Machine Learning/Deep Learning Concepts

Aashish Mahato

Agenda

+ Fine-Tuning
+ Model Parameters and Hyperparameters
+ Hyperparameter Tuning
+ Loss function/Cost function
+ Optimizer
+ Cross-Validation
+ K-Fold Cross-Validation

Fine-Tuning

What is Fine-Tuning?

+ Fine-tuning is the process of taking an already trained or pre-trained machine learning model and further training it on a smaller, targeted data set so it can perform specialized tasks.

+ Fine-tuning enables developers to save time and money by building on existing models instead of creating new ones.

+ For example, a pre-trained LLM model might have a general grasp of English, but a law firm may fine-tune it to understand legal jargon specifically.

Types of Fine-Tuning

Feature extraction fine-tuning

+ Feature extraction fine-tuning, sometimes referred to as parameter-efficient fine-tuning, “freezes” the top layers of the model while the final layers train on the specified data set. This keeps most of the existing model intact, making it more computationally efficient.

Types of Fine-Tuning

Full model fine-tuning

+ Full model fine-tuning is similar to pre-training in that the entire model trains on a data set and updates the weights accordingly. This process updates the whole model, potentially leading to stability issues.
+ This fine-tuning method is more computationally intensive than feature extraction fine-tuning. It's best suited to situations where the new data set is both sizable and distinct from the original training data set.

How does fine-tuning work?

+ Fine-tuning adapts a pre-trained model to a specific task through a systematic process, as outlined below:

  - Start with a pre-trained model: Choose a pre-trained model, such as a large language model (LLM) (from platforms like Hugging Face or Kaggle), that has already been trained on extensive datasets. Use this model as a foundation because it already encodes general features and broad knowledge. For example, start with a model like Llama.

How does fine-tuning work?

- Prepare a task-specific dataset: Collect and organize a dataset specific to the task you want the model to perform. Include labeled examples that reflect your domain. For instance, in a healthcare chatbot, a task-specific dataset could include patient inquiries (e.g., “What are the symptoms of the flu?”) paired with medically accurate responses.

How does fine-tuning work?

- Fine-tune the model: Use the task-specific dataset to fine-tune the pre-trained model and adjust its weights.
  - Focus on optimizing the later layers of the model to learn task-specific features. For example, train the model to classify sentiment in customer reviews to summarize product descriptions or improve sales strategies.
  - Decide whether to freeze or lightly fine-tune the early layers to preserve general features while adapting to your specific task.

How does fine-tuning work?

- Validate the model: Test the fine-tuned model on a validation set to measure its performance and ability to generalize to new data.
- Prepare the fine-tuned model for deployment by exporting it in a format compatible with your production environment, such as ONNX or TensorFlow SavedModel.

Benefits of fine-tuning

+ Cost savings: Fine-tuning is more affordable than developing a model from scratch. It reduces time, computing requirements, and the need for new hardware.

+ Improved accuracy for niche applications: Fine-tuning enhances performance for specialized tasks by refining a pre-trained model with a smaller, domain-specific data set.

+ Greater accessibility: Smaller organizations can leverage pre-trained models and fine-tune them with their own data, making advanced machine learning more attainable.

Challenges of fine-tuning

+ Due to the small data sets used in fine-tuning, you can risk overfitting your model. An overfitted model is so attuned to the training data that it doesn’t perform well with new data.

+ Fine-tuning a model can risk having the pre-trained model “forget” some of its original training. This happens when you target specific layers too heavily or use a data set that's too different from the training set.

+ If the original pre-trained model has certain biases, flaws, or security issues, your fine-tuned model will also have those issues.

Model Parameters and Hyperparameters

What are Hyperparameters?

+ Hyperparameters are external configuration variables that data scientists use to manage machine learning model training.

+ Sometimes called model hyperparameters, the hyperparameters are manually set before training a model.

+ Examples of hyperparameters include the number of nodes and layers in a neural network and the number of branches in a decision tree. Hyperparameters determine key features such as model architecture, learning rate, and model complexity.

What are Parameters?

+ Parameters on the other hand are internal to the model. That is, they are learned or estimated purely from the data during training as the algorithm used tries to learn the mapping between the input features and the labels or targets.

+ For example, weights or coefficients of independent variables in the linear regression model, weights and biases of a neural network, and cluster centroids in clustering algorithms.

Hyperparameter Tuning

What is Hyperparameter Tuning?

+ Hyperparameter tuning is the practice of identifying and selecting the optimal hyperparameters for use in training a machine learning model.

