HDS-LEE Course on Hyperparameter Optimization

Dr. Charlotte Debus, Karlsruhe Institute of Technology
Dr. Alexander Rüttgers, German Aerospace Center
What are hyperparameters?

“In machine learning, a hyperparameter is a parameter whose value is set before the learning process begins. By contrast, the values of other parameters are derived via training.” [Source: Wikipedia]

- Model Hyperparameters
  - Configures the model (model selection)

- Algorithm Hyperparameters
  - Configures learning process (speed and quality)
What are hyperparameters?

Examples for Hyperparameters in ML Models (other than NN)

<table>
<thead>
<tr>
<th>Machine Learning Process</th>
<th>Hyperparameter</th>
<th>Model vs Algorithm HP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logistic Regression</td>
<td>Regularization</td>
<td>Algorithm HP</td>
</tr>
<tr>
<td>K-means clustering</td>
<td>K (number of clusters)</td>
<td>Model HP</td>
</tr>
<tr>
<td>SVM</td>
<td>ζ (margin distance/Slack variable)</td>
<td>Model HP</td>
</tr>
<tr>
<td>Decision Tree</td>
<td>Impurity Metric (i.e. Gini impurity)</td>
<td>Algorithm HP</td>
</tr>
<tr>
<td>Random Forest</td>
<td>Number of trees</td>
<td>Model HP</td>
</tr>
<tr>
<td></td>
<td>bootstraping</td>
<td>Algorithm HP</td>
</tr>
<tr>
<td>Neural Networks</td>
<td>Number of hidden layers</td>
<td>Model HP</td>
</tr>
<tr>
<td></td>
<td>Mini-batch size</td>
<td>Algorithm HP</td>
</tr>
<tr>
<td>Stochastic Gradient Descent</td>
<td>Learning Rate</td>
<td>Algorithm HP</td>
</tr>
</tbody>
</table>
The „Validation Dataset“

- In machine learning tasks: 3 Datasets
  - Training → to train the model
  - Testing → unseen dataset, to evaluate trained model performance
  - …
  - Validation → Evaluation of trained model for a HP configuration
    Model is evaluated on validation set (substitute to get the expected performance on the true distribution of the data)
Hyperparameter optimization

Find the model hyperparameters with the best score on the validation set

- Hyperparameters $\hat{x}$
- Objective / loss function $f(\hat{x}) \rightarrow$ Validation score
- We need…
  - A validation set
  - A search space for the hyperparameters $\chi$
  - An optimization algorithm (aka tuning method)
Exhaustive search

Grid Search
Random Search
Exhaustive search

Grid Search

1. Discretize search space of hyperparameters (Cartesian grid)
2. Train model for every hyperparameter configuration
3. Evaluate loss for every configuration
4. Select the best configuration

- Trivial to parallelize (train several models at the same time)
- Curse of dimensionality

Random Search

1. Define Search space
2. Randomly draw samples for hyperparameters
3. Evaluate loss
   - Run number of trials (time budget)
4. Select best option

- If two hyper-parameters are little correlated, optima of each parameter more precisely found

[Link to blog post: https://nanonets.com/blog/hyperparameter-optimization/]
Exhaustive search

- Grid search and random search: blindly try a new configuration in the search space or make an educated guess of where the most interesting configuration might be.

Problem: For every different hyperparameter configuration, we have to train the model (training set), make predictions (validation set), and then calculate the validation metric.

➢ Expensive!!! → Model training
Bayesian Inference

Choose the next sample $x$ in an *informed manner* based on previous observations

- Probabilistic model that maps hyperparameters to a probability of objective function score
Surrogate-based optimization using Bayesian inference

„Let machine learning figure out the hyperparameters!“ (Snoek, Larochelle, Adams, NIPS 2012)
Black-Box optimization

\[ \hat{x}^* = \arg \min_{\hat{x} \in \chi} f(\hat{x}) \]

- We don’t have access to the true form of the objective function \( f(x) \)
  - No gradients
- We can only observe function values
  - Query \( f(x) \) with an input value \( x \) and get some output \( y \)

Sequential strategy which maps collected data to the next query point

Sequential Model-Based Optimization (SMBO)
Sequential Model-Based Optimization (SMBO)

