CAP 4630 – Ensemble Models

Instructor: Aakash Kumar

University of Central Florida

Introduction to Ensemble Models

- Definition: Ensemble modeling is a technique in machine learning that combines multiple models to achieve better predictive performance.
- Key Concept: The core idea is that several weak learners (models with limitations when used individually) can work together to form a strong learner with improved accuracy and robustness.

Why Ensemble Models Work

- Individual Model Limitations: A single model, or weak learner, often suffers from high bias (underfitting) or high variance (overfitting).
- Combining Models: Aggregating weak learners can counteract these limitations, reducing either bias or variance depending on the technique, and ultimately yielding a more accurate and generalizable model.

How Ensemble Models Work

- Process Overview:
  - Step 1: Train multiple machine learning models independently on the same dataset or different subsets.
  - Step 2: Aggregate their predictions through methods like voting, averaging, or weighted averaging.
- Goal: By combining predictions, ensemble models capitalize on the strengths of each learner and offset their weaknesses.

Advantages of Ensemble Models

- Complementary Strengths: Ensemble methods allow different models to complement each other and overcome individual weaknesses.
- Variance Reduction: Combining models helps in reducing variance, leading to a more stable and reliable prediction.
- Reduced Overfitting: The ensemble approach can prevent models from memorizing noise in the training data, making them more robust on unseen data.

Popular Ensemble Techniques

- Bagging: Reduces variance by training models on different random samples and averaging their predictions. Example: Random Forest.
- Boosting: Reduces bias by sequentially building models that correct errors made by the previous ones. Examples: AdaBoost, Gradient Boosting.
- Stacking: Combines predictions from different types of strong learners by training a meta-model on their outputs to improve final prediction accuracy.
- Applications: Used in tasks like classification, regression, and clustering to enhance model robustness and accuracy.

Understanding Expected Error in Ensemble Models

- Expected Test Error: The goal is to understand the components of the model's error.
- Error Decomposition:
  - **Variance**: Measures how much $f_D(x)$ (prediction from model trained on dataset D) deviates from $\bar{f}(x)$ (average prediction over different datasets).
  - **Bias**: Measures the difference between the average prediction $\bar{f}(x)$ and the true value $\bar{y}(x)$.
  - **Noise**: Represents the irreducible error due to randomness in the data.

$$\mathbb{E}_{D\sim P_n,(x,y)\sim P}\left[(f_D(x)-y)^2\right] = \text{Variance} + \text{Bias} + \text{Noise}$$

$$\mathbb{E}_{D\sim P_n,(x,y)\sim P}\left[(f_D(x)-y)^2\right] = \underbrace{\mathbb{E}_{x,D}\left[(f_D(x)-\bar{f}(x))^2\right]}_{\text{Variance}} + \underbrace{\mathbb{E}_{x,y}\left[(\bar{y}(x)-y)^2\right]}_{\text{Noise}} + \underbrace{\mathbb{E}_{x}\left[(\bar{f}(x)-\bar{y}(x))^2\right]}_{\text{Bias}}$$

where:

- **Variance**: $\mathbb{E}_{x,D}\left[(f_D(x)-\bar{f}(x))^2\right]$ – measures the variability of the model's predictions $f_D(x)$ around the average prediction $\bar{f}(x)$.
- **Noise**: $\mathbb{E}_{x,y}\left[(\bar{y}(x)-y)^2\right]$ – represents the inherent randomness in the data.
- **Bias**: $\mathbb{E}_{x}\left[(\bar{f}(x)-\bar{y}(x))^2\right]$ – measures the difference between the average model prediction $\bar{f}(x)$ and the true value $\bar{y}(x)$.

Goal: Reducing Variance in Ensemble Models

- Objective: Minimize variance to make model predictions stable and reliable.
- Strategy:
  - Train multiple models f_D on different samples, aiming for f_D to approach $\bar{f}$ (average model).
  - Use the Weak Law of Large Numbers: Averaging predictions over different models reduces variance, bringing the ensemble's prediction closer to the true mean.

