Overview on Automatic Tuning of Hyperparameters

Alexander Fonarev

http://newo.su

20.02.2016
Outline

• Introduction to the problem and examples
• Introduction to Bayesian optimization
• Overview of surrogate models
• Additional modifications
• Existing implementations
Hyperparameter examples

• Tree depth — decision trees
• Regularization coefficient — linear models
• Gradient descend step size — neural networks
• Normalization coefficient — data preprocessing
Examples of problems

- Training of the ranking in Yandex — days
- Parameters in NNs — tens
Popular and easy approaches

• Experience of experts
• Grid search
• **Random search**
• Manual coordinate-descend

https://www.oreilly.com/ideas/evaluating-machine-learning-models/page/5/hyperparameter-tuning
Gradient-free and global optimization

- Genetic algorithms
- Simulated annealing
- Response surfaces
- etc
Stochastic functions

How to optimize such functions?
How to choose the next point to evaluate?

Figure 4: Gaussian process from Figure 2, additionally showing the region of probable improvement. The maximum observation is at $x^+$. The darkly shaded area in the superimposed Gaussian above the dashed line can be used as a measure of improvement, $I(x)$. The model predicts almost no possibility of improvement by observing at $x_1$ or $x_2$, while sampling at $x_3$ is more likely to improve on $f(x^+)$. Nothing in this response-surface area is so simple. There always seems to be a counterexample. In this case, the difficulty is that the PI($\cdot$) method is extremely sensitive to the choice of the target. If the desired improvement is too small, the search will be highly local and will only move on to search globally after searching nearly exhaustively around the current best point. On the other hand, if $\epsilon$ is set too high, the search will be excessively global, and the algorithm will be slow to fine-tune any promising solutions.

A somewhat more satisfying alternative acquisition function would be one that takes into account not only the probability of improvement, but the magnitude of the improvement a point can potentially yield. In particular, we want to minimize the expected deviation from the true maximum $f(x^?)$, when choosing $x$. Some points to consider are $x_1$, $x_2$, and $x_3$.
Acquisition function

Different acquisition strategies

Built surrogate

Probability of improvement (PI)
$$PI(x) = P(f(x) \geq f(x^+) + \xi)$$

Expected improvement (EI)
$$EI(x) = \mathbb{E}(\max\{0, f(x) - f(x^+) - \xi\})$$

Upper Confidence Bound (UCB)
$$UCB(x) = \mathbb{E}f(x) + \nu\sigma(x)$$
Comparison of acquisition functions

Common algorithm

1. Initial design — evaluate the black box in some points

2. Adaptive design
   a) Build a surrogate
   b) Find the argmax of Expected Improvement
   c) Evaluate the black box in this point
   d) Go to step 2
How to perform the initial design?

- Best from previous experiments
- Ask for experts
- Random
- Grid
- Several optimal design criteria

https://en.wikipedia.org/wiki/Optimal_design
Types of surrogates

- Gaussian Processes (GP)
- Tree of Parzen Estimators (TPE)
- Sequential Model-based Algorithm Configuration (SMAC)
Gaussian Process (GP)

http://www.robots.ox.ac.uk/~mebden/reports/GPtutorial.pdf
https://www.youtube.com/watch?v=4vGiHC35j9s
https://github.com/JasperSnoek/spearmint (Spearmint)
Tree of Parzen Estimators (TPE)

Estimates $p(y)$ and $p(x|y)$ instead of $p(y|x)$. The core idea — nonparametric density approximations of $x$.

http://papers.nips.cc/paper/4443-algorithms-for-hyper-parameter-optimization.pdf
https://github.com/hyperopt/hyperopt (Hyperopt)
Reminder: Random Forest

Not robust (has high variance)
Reminder: Random Forest

Average of slightly different trees:

https://www.youtube.com/watch?v=3kYujfDgmNk
Sequential Model-based Algorithm Configuration (SMAC)

• Mean in each point — prediction of RF
• Variance in each point — variance of predictions of separate trees from RF

https://github.com/automl/pysmac (SMAC)
How to find the maximum of EI?

Use global optimization. Some options:

- Random search
- Genetic algorithms
- Simulated annealing
- Response surfaces
Parallelization

• Batch
• Asynchronous adaptive design

What else?

• Data structures for fast optimum search
• Cold start problem
• Different parameters require different learning time
• Labels transformation
• Different parameter types
## Open source implementations

<table>
<thead>
<tr>
<th>Package</th>
<th>License</th>
<th>URL</th>
<th>Language</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMAC</td>
<td>Academic non-commercial license.</td>
<td><a href="http://www.cs.ubc.ca/labs/beta/Projects/SMAC">http://www.cs.ubc.ca/labs/beta/Projects/SMAC</a></td>
<td>Java</td>
<td>Random forest</td>
</tr>
<tr>
<td>Hyperopt</td>
<td>BSD</td>
<td><a href="https://github.com/hyperopt/hyperopt">https://github.com/hyperopt/hyperopt</a></td>
<td>Python</td>
<td>Tree Parzen estimator</td>
</tr>
<tr>
<td>Spearmint</td>
<td>Academic non-commercial license.</td>
<td><a href="https://github.com/HIPS/Spearmint">https://github.com/HIPS/Spearmint</a></td>
<td>Python</td>
<td>Gaussian process</td>
</tr>
<tr>
<td>Bayesopt</td>
<td>GPL</td>
<td><a href="http://rmcantin.bitbucket.org/html">http://rmcantin.bitbucket.org/html</a></td>
<td>C++</td>
<td>Gaussian process</td>
</tr>
<tr>
<td>PyBO</td>
<td>BSD</td>
<td><a href="https://github.com/mwhoffman/pybo">https://github.com/mwhoffman/pybo</a></td>
<td>Python</td>
<td>Gaussian process</td>
</tr>
<tr>
<td>MOE</td>
<td>Apache 2.0</td>
<td><a href="https://github.com/Yelp/MOE">https://github.com/Yelp/MOE</a></td>
<td>Python / C++</td>
<td>Gaussian process</td>
</tr>
</tbody>
</table>
How to evaluate method?

