Why ML Strategy?: Understanding the importance of defining a structured approach to machine learning projects for efficient problem-solving.

Orthogonalization: A principle in ML strategy that breaks down complex problems into independent components to improve performance.

Setting Up your Goal

Single Number Evaluation Metric: Choosing a single metric to evaluate model performance, ensuring clarity in comparison and optimization.

Satisficing and Optimizing Metric: Balancing trade-offs between multiple evaluation metrics to achieve practical and effective model performance.

Train/Dev/Test Distributions: Understanding how to split data into training, development, and test sets to prevent data leakage and ensure robust model evaluation.

Size of the Dev and Test Sets: Deciding the appropriate size of the development and test sets to obtain reliable performance estimates.

When to Change Dev/Test Sets and Metrics?: Recognizing situations where modifying data splits or evaluation metrics is necessary for improved model assessment.

Comparing to Human-level Performance

Why Human-level Performance?: Using human performance as a benchmark to determine the potential accuracy of machine learning models.

Avoidable Bias: Identifying and reducing systematic bias in models to improve performance and generalization.

Understanding Human-level Performance: Analyzing how well humans perform on specific tasks to guide the expectations for ML models.

Surpassing Human-level Performance: Exploring strategies to develop ML models that exceed human-level accuracy through advanced training techniques.

Improving your Model Performance: Methods for refining models, including data augmentation, hyperparameter tuning, and better feature selection.

Error Analysis

Carrying Out Error Analysis: A process of systematically identifying and categorizing model errors to improve performance.

Cleaning Up Incorrectly Labeled Data: Detecting and correcting mislabeled data points to ensure accurate model training and evaluation.

Build your First System Quickly, then Iterate: An approach emphasizing rapid prototyping and iterative improvements for effective model development.

Mismatched Training and Dev/Test Set

Training and Testing on Different Distributions: Handling cases where the training data distribution differs from the testing distribution to enhance model generalization.

Bias and Variance with Mismatched Data Distributions: Analyzing how bias and variance are affected when training and testing data are not from the same distribution.

Addressing Data Mismatch: Strategies for bridging the gap between different data distributions, such as domain adaptation techniques.

Learning from Multiple Tasks

Transfer Learning: A technique where a pre-trained model is adapted to a new but related task to improve learning efficiency.

Multi-task Learning: Training a model on multiple tasks simultaneously to leverage shared knowledge and improve performance across tasks.

End-to-end Deep Learning

What is End-to-end Deep Learning?: A model training approach where raw input data is processed directly through a neural network without manual feature engineering.

Whether to use End-to-end Deep Learning: Considerations for applying end-to-end deep learning, including data availability, computational cost, and task complexity.