Module 18: Hyperparameters and Hyperparameter Tuning

Learning outcomes:

1. Identify the role of hyperparameters and regularisation techniques in machine learning models.
2. Identify the challenges involved in hyperparameter tuning.
3. Identify the benefits and drawbacks of common hyperparameter tuning methods.
4. Discuss contemporary research and competition insights on hyperparameter tuning.
5. Evaluate a hyperparameter tuning approach and its future applications.

Hyperparameters

Hyperparameters are the external settings defined before training that determine how a model learns. They differ from parameters, which are learned automatically by the model during training.

Parameters vs hyperparameters

- **Parameters:** learned from data (e.g. weights in neural networks, regression coefficients)
- **Hyperparameters:** set by the user (e.g. learning rate, batch size, number of epochs, tree depth, dropout rate)

Why hyperparameters matter

- Hyperparameters control model complexity and training dynamics.
- Poor choices can lead to underfitting (too simple) or overfitting (too complex).
- Regularisation methods, such as dropout, weight decay and early stopping, rely on hyperparameters to improve generalisation.

Gradient descent with momentum

- Standard gradient descent can converge slowly or get stuck in local minima.
- Adding momentum helps accelerate learning by combining past updates with current gradients.
- Analogy: like pushing a ball down a hill, momentum builds speed in consistent directions and smooths noisy updates.

Challenges of hyperparameter tuning

- **Search space size:** there are many hyperparameters, each with multiple possible values.
- **Training cost:** evaluating each configuration can be computationally expensive.

- **Dependency:** some hyperparameters only matter if others are active.
- **Local optima:** search spaces may have multiple ‘good enough’ regions, requiring global search methods.

Methods for hyperparameter tuning

- **Manual search:** trial and error, guided by experience
- **Grid search:** exhaustive exploration of defined ranges; simple but expensive
- **Random search:** samples configurations at random; more efficient in large spaces
- **Bayesian optimisation:** builds a surrogate model (e.g. Gaussian process) to guide search strategically
- **Hyperband:** allocates resources adaptively, stopping poor performers early
- **Evolutionary algorithms:** evolve candidate solutions through selection, crossover and mutation
- **Gradient-based optimisation:** directly updates differentiable hyperparameters using gradients

Key takeaways from NeurIPS 2020

- Winning teams combined massive compute power with ensemble methods.
- Handling non-stationarity and heteroscedasticity in search spaces was critical.
- Approaches such as TuRBO and warped Gaussian processes helped adapt models to changing function behaviours.
- AutoML continues to grow as a way to automate hyperparameter selection entirely.