Implementing Variance Reduction with Bagging

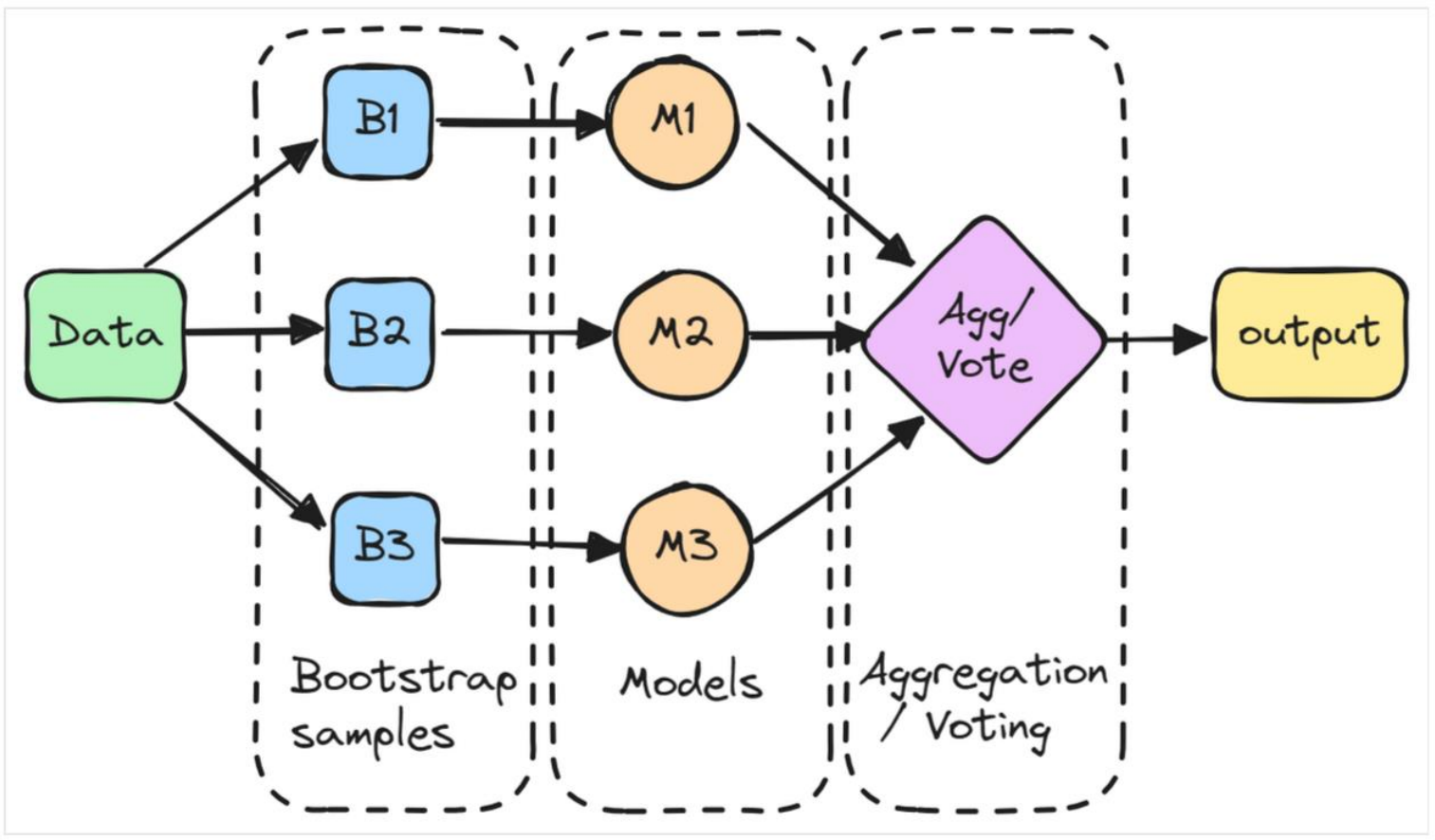
- Approach: Draw multiple random training datasets D_i
  - Draw multiple random training datasets $D_i \sim P$.
  - Train each model f_i independently.
  - Aggregate predictions: Use the average prediction to reduce variance
- Challenge: In practice, we may not have multiple datasets, so techniques like Bootstrap Aggregating (Bagging) are used to simulate this approach by resampling.

Bagging

Bagging (Bootstrap Aggregating)

- Definition: Bagging is an ensemble method where multiple models are trained independently on different random subsets of the original dataset, sampled with replacement (bootstrap samples).
- Purpose: This method reduces variance and improves model stability by combining predictions from multiple models, often through voting (for classification) or averaging (for regression).
- Key Inventor: Proposed by Leo Breiman as a robust method to mitigate overfitting, especially for high-variance models like decision trees.