• Use real data
• Use synthetic data

http://infinity77.net/global_optimization/test_functions.html#test-functions-index
https://github.com/andyfaff/ampgo
Comparison of different approaches

<table>
<thead>
<tr>
<th>Experiment</th>
<th>#evals</th>
<th>SMAC Valid. loss</th>
<th>SMAC Best loss</th>
<th>Spearmint Valid. loss</th>
<th>Spearmint Best loss</th>
<th>TPE Valid. loss</th>
<th>TPE Best loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>branin (0.398)</td>
<td>200</td>
<td>0.655±0.27</td>
<td>0.408</td>
<td>0.398±0.00</td>
<td>0.398</td>
<td>0.526±0.13</td>
<td>0.422</td>
</tr>
<tr>
<td>har6 (-3.322)</td>
<td>200</td>
<td>-2.977±0.11</td>
<td>-3.154</td>
<td>-3.133±0.41</td>
<td>-3.322</td>
<td>-2.823±0.18</td>
<td>-3.039</td>
</tr>
<tr>
<td>Log. Regression</td>
<td>100</td>
<td>8.6±0.9</td>
<td>7.7</td>
<td>7.3±0.2</td>
<td>7.0</td>
<td>8.2±0.6</td>
<td>7.5</td>
</tr>
<tr>
<td>LDA ongrid</td>
<td>50</td>
<td>1269.6±2.9</td>
<td>1266.2</td>
<td>1272.6±10.3</td>
<td>1266.2</td>
<td>1271.5±3.5</td>
<td>1266.2</td>
</tr>
<tr>
<td>SVM ongrid</td>
<td>100</td>
<td>24.1±0.1</td>
<td>24.1</td>
<td>24.6±0.9</td>
<td>24.1</td>
<td>24.2±0.0</td>
<td>24.1</td>
</tr>
<tr>
<td>HP-NNET convex</td>
<td>100</td>
<td>19.5±1.5</td>
<td>17.0</td>
<td>20.6±0.3</td>
<td>20.1</td>
<td>19.5±1.6</td>
<td>17.4</td>
</tr>
<tr>
<td>HP-NNET convex</td>
<td>200</td>
<td>18.3±1.9</td>
<td>15.2</td>
<td>20.0±0.9</td>
<td>17.3</td>
<td>18.5±1.4</td>
<td>16.2</td>
</tr>
<tr>
<td>HP-NNET MRBI</td>
<td>100</td>
<td>51.5±2.8</td>
<td>46.1</td>
<td>52.2±3.3</td>
<td>46.5</td>
<td>50.0±1.7</td>
<td>47.3</td>
</tr>
<tr>
<td>HP-NNET MRBI</td>
<td>200</td>
<td>48.3±1.80</td>
<td>46.1</td>
<td>51.4±3.2</td>
<td>46.5</td>
<td>48.9±1.4</td>
<td>46.9</td>
</tr>
<tr>
<td>HP-DBNET convex</td>
<td>100</td>
<td>16.4±1.2</td>
<td>14.5</td>
<td>20.74±6.9</td>
<td>15.5</td>
<td>17.29±1.7</td>
<td>15.3</td>
</tr>
<tr>
<td>HP-DBNET convex</td>
<td>200</td>
<td>15.4±0.8</td>
<td>14.0</td>
<td>17.45±5.6</td>
<td>14.6</td>
<td>16.1±0.5</td>
<td>15.3</td>
</tr>
<tr>
<td>Auto-WEKA</td>
<td>30h</td>
<td>27.5±4.9</td>
<td>22.3</td>
<td>40.64±7.2</td>
<td>31.9</td>
<td>35.5±2.9</td>
<td>28.8</td>
</tr>
<tr>
<td>Log. Regression 5CV</td>
<td>500 folds</td>
<td>8.1±0.2</td>
<td>7.8</td>
<td>8.2±0.1</td>
<td>7.9</td>
<td>8.9±0.5</td>
<td>8.1</td>
</tr>
<tr>
<td>HP-NNET convex 5CV</td>
<td>500 folds</td>
<td>18.2±1.5</td>
<td>16.9</td>
<td>23.0±5.0</td>
<td>19.7</td>
<td>20.9±1.3</td>
<td>18.6</td>
</tr>
<tr>
<td>HP-NNET MRBI 5CV</td>
<td>500 folds</td>
<td>47.9±0.7</td>
<td>47.2</td>
<td>52.8±5.1</td>
<td>46.6</td>
<td>50.8±1.4</td>
<td>48.2</td>
</tr>
</tbody>
</table>

Terminology

• Bayesian optimization
• Reinforcement learning
• Surrogate model
• Kriging
What else?

• Data structures for fast optimum search
• Cold start problem
• Different parameters require different learning time
• Labels transformation
Summary

- We usually need to optimize stochastic functions
- Surrogate model should be fit to the data
- Several good implementations already exist
  - Random search is much better than you thought!
Thank you!

Alexander Fonarev

http://newo.su