1. Build a model of what the objective function might look like
   - **Surrogate probability (Prior):** Initial view of the world
2. Find the hyperparameters that perform best on the surrogate
   - **Acquisition/Selection Function:** What are the points that we should evaluate next?
3. Apply these hyperparameters to the true objective function
4. Update surrogate model incorporating the new results
   - **Posterior:** Updated view of the world
5. Repeat
   - Keep track of past evaluation results
Surrogate model of the objective function: Prior

- Model:
  - Make predictions
  - Maintain a measure of uncertainty over those predictions
  - Incorporate prior knowledge
  - Find a distribution over the possible functions $f(x)$ that are consistent with the observed data

Gaussion Processes
Gaussian Processes

- Distributions over functions \( f(x) \) of which the distribution is defined by a mean function \( m(x) \) and positive definite covariance function \( \Sigma = k(X, X') \), with \( (X, X') \) all possible pairs in the input domain \( x \in X \),

\[
f(x) \sim GP(m(x), k(x, x'))
\]

where for any finite subset \( X = \{x_1, ..., x_n\} \), the marginal distribution is a multivariate Gaussian distribution

\[
f(X) \sim \mathcal{N}(m(X), k(X, X))
\]

- Multivariate Gauss: captures a finite number of jointly distributed Gaussians
- Gaussian Process: mean and covariance are defined by a function; → Infinite dimensional Gaussian random variable
- Specification of covariance function (kernel) implies a distribution over functions \( f(x) \)
  - E.g exponential quadratic function (RBF)
  - Set prior information on this distribution

\[
k(x_a, x_b) = \exp\left(-\frac{1}{2\sigma^2}||x_a - x_b||^2\right)
\]
Gaussian Processes

- Flexible model of continuous functions
  - If you have no idea about the effects of the hyperparameters
  - Consider every possible function that matches the data
- Side note: Inference with GP requires matrix inversion (negligible for expensive model evaluations)
Exploration Strategy

- Policy which maps a model to the next query point (criteria for evaluating which hyperparameters to choose next from the surrogate model)

- Acquisition/Selection function
  - **Probability of improvement**: evaluates $f$ at the point most likely to improve upon this value
  - **Expected Improvement**: evaluates $f$ at the point that, in expectation, improves upon $f'$ (the minimal value of $f$ observed so far) the most
  - **Entropy search**: minimize the uncertainty we have in the location of the optimal value
  - **Upper confidence bound
  - ...  

- Maximize the acquisition function $\rightarrow$ Find best new sample point
Exploration – Exploitation Trade-off

- Improve on an already good sample point
- vs.
- Evaluation new points in unexplored areas

- Next point is not necessarily better point (with respect to the objective function), but a point that yields *most information*
Surrogate-based optimization

- Bayesian model-based optimization:
  - Build a probability model of the objective function to propose smarter choices for the next set of hyperparameters
- Sequential model-based optimization
  - Formalization of Bayesian Approach
  - More efficient at finding the best hyperparameters compared to random or grid search.

Problem:
- Sequentiallity
- Sampling strategy = Optimization → Computationally expensive
- Exploration vs. Exploitation trade-off
Evolutionary Strategies

Generic population-based metaheuristic optimization algorithms
Evolution ... Survival by adaptation to environment

- Population
- Genome: Properties of each individual
- Fitness: measure of degree of adaptation of organism to environment
- Reproduction: Adaptation to Selection
  - Recombination/Crossover:
    - How two individuals combine to form offspring
    - Inheritance of parental properties
- Mutation: introduces new „features“ → Variability
  - Can have positive, negative, or no effect
- Selection: the „weakest“ are removed from the gene pool (aka get eaten)
Evolutionary Strategies

P = Create_Population(N)
P.Calc_Fitness()

While (it < numberIterations):
    p1, p2 = Choose_Parents(P)
    child = Create_Offspring(p1, p2)
    Mutate_Offspring(child)
    fitness = Rate_Offspring(child)
    P.add(child, fitness)
    Remove_Unfittest(P)
Next
Evolutionary Strategies

