Hyperparameter Tuning

July 6, 2025

What is Hyperparameter Tuning?

- Hyperparameters are "knobs" that control model training
- Examples: learning rate, max depth, number of estimators
- They are **not learned** from data — must be set manually
- Tuning is the process of finding the best values for these settings

Grid Search: Intuition

- Try all combinations of hyperparameters in a fixed grid
- Like a chef testing every recipe from a list of ingredients
- Works well when search space is small

Grid Search: Math & Algorithm

Objective:

$$\theta^* = \arg\min_{\theta \in \Theta} f(\theta)$$

Algorithm:

1. Define grid $\Theta = \theta_1 \times \cdots \times \theta_d$
2. For each $\theta \in \Theta$, train model and evaluate $f(\theta)$
3. Return best θ^*

Random Search: Intuition

- Sample random combinations of hyperparameters
- Like spinning a roulette wheel to find good combos
- Often outperforms grid search in high dimensions

Random Search: Math & Algorithm

Objective:

$$\theta^* = \arg\min_{\theta \sim p(\theta)} f(\theta)$$

Algorithm:

1. Define sampling distributions $p_1, \ldots, p_d$
2. For $i = 1$ to N: sample $\theta^{(i)}$, train model, evaluate
3. Return best $\theta^{(i)}$

Particle Swarm Optimization: Intuition

- Inspired by birds or fish searching for food
- Each "particle" (solution) flies in search space
- Particles learn from their own and neighbors' best positions

PSO: Equations

Velocity Update:

$$v_i(t+1) = wv_i(t) + c_1 r_1(p_i - x_i(t)) + c_2 r_2(g - x_i(t))$$

Position Update:

$$x_i(t+1) = x_i(t) + v_i(t+1)$$

Where:

- w: inertia weight
- c_1, c_2: cognitive/social coefficients
- $r_1, r_2 \sim \text{Uniform}(0, 1)$

PSO: Personal Best p and Global Best g

Personal Best p_i:

$$p_i = \arg \min_{x \in \{x_i(1), \dots, x_i(t)\}} f(x)$$

Best position particle i has seen so far (based on validation loss).

Global Best g:

$$g = \arg \min_i f(p_i)$$

Best position found by any particle in the swarm.

Intuition:

- Each particle remembers its own best solution — p_i
- Particles share knowledge to follow the global best — g
- Movement balances exploration (randomness) and exploitation (toward p_i and g)

PSO: Algorithm

1. Initialize particles x_i and velocities v_i
2. Evaluate $f(x_i)$, update personal best p_i and global best g
3. Update velocity and position:
$$v_i \leftarrow \text{update formula}$$
$$x_i \leftarrow x_i + v_i$$
4. Repeat for T iterations
5. Return best g

MoDeVa: Hyperparameter Tuning Overview

- MoDeVa supports 3 tuning strategies:
  - `ModelTuneGridSearch`
  - `ModelTuneRandomSearch`
  - `ModelTunePSO`
- Common arguments:
  - `param_distributions`: hyperparameter search space
  - `metric`: e.g., "MSE", "ACC", "F1"
  - `cv`: number of cross-validation folds
  - `n_iter`: only for random search

Example: Grid Search in MoDeVa

```
from modeva.models import ModelTuneGridSearch
model = mz.get_model("XGB-Depth2")

param_space = {
    "max_depth": [2, 4, 6],
    "learning_rate": [0.01, 0.1, 0.2]
}

hpo = ModelTuneGridSearch(dataset=ds, model=model)
result = hpo.run(param_distributions=param_space,
                 metric="MSE",
                 cv=5)
result.table
```

Example: Random Search in MoDeVa

```
from scipy.stats import uniform, randint
from modeva.models import ModelTuneRandomSearch
model = mz.get_model("XGB-Depth2")

param_space = {
    "learning_rate": uniform(0.001, 0.3),
    "n_estimators": randint(100, 1000)
}

hpo = ModelTuneRandomSearch(dataset=ds, model=model)
result = hpo.run(param_distributions=param_space,
                 n_iter=10,
                 metric="MSE",
                 cv=5)
result.table
```

Example: PSO Search in MoDeVa

```
from modeva.models import ModelTunePSO
model = mz.get_model("XGB-Depth2")

param_space = {
    "learning_rate": (0.001, 0.3),
    "max_depth": (2, 8)
}

hpo = ModelTunePSO(dataset=ds, model=model)
result = hpo.run(param_distributions=param_space,
                 metric="MSE",
                 cv=5)
result.table
```