+ Hyperparameter tuning is an experimental practice, with each iteration testing different hyperparameter values until the best ones are identified.

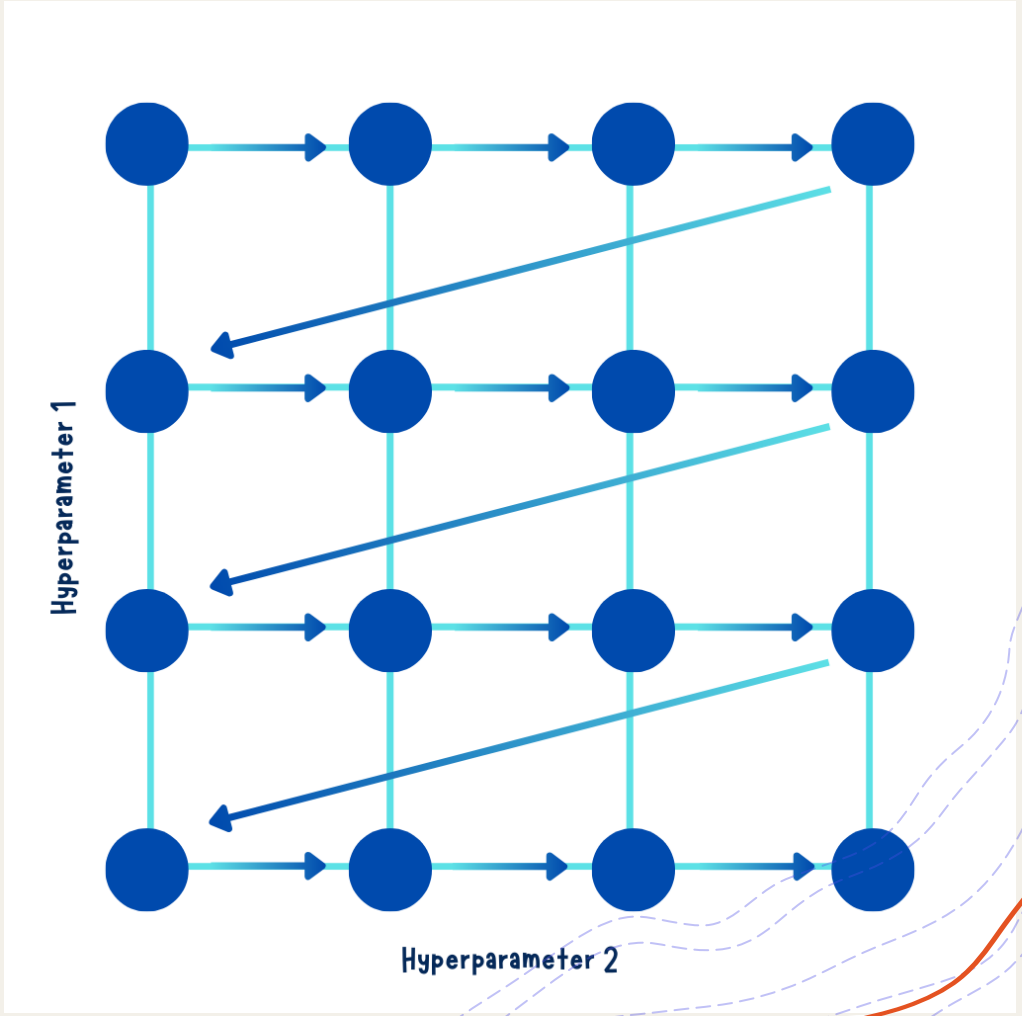
Methods for Hyperparameter Tuning

+ Numerous hyperparameter tuning algorithms exist, the most commonly used types are given below:

- Grid Search
- Random Search
- Genetic Search
- Bayesian Optimization

Grid Search

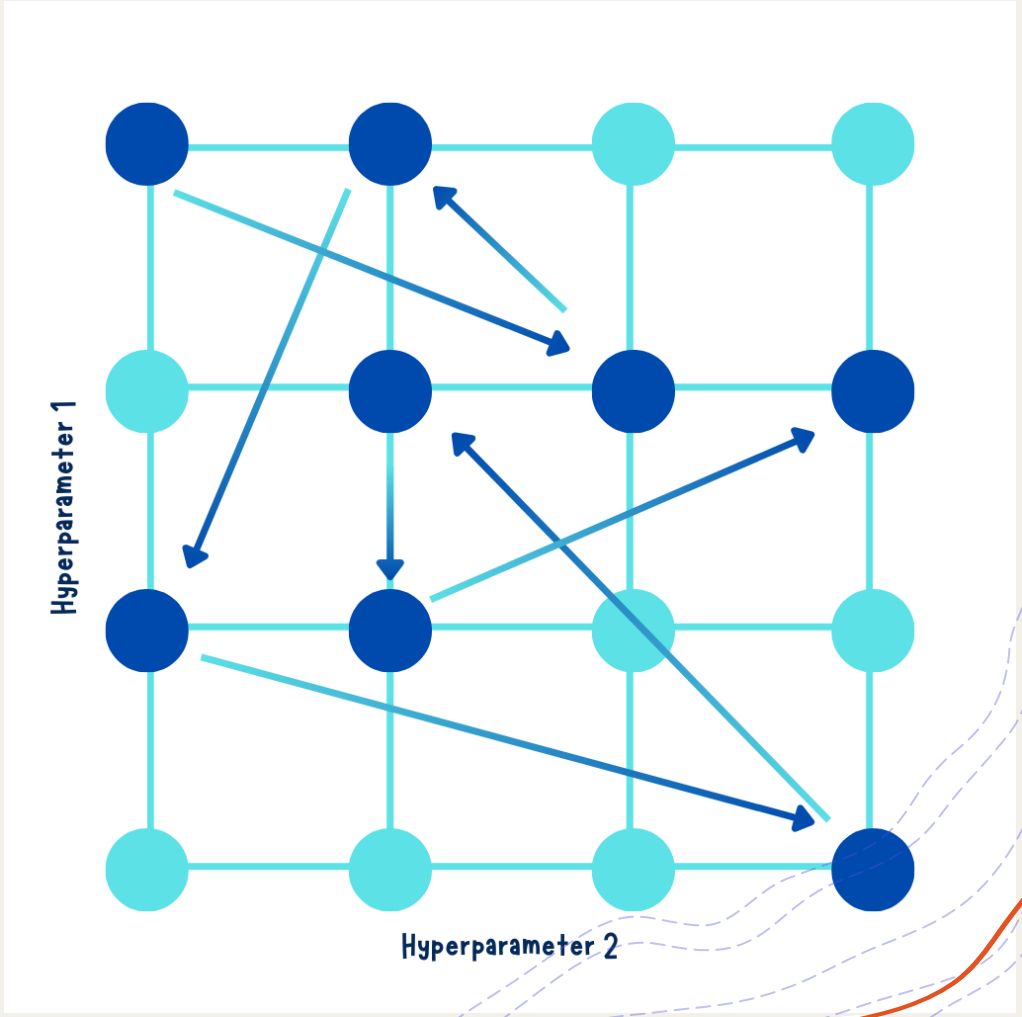
+ Grid search is a brute-force approach. It systematically evaluates all possible combinations of hyperparameter values.
+ How it Works:
    - Define a grid of hyperparameter values.
    - Train the model for every combination of values in the grid.
    - Evaluate each model using cross-validation.
    - Select the combination that performs best.



Random Search

+ Random search explores random combinations of hyperparameters within specified ranges. It doesn't evaluate all combinations, making it faster than grid search.

+ How it Works:
    - Specify ranges for hyperparameters.
    - Randomly sample combinations and evaluate the model.
    - Select the best combination based on performance.



Genetic Search

+ Genetic Algorithms (GAs) are a class of optimization and search algorithms inspired by the process of biological evolution. They simulate the process of natural selection to find solutions to complex problems.

+ The key components of a genetic algorithm include:

    - Population: A population represents a group of potential solutions to a problem. These solutions are often referred to as "individuals" or "chromosomes."

Genetic Search

- Selection: The selection process determines which individuals from the population will be chosen as parents for reproduction based on their fitness (how well they solve the problem).
- Crossover (Recombination): Crossover involves combining genetic information from two parent individuals to create one or more offspring. This process mimics genetic recombination in biology.

Genetic Search

- Mutation: Mutation introduces small random changes or perturbations into the genetic information of individuals. It adds an element of diversity to the population, allowing for exploration of new solutions.
- Fitness Function: The fitness function quantifies how well an individual solves the problem. It assigns a fitness score to each individual based on their performance, and this score guides the selection process.

Loss Function/Cost Function

What is a loss function/cost function?

+ A loss function is used to measure model performance by calculating the deviation of a model's predictions from the correct, "ground truth" predictions.

+ In Machine Learning, we have multiple observations using which we train our models to solve a particular problem statement. The cost function is nothing but the average of the loss values coming from all the data samples.

Difference between loss function and cost function

+ We usually consider both terms synonyms and can use them interchangeably. But, the Loss function is associated with every training example, and the cost function is the average of the loss function values over all the data samples. In Machine learning, we optimize our cost rather than our loss function.