Bagging (Bootstrap Aggregating)



How Does Bagging Work?

- Bootstrap Sampling: Each model is trained on a random subset of the data, known as a bootstrap sample. These samples are drawn with replacement, allowing individual data points to be chosen multiple times in one subset.
- Training Multiple Models: By training models on different bootstrap samples, Bagging reduces variance across the models, leading to a more stable and robust ensemble.
- Aggregation of Predictions:
  - For classification tasks, the predictions from each model are combined through majority voting.
  - For regression tasks, averaging the predictions is used to create the final output.
- Effectiveness: Bagging is particularly effective at reducing overfitting, especially with high-variance models like decision trees. It allows the ensemble to capture the strengths of individual models while canceling out their errors.

Advantages of Bagging - Overfitting and Variance Reduction

- Reduces Overfitting: Bagging minimizes the risk of overfitting by using multiple models trained on different data subsets, resulting in improved accuracy on unseen data.
- Decreases Model Variance: By averaging predictions across models trained on various subsets, bagging reduces the variance, leading to a more stable ensemble.
- Improves Stability: Changes in the training dataset have less impact on the overall bagged model, making it more resilient to variations in the data.

Advantages of Bagging - Flexibility and Practical Benefits

- Handles High Variability: Particularly effective for high-variance algorithms like decision trees, making bagging a popular choice for such models.
- Parallelizable Computation: Each model in the ensemble can be trained independently, supporting parallel processing and efficient computational use.
- Good with Noisy and Imbalanced Data:
  - Noisy Data: The averaging process in bagging reduces the impact of noise in individual predictions.
  - Imbalanced Data: Bagging can enhance performance on imbalanced datasets by balancing model focus through multiple sampling.

Random Forest - A Popular Bagging Algorithm

- Overview: Random Forest is a powerful ensemble method based on bagging, known for its high performance and ease of use in classification and regression tasks.
- Process:
  - Bootstrap Sampling: Draw m bootstrap samples $D_1, D_2, \ldots D_m$ from the original dataset D, allowing replacement.
  - Train Decision Trees: For each sample D_j, train an independent decision tree f_j.
- Feature Subsampling: At each node split within a tree, randomly select a subset of $k \leq d$ features (where d is the total number of features) and choose the best split only from this subset. This increases diversity among trees and reduces overfitting.

Challenges with Algorithm Usability - Hyperparameters

- Hyperparameter Tuning:
  - Many algorithms have sensitive hyperparameters that impact performance.
- Examples:
  - Learning rates too low (<0.001) cause non-learning, while too high (>0.04) causes oscillation.
- Trial and Error: Certain algorithms may only converge after many attempts due to sensitive initialization.
- Optimal Settings: Algorithms with numerous hyperparameters (e.g., 7 or more) can require meticulous tuning for effective performance.

Feature and Data Requirements

- Feature Limitations:
  - Some algorithms have specific requirements for feature types and dimensionality.
  - For instance, high-dimensional data may need dimension reduction or selection.
- Data Encoding: Certain algorithms may require specialized encoding (e.g., categorical to numerical) to work effectively.

