Plan of Attack

13 April 2024 20:13

Introduction

13 April 2024 08:29

- LightGBM stands for Light Gradient Boosting Machine
- It is an open source Gradient Boosting Framework
- Developed by Microsoft DMTK in 2016
- Goal was to develop high performance Gradient Boosting framework optimized for speed.
- Core Features
    - Speed and memory efficient
    - Accuracy
    - Works with categorical data
    - Multiple Language support
    - Supports Distributed Computing
    - Supports GPU
    - Can handle missing values and sparse data
    - Can work with custom loss functions

1. Boosting and Objective Function

13 April 2024 13:58

2. Histogram based Split Finding

13 April 2024 13:58

3. Best-fit Tree (Leaf-wise growth strategy)

13 April 2024 13:36

Characteristics of Leaf-first tree

1. Unbalanced Trees
2. Can find more accurate trees
3. Computational savings - by focusing on the most promising split, it often requires fewer splits than other strategies to reach same accuracy
4. Can lead to overfitting

4. GOSS (Gradient based One Side Sampling)

13 April 2024 13:38

GOSS is an innovative data sampling technique that addresses a common problem in the training of gradient boosting models: the balancing act between speeding up training by using a subset of the data (data sampling), and maintaining the accuracy of the learned model.

x1	x2	y	|-g|
-0.4	-1.3	0	1.3
1.8	1.7	1	0.2
0.8	0.3	1	2.8
-1.2	-0.7	0	1.0
1.5	2.8	0	1.5
-0.4	0.3	1	0.8

x1	x2	y	|-g|
0.8	0.3	1	2.8
1.5	2.8	0	1.5
-0.4	-1.3	0	1.3
-1.2	-0.7	0	1.0
-0.4	0.3	1	0.8
1.8	1.7	1	0.2

5. EFB (Exclusive Feature Bundling)

13 April 2024 14:30

Hyperparameters

17 April 2024 18:08

1. Boosting

- Description: Specifies the type of boosting framework to use.
- Possible Values: "gbdt", "rf", "dart", "goss".
- Guideline: Select "gbdt" for gradient boosting decision tree, "rf" for random forest, "dart" for Dropouts meet Multiple Additive Regression Trees, and "goss" for Gradient-based One-Side Sampling.

2. Objective

- Description: Specifies the objective function, or the loss to be minimized.
- Possible Values: "regression", "regression_l1", "binary", "multiclass", "cross_entropy", etc.
- Guideline: Match the objective to your task type—classification, regression, etc.

3. Metric

- Description: Metric(s) to be used for validation.
- Possible Values: "mae", "mse", "binary_logloss", "binary_error", "multi_error", "multi_logloss", "auc", etc.
- Guideline: Choose a metric suitable for evaluating model performance on your specific objective.

4. max_depth

- Description: Maximum depth of a tree.
- Possible Values: Any integer greater than 0, or -1 for no limit.
- Guideline: A deeper tree can model more complex relationships but may lead to overfitting.

5. learning_rate

- Description: Shrinkage rate.
- Possible Values: Any positive float.
- Guideline: Typical values range from 0.01 to 0.2. Lower rates require more trees but can yield better models.

6. num_leaves

- Description: Maximum number of leaves in one tree.
- Possible Values: Any integer greater than 1.
- Guideline: More leaves will make the model more complex and more likely to overfit.

7. feature_fraction

- Description: Fraction of features to be used in each iteration.
- Possible Values: Float in (0, 1].
- Guideline: Less than 1.0 can be used for speeding up training and dealing with overfitting.

8. min_data_in_leaf

- Description: Minimum number of data points in one leaf.
- Possible Values: Any positive integer.
- Guideline: Can be used to deal with overfitting.

9. min_gain_to_split

- Description: The minimum gain to make a further partition on a leaf node.
- Possible Values: Any non-negative float.
- Guideline: Adjust to control the tree growing.

10. device_type

- Description: Device to use for training.
- Possible Values: "cpu", "gpu".
- Guideline: Use "gpu" for faster training if a compatible GPU is available.

11. max_bin

- Description: Maximum number of bins that feature values will be bucketed in.
- Possible Values: Any positive integer.
- Guideline: Increasing this value can improve accuracy but also lead to overfitting and increased memory usage.

12. Verbose

- Description: Controls the verbosity of the output.
- Possible Values: Any integer (higher values lead to more verbose output).