- No assumption about the underlying “fitness landscape” → ability to approximate solutions to all types of problems
- Populations evolving in different niches can independently develop different (but similar) solutions to the same problem
- Solutions may reach a local optimum
- But: possibility that populations leave local optimum and find better solution
- Fitness of individuals may depend on the other individuals in the population
- Historically four “flavors”: genetic algorithms, evolutionary algorithms, genetic programming, evolutionary programming
  - differ in genetic representation, implementation details, nature of particular applied problem
Putting it all together…
### Frameworks offering Hyperparameter optimization

<table>
<thead>
<tr>
<th>Framework</th>
<th>Description</th>
<th>Grid/Random</th>
<th>SMBO</th>
<th>Evolutionary</th>
</tr>
</thead>
<tbody>
<tr>
<td>scikit-learn</td>
<td>Python package</td>
<td>✓</td>
<td>(✓) Auto-sklearn</td>
<td>(✓) sklearn-deap</td>
</tr>
<tr>
<td>Tune</td>
<td>Python library for distributed HP tuning</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Talos</td>
<td>For Keras</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Keras Tuner</td>
<td>For Keras / Tensorflow</td>
<td>✓ (Random)</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>hyperopt (hyperas, hyperopt-sklearn)</td>
<td>Python packages</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Katib</td>
<td>Kubernetes-native system</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Optuna</td>
<td>Python package for black box optimization</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>tuneRanger</td>
<td>R package for tuning random forests using MBO</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>SMAC</td>
<td>Python/Java library</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>deap</td>
<td>Python framework for general evolutionary computation</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>BOCS</td>
<td>Matlab package for minimizing a black-box function</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>
References & Acknowledgements

- Matthew Hoffman „Bayesian Optimization“, UAI 2018 https://www.youtube.com/watch?v=C5nqEHpdyoE
- Oskar Knagg „An intuitive guide to Gaussian processes” https://towardsdatascience.com/an-intuitive-guide-to-gaussian-processes-ec2f0b45c71d
- Wikipedia, XKCD, Sci-Kit Learn documentation
- ...
No Free Lunch Theorem

No algorithm exists which outperforms every other algorithm for every problem
Hyperparameter Optimization

Part II: hands-on course
Excercises

- **Aim:** hyperparameter optimization of a regression problem:
  - manual approach (1st Jupyter notebook)
  - machine-assisted approach (2nd Jupyter notebook)

- **Download:** https://github.com/DLR-SC/Hyperparameter_tutorial
The aim of this notebook is the following:

- Introduction of the **Boston house price regression problem**.
- A (very) short introduction of **Keras**.
- **Manual approach** to optimize hyperparameters (i.e. one exercise).

---

**Exercise 1:**

- Play around with the different hyperparameters to build a better regression model
- Can you achieve a better result as build_better_model()

```python
from keras import regularizers

def build_better_model_task():
    model = models.Sequential()
    
    # replace 'number_of_neurons' with a value of your choice, e.g. 8, 16, 32, 64, 128
    number_of_neurons = 32
    model.add(layers.Dense(number_of_neurons, activation='relu', input_shape=(train_data.shape[1], )))
    
    # maybe add a second hidden layer?
    # model.add(layers.Dense(XXX, activation='relu'))
```
The aim of the 2nd notebook is:
- Introduction of Talos and machine-assisted hyperparameter optimization
- Give general guidelines on hyperparameter optimization

Due to the limited computing time (i.e. we want to have lunch at 12pm) you will consider a „toy problem“ with all its drawbacks (low number of training samples, …)

Instead of focusing too much on the specific problem, the important aspect is understanding the machine-assisted approach itself.

Try to solve the exercises yourself and do not directly look at WAW_Hyperparameter_Tutorial_Part2_solution.ipynb

Thank you for your attention and enjoy!
Wrap-up: Guidelines on machine-assisted hyperparameter optimization

- Hyperparameter optimization is an **iterative process**.
  - find out the **important parameters** for your specific problem (i.e. that correlate with performance metric)
  - the parameter boundaries should decrease with every iteration
- Avoid a full grid parameter search. Use other search strategies such as **random search** instead.
- Use an **additional validation set** for finding adequate hyperparameters, not the test set.
- Important to first get some kind of understanding which **model architecture** might be adequate.
Thank you for your participation!

We hope that you have enjoyed this tutorial.
Charlotte Debus & Alexander Rüttgers