Why Random Forest is Easy to Use

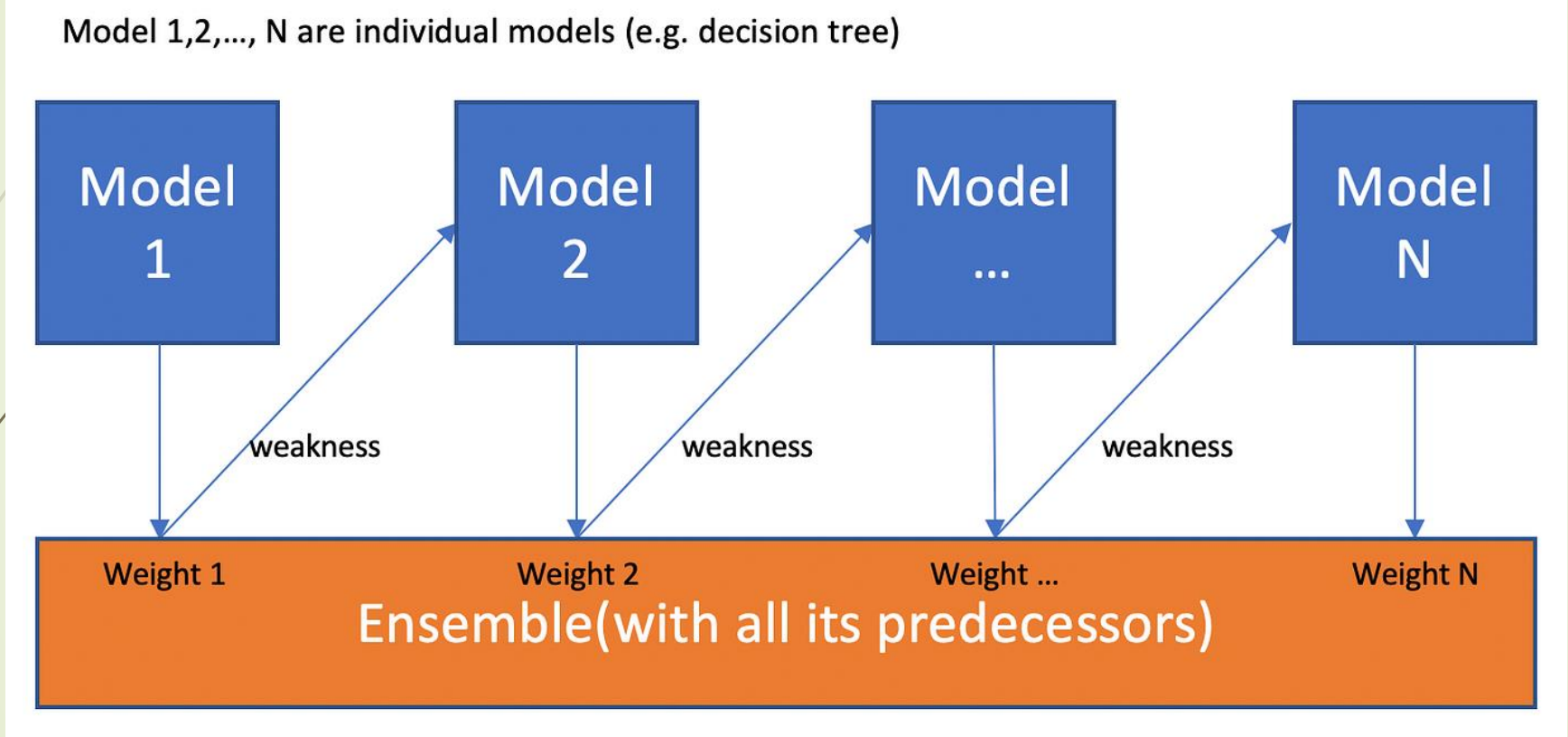
- Simple Hyperparameters:
  - Only two main hyperparameters: number of trees *m* and subset size *k* for splitting.
  - Insensitive to Hyperparameter Tuning: Performance is robust to suboptimal settings of *m* and *k*.
  - Suggested values: k=sqrt{d} (where d is the number of features), and increasing m generally improves results.
- Flexible and Practical
  - Works well with different data types (categorical, continuous) without strict pre-processing.
  - Continue training until time/budget constraints are met.

