Fine-tuning Your Model: A Guide to Optimisation

We'll explore the essential components that drive machine learning performance, from the fundamental building blocks of models and parameters to the sophisticated techniques of optimisers and hyperparameter search.

Models & Parameters

Models: The Learning Algorithms

Models are the core algorithms that learn patterns and relationships from data. They range from simpler statistical methods to complex deep learning architectures.

- **Examples:** Linear Regression, Logistic Regression, Decision Trees, Random Forests, Neural Networks.

Parameters: The Learned Variables

Parameters are internal variables within the model that are automatically learned from the training data during the optimisation process. These values define the model's specific mapping from inputs to outputs.

- **Regression Models:** Coefficients and intercept terms.
- **Decision Trees:** Split points and leaf values.
- **Neural Networks:** Weights and biases connecting neurons.

Optimisers: The Engine of Learning

Optimisers are algorithms designed to update a model's internal parameters (weights, coefficients) iteratively to minimise a predefined loss function. Their primary goal is to find the set of parameter values that results in the lowest possible error or discrepancy between the model's predictions and the actual data. They achieve this by using the gradient (the slope) of the loss function to guide the adjustment of weights in the direction that reduces the loss.

Core Gradient Descent (GD) Types

- **Batch Gradient Descent:** Uses the entire dataset for each parameter update. It's stable and converges smoothly but can be very slow for large datasets.
- **Stochastic Gradient Descent (SGD):** Updates parameters using only one sample at a time. This makes it fast and responsive but can result in noisy updates and less stable convergence.
- **Mini-Batch Gradient Descent:** Strikes a balance by updating parameters using small, user-defined batches of data. This offers both speed and stability, making it a widely adopted approach.

Advanced Optimiser Variants

- **Momentum:** Incorporates a fraction of the previous update vector to accelerate convergence in the right direction, helping to overcome local minima.
- **RMSProp:** Adapts the learning rate for each parameter by dividing it by an exponentially decaying average of squared gradients. This helps in dealing with different scales of gradients.
- **Adam (Adaptive Moment Estimation):** Combines the best features of Momentum and RMSProp. It computes adaptive learning rates for each parameter by storing an exponentially decaying average of past squared gradients and past gradients. It's often the default choice.
- **Adagrad:** Adapts the learning rate to the parameters, performing larger updates for infrequent parameters and smaller updates for frequent parameters. This is particularly effective for sparse data.

Hyperparameters: Guiding Model Behaviour

Unlike parameters, hyperparameters are external configurations that are not learned from the data but are set by the practitioner before the training process begins. They play a crucial role in controlling the model's behaviour, influencing its learning speed, structure, and ability to generalise to new, unseen data. Selecting appropriate hyperparameters is key to achieving optimal model performance.

1

Model Hyperparameters

These define the intrinsic structure and complexity of the model.

- **Decision Trees:** Maximum depth, minimum samples per leaf.
- **Random Forests:** Number of individual trees in the ensemble.
- **Neural Networks:** Number of hidden layers, number of neurons per layer.

2

Training Hyperparameters

These dictate the dynamics of the learning process its

- **Learning Rate:** Crucial for optimisers like Gradient Descent and Adam, controlling the step size during parameter updates.
- **Batch Size:** The number of samples processed before the model's internal parameters are updated.
- **Number of Epochs:** The total number of complete passes through the entire training dataset.
- **Dropout Rate (NNs):** The fraction of neurons randomly set to zero during training to prevent overfitting.

3

Regularisation Hyperparameters

These are employed to prevent overfitting and improve generalisation.

- **L1/L2 Penalty Strength:** Controls the magnitude of the regularisation applied to model parameters.
- **Early Stopping Criteria:** Rules to halt training when the model's performance on a validation set begins to degrade, despite continued improvement on the training set.

Hyperparameter Search Methods

Since hyperparameters are manually set, finding their optimal values is a critical step in model development. This often involves systematic search strategies to explore the hyperparameter space and identify the combination that yields the best model performance.

Manual Search

A trial-and-error approach where practitioners intuitively adjust hyperparameters based on experience and observed performance. Most suitable for simple models or initial exploration.

Grid Search

An exhaustive search method that evaluates the model for every possible combination of hyperparameters within a predefined grid. Guarantees finding the best combination within the specified range but is computationally intensive and scales poorly with more hyperparameters.

Random Search

Instead of an exhaustive grid, random search samples hyperparameters randomly from the defined search space. It's often more efficient than grid search, especially when only a few hyperparameters significantly impact performance.

Bayesian Optimisation

Constructs a probabilistic model of the objective function (e.g., model performance) and uses it to select the most promising hyperparameters to evaluate next. This intelligent approach is highly efficient for expensive models, as it minimises the number of evaluations.

Advanced Methods

- **Hyperband:** An adaptive resource allocation method.
- **Genetic Algorithms:** Evolutionary strategies for search.
- **Population-Based Training:** Used in deep reinforcement learning.

Conceptual Flow: Tuning for Success

1. Choose a Model

Select an appropriate model, e.g., Random Forest or Neural Network, based on the problem and data characteristics.

2. Set Initial Hyperparameters

Define starting values for key hyperparameters, such as learning rate and tree depth.

3. Use Optimiser

Employ an optimiser to iteratively update the model's internal parameters during training.

4. Evaluate Performance

Assess the model's performance on validation data to gauge its effectiveness and identify areas for improvement.

5. Adjust via Search Methods

Apply hyperparameter search methods to refine hyperparameter settings for better results, looping back to step 2.

This iterative cycle ensures that models are continuously improved, leading to more accurate and generalisable predictions. Mastering this flow is essential for any machine learning practitioner.

End
Questions ??

Will try and answer best of my understanding

….