AMNIL TECHNOLOGIES

Presentation on Model Fine Tuning and Hyperparameter Optimization

Prepared by: Sejal Panta
19th Sept, 2025

Model Fine-Tuning

- Fine-tuning is the process of adjusting a machine learning model's parameters and hyperparameters to improve performance on a specific task or dataset.

- It ensures the model generalizes well and avoids overfitting or underfitting.

Example: Image Classification

- A model like ResNet was trained on ImageNet (1.2 million images, 1000 categories like cats, cars, fruits, etc.).
- Now, you want to build a model to classify types of flowers (rose, tulip, daisy, sunflower, etc.).

Types of Parameters

1. Model Parameters

- Learned from training data during model fitting.
- Example: weights and biases in neural networks, coefficients in linear regression.
- Directly impact predictions.
- Found by optimization algorithms (e.g., gradient descent).

2. Hyperparameters

- Set before training, not learned from data.
- Define how the model is trained.
- Examples: learning rate, number of hidden layers, batch size, regularization strength.
- Chosen using search methods like Grid Search, Random Search, or Genetic Algorithms.

Hyperparameter Tuning Methods

1. Grid Search

- Exhaustively tests all possible combinations of hyperparameters.
- Guarantees best result from tested set.
- Very computationally expensive.

2. Random Search

- Samples random combinations of hyperparameters.
- Faster than grid search, covers wide search space.
- Often finds near-optimal solutions.

3. Genetic Algorithm

- Inspired by natural selection (mutation, crossover, selection).
- Evolves hyperparameter sets across generations.
- Good for complex, non-linear search spaces.

4. Bayesian Optimization

- Uses previous test results to intelligently guess the next best hyperparameters.
- Efficient, fewer trials needed.
- More complex to implement.

Optimization Concepts

1. Cost Function (Loss Function)

- A mathematical formula that measures error between predicted output and actual output.
- It guides how well the model is performing.

Examples:

- Regression → Mean Squared Error (MSE):

$$MSE = \frac{1}{n}\sum(y_{pred} - y_{true})^2$$

- Classification → Cross-Entropy Loss:

$$-\frac{1}{n}\sum y \log(\hat{y})$$

2. Gradient Descent

- Iteratively updates parameters to minimize the cost function in the direction of steepest descent of cost function.
- Update rule:
  $\theta = \theta - \alpha \cdot \nabla J(\theta)$
  Where, θ: model parameters, α = learning rate, $\nabla J(\theta)$: slope/gradient of cost function

Types of Gradient Descent:

- Batch Gradient Descent → uses all data in each update (accurate but slow).
- Stochastic Gradient Descent (SGD) → updates using one sample at a time (fast, noisy).
- Mini-Batch Gradient Descent → compromise, uses small batches (widely used).

3. Optimizers

- Algorithms that improve gradient descent for faster and more reliable convergence.
- Handle issues like slow training, oscillations, and vanishing gradients.

Common Optimizers:

- SGD (Stochastic Gradient Descent) → baseline optimizer.
- Momentum → remembers past gradients to smooth updates.
- Adam (Adaptive Moment Estimation) → adapts learning rate for each parameter, most popular.
- RMSprop → scales learning rate based on recent gradients, good for RNNs.
- Adagrad → adapts learning rate for sparse data (works well for text).

How to Fine-Tune a Model?

1. Choose a base model (pre-trained or simple).
2. Define performance metric (accuracy, RMSE, F1-score, etc.).
3. Select hyperparameters to tune.
4. Apply search methods (Grid/Random/Genetic Algorithm).
5. Train multiple models with different hyperparameters.
6. Evaluate using cross-validation or test set.
7. Pick the best-performing configuration.

Importance of Fine-Tuning

- Improves model accuracy and generalization.
- Prevents underfitting/overfitting.
- Adapts pre-trained models to specific tasks (transfer learning).
- Optimizes training efficiency (faster convergence, fewer resources).
- Enhances model robustness for real-world use.

Why Fine-Tuning is Performed?

- To maximize predictive performance.
- To adjust models for specific datasets/domains.
- To balance trade-off between bias and variance.
- To reduce computational cost by finding efficient configurations.
- To make models production-ready.

Challenges of fine-tuning:

- Overfitting – model memorizes small dataset.
- Catastrophic forgetting – loses old knowledge.
- Data issues – not enough, noisy, or biased data.
- High cost – needs lots of GPU, time, memory.
- Hyperparameter tuning – tricky to get right.
- Domain shift – struggles if new data is very different.
- Storage – fine-tuned models take space to manage

Example: Titanic dataset

- Here, fine-tuning Random Forest's depth, number of trees, and Logistic Regression's regularization significantly improves accuracy since adjusting hyperparameters (like tree depth, number of trees) helps the model better listen to important features (e.g., age, gender, ticket class).
- Concepts applied: hyperparameter tuning (grid/random search), cross-validation, cost function evaluation, optimizers.
- Advantage: Higher survival prediction accuracy with fewer errors.

Benefits:

- Higher accuracy (predicts survival more correctly).
- Avoids overfitting (doesn't just memorize training data).
- Works better on new, unseen passengers (test set).
- Makes the model reliable for real-world use.

THANK YOU!