Boosting

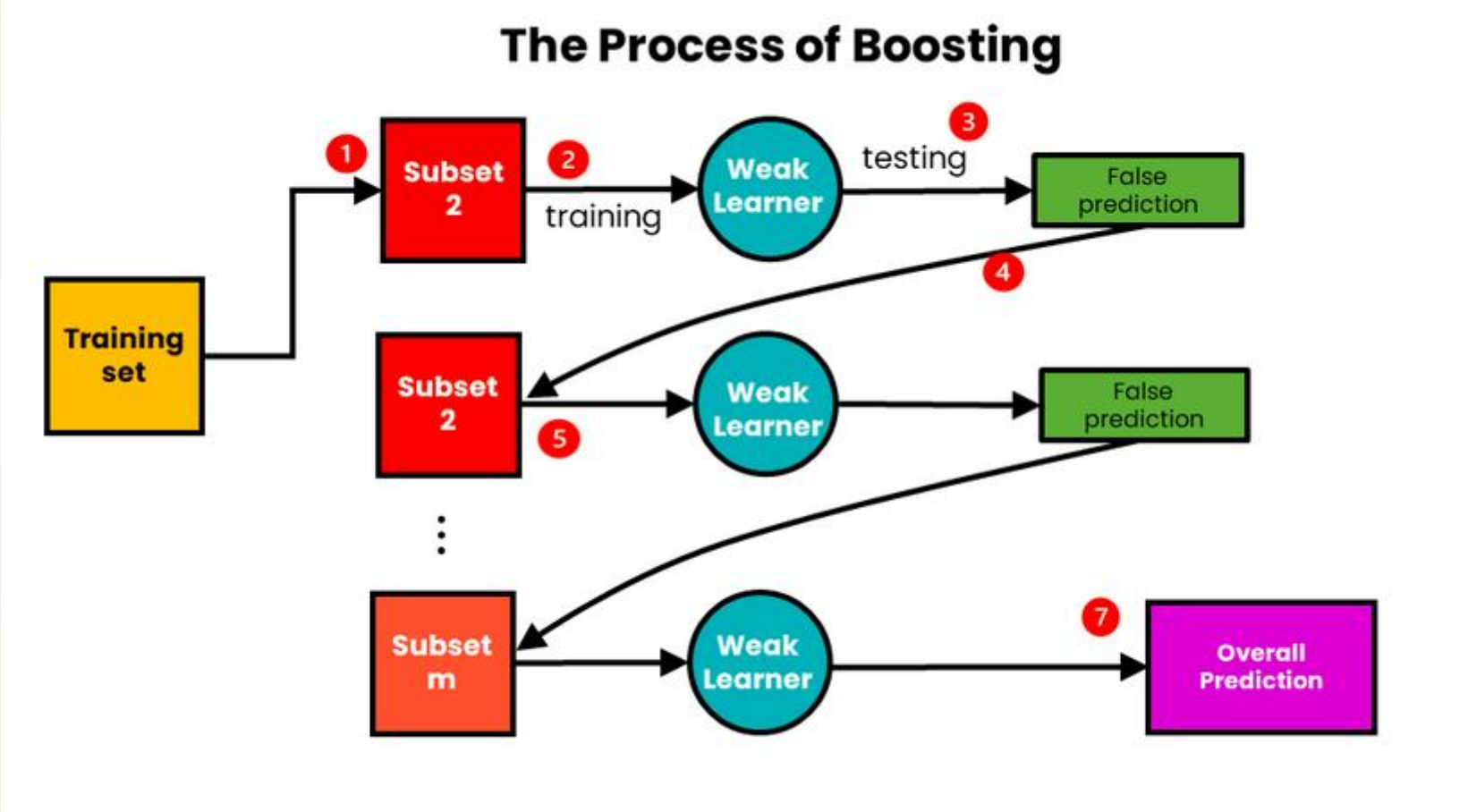
$$\mathbb{E}_{D\sim P_n,(x,y)\sim P}\left[(f_D(x)-y)^2\right] = \underbrace{\mathbb{E}_{x,D}\left[(f_D(x)-\bar{f}(x))^2\right]}_{\text{Variance}} + \underbrace{\mathbb{E}_{x,y}\left[(\bar{y}(x)-y)^2\right]}_{\text{Noise}} + \underbrace{\mathbb{E}_{x}\left[(\bar{f}(x)-\bar{y}(x))^2\right]}_{\text{Bias}}$$

where:

- **Variance**: $\mathbb{E}_{x,D}\left[(f_D(x)-\bar{f}(x))^2\right]$ – measures the variability of the model's predictions $f_D(x)$ around the average prediction $\bar{f}(x)$.
- **Noise**: $\mathbb{E}_{x,y}\left[(\bar{y}(x)-y)^2\right]$ – represents the inherent randomness in the data.
- **Bias**: $\mathbb{E}_{x}\left[(\bar{f}(x)-\bar{y}(x))^2\right]$ – measures the difference between the average model prediction $\bar{f}(x)$ and the true value $\bar{y}(x)$.


Model 1,2,…, N are individual models (e.g. decision tree)
Model 1
Model 2
Model …
Model N
weakness
weakness
weakness
Weight 1
Weight 2
Weight …
Weight N
Ensemble(with all its predecessors)

Boosting


The Process of Boosting
Training set
1
Subset 2
2
training
Weak Learner
3
testing
False prediction
4
Subset 2
5
Weak Learner
False prediction
Subset m
Weak Learner
7
Overall Prediction

Combining Weak Learners

- Weak Learner: A simple model that performs only slightly better than random guessing.
- Research Question: Michael Kearns (1988) asked if weak learners could be combined to form a strong learner with low bias.
- Answer: Robert Schapire (1990) demonstrated that it's possible, laying the foundation for boosting techniques.