- Guideline: Set to a higher number to monitor training progress more closely or set to 0 for silent mode.

13. num_iterations

- Description: Number of boosting iterations.
- Possible Values: Any positive integer.
- Guideline: More iterations can lead to a better performance but risk overfitting.

14. top_rate

- Description: The fraction of large gradient data instances to be kept during training.
- Possible Values: Float in (0,1).
- Guideline: Used when boosting_type is "goss"; adjust to tweak the balance between retaining informative cases and random sampling.

15. other_rate

- Description: The fraction of small gradient data instances to be randomly sampled for training.
- Possible Values: Float in (0,1).
- Guideline: Used when boosting_type is "goss"; helps in ensuring diversity of the data that the model learns from.

16. min_sum_hessian_in_leaf

- Description: Minimum sum of Hessian (second derivative of the objective function) required in a leaf.
- Possible Values: Any positive float.
- Guideline: Increase to deal with overfitting.

17. lambda_l1 (reg_alpha)

- Description: L1 regularization term on weights, equivalent to Lasso regression.
- Possible Values: Any non-negative float.
- Guideline: Increases model regularization to prevent overfitting; higher values make the model more conservative.

18. lambda_l2 (reg_lambda)

- Description: L2 regularization term on weights, equivalent to Ridge regression.

- Possible Values: Any non-negative float.
- Guideline: Also increases model regularization; generally, L2 is more effective in preventing overfitting than L1.

19. early_stopping_round

- Description: Activates early stopping. The training process will stop if the validation metric does not improve for a specified number of consecutive rounds.
- Possible Values: Any positive integer.
- Guideline: Use this parameter to prevent overfitting. It is particularly useful when you have a separate validation dataset. A common setting is anywhere from 10 to 100 rounds, depending on the size and variability of your dataset. This parameter must be used with at least one validation set specified.

20. num_threads

- Description: Specifies the number of threads to use for LightGBM.
- Possible Values: Any positive integer, or -1 to use all available threads.
- Guideline: Setting this parameter allows LightGBM to utilize multicore processing, which can significantly speed up computation. The default setting, -1, automatically uses all cores available, but it can be set to a specific number if there is a need to limit resource use.

21. tree_learner

- Description: Specifies the type of tree learner to use for parallel learning.
- Possible Values: "serial", "feature", "data", "voting"
- Guideline:
- Use "serial" for no parallel learning.
- "feature" for parallel learning by splitting on features which is useful when you have a large number of features.
- "data" for parallel learning by distributing data across machines which is beneficial when the dataset itself is very large.
- "voting" is used for a parallel learning method that involves all machines to communicate with all other machines by sending their local histograms

22. num_machines

- Description: Specifies the number of machines to use for distributed learning.
- Possible Values: Any positive integer.

- Guideline: This should be set to the number of machines you intend to use; it's essential for managing data and workload distribution when LightGBM is run in a distributed environment.

23. machine_list_file

- Description: Path to the file containing the list of machines (IP/port) used for distributed learning.
- Possible Values: String (path to file).
- Guideline: Necessary for setting up distributed learning when using the "data" or "voting" tree learners. The file needs to contain the IP address and port of each machine involved in the training process.

24. local_listen_port

- Description: TCP port that LightGBM will listen on for connections from other instances in a distributed learning environment.
- Possible Values: Any valid port number.
- Guideline: This should be set if there are firewall or port routing issues in your network setup; it ensures that each node can communicate effectively.

25. Timeout

- Description: The socket's timeout in milliseconds.
- Possible Values: Any non-negative integer.
- Guideline: Increase this if your datasets are very large and network delays are expected, to prevent premature disconnection or timeouts during distributed training.

26. network_interface

- Description: Network interface for distributed learning.
- Possible Values: String (name of the network interface).
- Guideline: This can be set to bind communication to a specific network interface, which is particularly useful in complex network environments where machines have multiple network interfaces.

27. is_unbalance

- Description: Used to handle the training data's imbalance when the training data has binary labels (0 or 1).
- Possible Values: true or false
- Guideline: Set to true if the training data is imbalanced and the objective is binary classification. This makes the model pay more attention to the minority class.

28. scale_pos_weight

- Description: Used to scale the gradient for the positive class, primarily in binary classification.
- Possible Values: Any positive float.
- Guideline: Useful for tweaking the model's focus on the positive class in an imbalanced dataset. Typically, this is set to the ratio of the number of negative class samples to the number of positive class samples.