Types of Loss/Cost Function

Regression loss functions

+ Regression models deal with predicting a continuous value for example salary of an employee, price of a car, loan prediction, etc. A cost function used in the regression problem is called “Regression Cost Function”. They are calculated on the distance-based error.
+ Examples include, Mean Squared Error (MSE), Mean Absolute Error (MAE), Root Mean Squared Error (RMSE)

Types of Loss/Cost Function

Classification loss functions

+ Cost functions used in classification problems are different than what we use in the regression problem. A commonly used loss function for classification is cross-entropy loss.

Optimizer

What is an optimizer?

+ Optimizers are algorithms or methods used to minimize an error function (loss function) or to maximize the efficiency of production.

+ Optimizers are mathematical functions which are dependent on model's learnable parameters i.e., Weights & Biases.

+ Optimizers help to know how to change weights and learning rate of neural network to reduce the losses.

Types of optimizers

+ Gradient Descent
+ Stochastic Gradient Descent
+ Mini-Batch Gradient Descent
+ SGD with Momentum
+ AdaGrad(Adaptive Gradient Descent)
+ RMS-Prop (Root Mean Square Propagation)
+ Adam(Adaptive Moment Estimation)

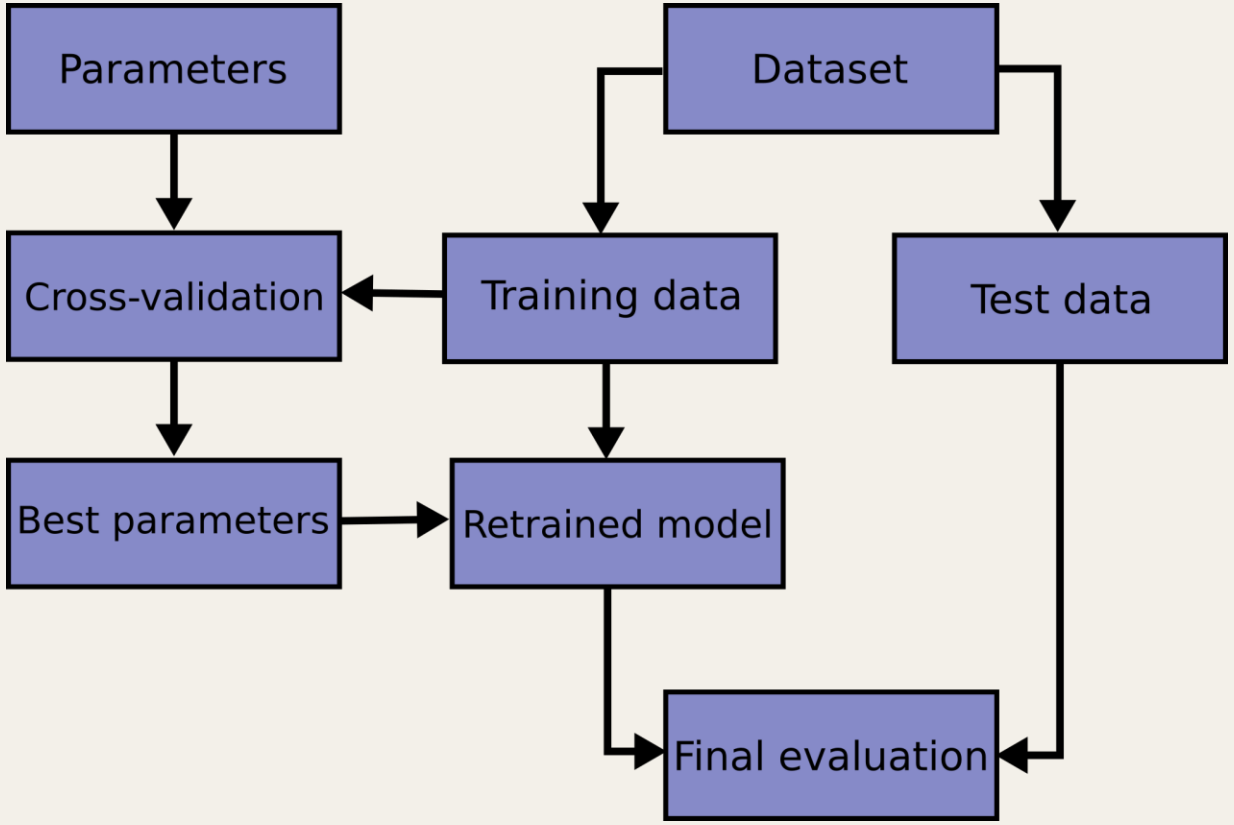
Cross-Validation

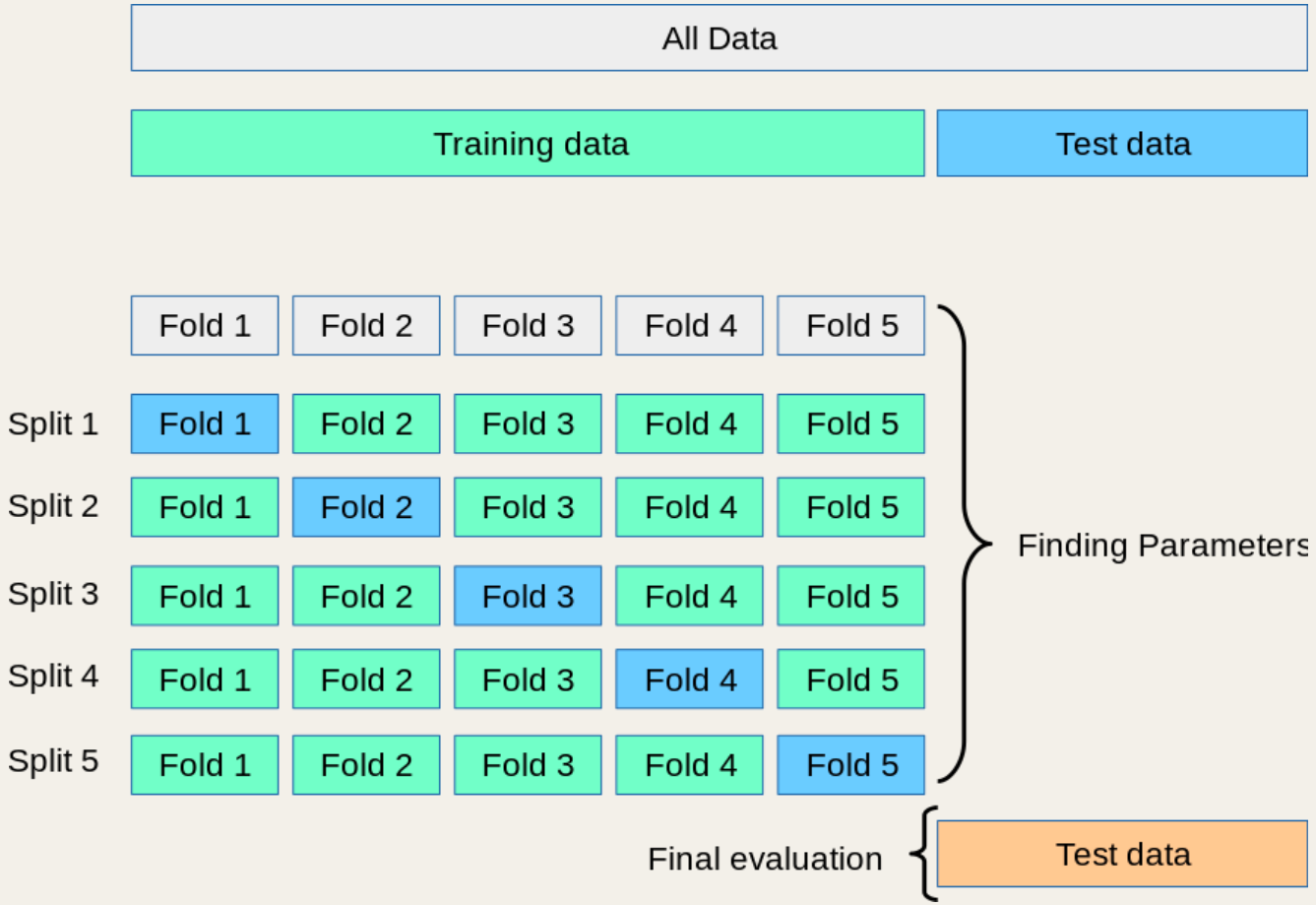
What is cross-validation?

+ Cross-validation is one of the most important concepts in machine learning. This is because it allows us to create models capable of generalization – that is, capable of creating consistent predictions even on data not belonging to the training set.

+ Cross-validation means dividing our training data into different portions and testing our model on a subset of these portions. The test set continues to be used for the final evaluation, while the model performances are evaluated on the portions generated by the cross-validation. This method is called K-Fold cross-validation.

Cross-Validation





K-Fold Cross-Validation

Thank You