Introduction to Boosting

- Definition: Boosting is an ensemble technique that converts a series of weak models into a strong classifier.
- Process: Weak learners are trained sequentially, with each model focusing on the errors made by the previous ones.

How Boosting Works

- Step-by-Step Process:
- Step 1: Train a model on the training data.
- Step 2: Build subsequent models to correct errors from the previous model.
- Repeat: Continue adding models until the error rate stabilizes or a maximum number of models is reached.

Mathematics of Boosting

- Starting Point: Begin with a set of weak learners $f_i(x)$.
- Example: Use shallow trees as weak learners.
- Ensemble Model: The final model is a weighted sum of all weak learners:

$$F(x) = \sum_{i=1}^{m} \alpha_i f_i(x)$$

- Inference: The ensemble classifier uses weighted voting for predictions.

Boosting Training Process

- Goal: Sequentially build models F1(x),F2(x),…to improve prediction accuracy.
- Gradient Descent-Like Approach: Instead of modifying parameters, each iteration adds a new weak learner to minimize the loss.
- Loss Function:

$$\mathcal{L}(F) = \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(F(x_i), y_i)$$

Optimizing Boosting

- Objective at Each Step: At iteration t, select a new function f_{t+1} to minimize the loss:

$$f_{t+1} = \arg\min_{f,\alpha} \mathcal{L}(F_t + \alpha f)$$

- Update Rule: The new classifier is:

$$F_{t+1} = F_t + \alpha f_{t+1}$$

Gradient Boosting Basics

- What is Gradient Boosting?
  - A boosting technique that builds a strong model by combining the predictions of multiple weak learners, trained on the same dataset.
  - Follows a stage-wise addition approach, iteratively improving the model by minimizing the error from previous iterations.
- How it Works
  - Step 1: The first weak learner simply outputs the mean of the target variable.
  - Step 2: Calculate residuals (errors) from the previous model's predictions.
  - Step 3: Train the next weak learner to predict these residuals.
  - Repeat: Continue adding models, each correcting the errors of the previous ones until a stopping criterion is met (like minimum residuals or max iterations).

AdaBoost Overview

- Purpose: Primarily for classification, though extended to regression.
- Weak Learners: Binary classifiers, typically decision stumps (shallow decision trees).
- Loss Function: Uses an exponential loss function:

$$\mathcal{L}(F) = \sum_{i=1}^{n} e^{-y_i F(x_i)}$$

  - n: The number of samples in the dataset.
  - y_i: The actual label for the i-th sample (usually +1 or −1 in AdaBoost).
  - $F(x_i)$: The model's prediction for the i-th sample.

- Step Size: Can compute the optimal step size in closed form; adjust weights and normalize after each iteration.

How AdaBoost Works

- Adaptive Nature: Focuses on misclassified instances by adjusting weights, helping subsequent models correct previous errors.
- Convergence: Fast exponential decrease in training loss, achieving zero training error in O(log(n)) time.
- Strong Learner Creation: Turns weak learners into a robust classifier with minimal bias and low variance.

AdaBoost with Decision Trees

- Versatile Algorithm: AdaBoost with decision trees is competitive method, transforming weak classifiers (accuracy > 50%) into a strong model.
- Practical Use: Often used until all misclassifications are minimized or a set iteration limit is reached.

Advantages of Boosting

- 1. Ease of Implementation:
  - Boosting algorithms are straightforward and interpretative, learning iteratively from their mistakes.
  - They typically don't require extensive data preprocessing and often include routines to handle missing data.
  - Most programming languages offer libraries with built-in implementations, allowing for fine-tuning.
- 2. Bias Reduction:
  - Boosting reduces high bias by combining multiple weak learners in sequence, iteratively improving model accuracy.
  - This sequential approach refines predictions, addressing initial inaccuracies in the model.
- 3. Computational Efficiency:
  - Boosting algorithms focus on features that enhance accuracy, reducing unnecessary computations.
  - They can efficiently handle large datasets by prioritizing relevant attributes during training.

References

- https://www.cs.ucf.edu/~lboloni/Teaching/CAP4611_Fall2023/
- https://www.datacamp.com/tutorial/what-bagging-in-machine-learning-a-guide-with-examples
- https://www.geeksforgeeks.org/boosting-in-machine-learning-boosting-and-adaboost/
- https://www.analyticsvidhya.com/blog/2023/01/ensemble-learning-methods-bagging-boosting-and-stacking/