert out of the loop

  “Civilization advances by extending the number of important operations which we can perform without thinking of them”
  (Alfred North Whitehead)
Automated Machine Learning

• First workshop on AutoML at the International Conference on Machine Learning (ICML)
  – Beijing, China, two weeks ago
  – Some topics:
    • Hyperparameter optimization
    • Combined search over model types & their hyperparameters
    • Combined architecture and hyperparameter optimization
    • Prediction of learning curves of deep neural networks
    • Automated learning of ensembles
    • Learning across data sets

• **AutoML challenge**, organized by Isabelle Guyon
  – In 5 phases, September – April
  – Opportunity for interesting MSc projects/theses
Outline of Today’s Class

• Extensions of Bayesian optimization
  – Challenges for Bayesian optimization
  – Random forests
  – Bayesian optimization with random forests & applications
  – Which hyperparameters matter?

• Overview of advanced methods not covered

Demo of systems you can use for the project