Practical Machine Learning in R

Tuning

Lars Kotthoff\textsuperscript{1,2}
larasko@uwyo.edu

\textsuperscript{1}with slides from Bernd Bischl and Michel Lang
\textsuperscript{2}slides available at http://www.cs.uwyo.edu/~larasko/ml-fac
Tuning

Hyperparameter Tuning

▷ used to find “best” hyperparameters for a method in a data-dependent way
▷ important to achieve good performance in practice
▷ essential for some methods, e.g. SVMs
Grid and Random Search

Grid Layout

Random Layout

Population-Based Methods

- e.g. Racing and Genetic Algorithms
- start with population of random configurations
- eliminate “weak” individuals
- generate new population from “strong” individuals
- iterate
Model-Based Search

▷ currently considered state-of-the-art
▷ build surrogate model of parameter-response surface
▷ evaluate cheap model instead of expensive target function
▷ use model to propose next point to evaluate target function at
▷ iterate
Model-Based Search – Components

- learner for surrogate model
- method for generating set of initial observations
- infill criterion – how to get next evaluation point
- termination criterion
Iter = 1, Gap = 1.9909e−01
Model-Based Search Example – 1D

Iter = 2, Gap = 1.9909e−01

\[
\begin{array}{c}
y(1) & y(2) & y(3) \\
0.0 & 0.4 & 0.8 \\
0.00 & 0.01 & 0.02 & 0.03 \\
\end{array}
\]
Iter = 3, Gap = 1.9909e−01
Model-Based Search Example – 1D

Iter = 4, Gap = 1.9992e−01
Model-Based Search Example – 1D

Iter = 5, Gap = 1.9992e−01
Model-Based Search Example – 1D

Iter = 6, Gap = 1.9996e−01
Model-Based Search Example – 1D

Iter = 7, Gap = 2.0000e−01
Model-Based Search Example – 1D

Iter = 8, Gap = 2.0000e−01
Model-Based Search Example – 1D

Iter = 9, Gap = 2.0000e−01

- Graph showing a 1D model-based search example.
- The graph displays a curve with markers indicating different types of points: red for initial points (init), blue for proposed points (prop), green for sequence points (seq), black for actual data points (y), and dashed black for predicted data points (yhat).
- The x-axis ranges from -1.0 to 1.0, and the y-axis ranges from 0.0 to 1.0 with logarithmic scale for y-axis values.
- The graph highlights the iteration and gap values.
Model-Based Search Example – 1D

Iter = 10, Gap = 2.0000e−01
Model-Based Search Example – 2D

- $y$
- $yhat$
- $ei$
- $se$

**Type**
- **init**
- **prop**
- **seq**
Model-Based Search Example – 2D

- **y**
  - Type: init
  - Prop: seq

- **yhat**
  - Type: init
  - Prop: seq

- **ei**
  - Type: init
  - Prop: seq

- **se**
  - Type: init
  - Prop: seq
Model-Based Search Example – 2D

- $y$
- $y_{\text{hat}}$
- $e_i$
- $s_e$

- $\text{type}$
  - $\text{init}$
  - $\text{prop}$
  - $\text{seq}$
Model-Based Search Example – 2D

- $y$
- $yhat$
- $ei$
- $se$

Type:
- \textcolor{red}{init}$
- \textcolor{blue}{prop}$
- \textcolor{green}{seq}$

Legend:
- 0.002
- 0.004
- 0.006
- 0.008
- 0.01
- 0.1
- 0.2
- 0.3
- 0.4
- 0.5
- 0.6
- 0.7
- 0.8
- 0.9
- 1.0
- 1.5
- 2.0
Model-Based Search Example – 2D
Model-Based Search Example – 2D

y

eyhat

ei

$y$
type
● init
△ prop

$yhat$
type
● init
△ prop

$ei$
type
● init
△ prop

$se$
type
● init
△ prop

seq

$se$
Model-Based Search Example – 2D

Graphs showing the model-based search example in 2D. The graphs display the following:
- **y** variable with points marked as red dots for initial points (init) and blue triangles for proposed points (prop).
- **yhat** variable with similar red and blue markers.
- **ei** variable with points distributed across the graph.
- **se** variable with points also distributed across the graph.

The graphs use color scales to represent different values of the variables.
Model-Based Search Example – 2D

- **y**
- **ei**
- **yhat**
- **se**

- Color bars:
  - **y**: 2.0 - 0.0
  - **ei**: 0.0020 - 0.0005
  - **yhat**: 1.5 - 0.0
  - **se**: 0.4 - 0.1

- Data points:
  - **init**, **prop**, **seq**
Model-Based Search Example – 2D

- y
- yhat
- ei
- se

Legend:
- **init**
- **prop**
- **seq**
When are we done?

- most approaches incomplete
- cannot prove optimality, not guaranteed to find optimal solution (in finite time)
- performance highly dependent on configuration space

→ How do we know when to stop?
Time Budget

How much time/how many function evaluations?

▷ too much $\rightarrow$ wasted resources
▷ too little $\rightarrow$ suboptimal result
▷ experiment with different settings
▷ run several times with different random initializations
Evaluation

▷ repeated evaluation with same train/test split statistically unsound → violates independence assumption
▷ example: parameters have no real effect, only random variation → still one parameter setting will “win”
▷ solution: different train/test splits
Nested Resampling

Outer resampling
Estimate performance

Use tuned parameters

Inner resampling
Tune parameters

Use tuned parameters

Training set outer resampling
Test set outer resampling
Training set inner resampling
Test set inner resampling
In mlr

▷ tuning with different methods available as wrapper
▷ model-based optimization available in mlrMBO package
▷ nested resampling available as resampling method
http://www.cs.uwyo.edu/~larsko/ml-fac/07-tuning-exercises.Rmd