Model Tuning

Outline

- Model Tuning
- Data Science Project Life Cycle
- Case Study

Model Parameters

Model parameter is a configuration variable that is internal to the model whose value can be estimated from the underlying data

- These are required by the model when making predictions
- These values define the skill of the model on your problem
- These are estimated or learned from the underlying data
- These are not set manually by the data scientist
- They are often saved as part of the learned model

- Parameters are key to machine learning algorithms. They are part of the model that is learned from historical training data
- Examples:
    - The coefficients in a linear or logistic regression

Model Hyperparameters

- Model parameters are learnt during training(eg: intercept and slopes in Linear Regression)
- Model hyperparameters are set by data scientist ahead of training
- Hyperparameters are parameters that are not directly learnt within estimators
  - Examples for hyperparameters?
- In scikit-learn these are passed as arguments to the constructor of the estimator classes
- Intercept and slopes learnt during training of a Linear Regression model are model parameters while the number of trees in a Random Forest is a model hyperparameter because it is set by the data scientist
- It is possible and recommended to search the hyperparameter space for the best cross validation score.

Model Hyperparameters

- A model hyperparameter is a configuration that is external to the model and whose value cannot be estimated from data
  - They are often used in processes to help estimate model parameters
  - They are often specified by data scientist
  - They can often be set using heuristics
  - They are often tuned for a given predictive modelling problem
- We cannot know the best value for a model hyperparameter on a given problem. We may use rule of thumb, copy values used in other problems, or search for the best value by trial and error
- When a machine learning algorithm is tuned for a specific problem, then you are tuning the hyperparameters of the model that result in the most skilful predictions

Model Hyperparameters

- Hyperparameters can be thought of as model settings
- These settings need to be tuned for each problem as the best model hyperparameters for one particular data set will not be the best across all data sets
- The process of hyperparameter tuning (also referred as hyperparameter optimization) means finding the combination of hyperparameter values for a machine learning model that performs the best – as measured on a test data set – for a given problem
- A lot of data scientists use the terms ‘parameters’ and ‘hyperparameters’ interchangeably to refer to the model settings. This is technically incorrect.
- If you have to specify a model parameter manually then it is probably a model hyperparameter

Hyper parameters: Examples

Linear Regression, Logistic Regression, Naïve Bayes

- No hyper parameters

Decision Trees

- min_samples_leaf
- min_samples_split
- criterion

AdaBoost

- n_estimators
- learning_rate

kNearestNeighbors

- Number of neighbors

Random Forest

- n_estimators
- max_depth
- min_samples_split
- min_samples_leaf
- max_features
- etc

Gradient Boosting

- n_estimators
- max_depth
- learning_rate
- max_features
- etc

Optimal Parameters Search

- It is common that a small subset of hyper-parameters can have a large impact on the predictive or computation of the model and other hyper-parameters can be set to default

Optimal Parameters Search

- Optimal Parameters search consists of:
  - An estimator (eg: regressor or classifier)
  - A parameter space
  - A method for searching or sampling candidates
  - A cross-validation scheme
  - A score function

Optimal Hyper Parameters Search

- Some models allow for specialized, efficient parameter search strategies
- Approaches for optimal hyper parameters search
  - Manual
  - Grid search
    - Exhaustively considers all parameter combinations
  - Random search (not covered in class)
  - Automated hyper parameter tuning (not covered in class)

Grid Search

- Grid search is provided by *GridSearchCV*
- Generates candidates from a grid of parameter values specified with the *param_grid* parameter
- Example *param_grid* for RandomForest:
  - grid = { "n_estimators" : [100,200,500],
    "criterion" : ["gini", "entropy"],
    "max_features" : ['sqrt','log2',0.2,0.5,0.8],
    "max_depth" : [3,4,6,10],
    "min_samples_split" : [2, 5, 20,50] }
- The GridSearchCV instance implements the usual estimator API: when 'fitting' it on a dataset all the possible combinations of parameter values are evaluated and the best combination is retained

Grid Search

- Lab (“Lab1_gridSearch_MLmodels.ipynb “)

Tips for parameter search

- Specifying an objective metric
- Composite estimators and parameter spaces
- Model selection: development and evaluation
- Parallelism

Pipeline

- Pipeline can be used to chain multiple estimators into one.
- This is useful as there is often a fixed sequence of steps in processing the data, for example feature selection, normalization and classification.
- Pipeline serves two purposes here:
  - Convenience and encapsulation: We only have to call fit and predict once on our data to fit a whole sequence of estimators.
  - Joint parameter selection: We can grid search over parameters of all estimators in the pipeline at once.

- Safety: Pipelines help avoid leaking statistics from our test data into the trained model in cross-validation, by ensuring that the same samples are used to train the transformers and predictors.

Pipeline : usage

- The Pipeline is built using a list of (key, value) pairs, where the key is a string containing the name we want to give this step and value is an estimator object

Data Science Project Life Cycle

1. Prepare Problem
   1. Load libraries
   2. Load dataset
2. Summarize Data
   1. Descriptive statistics
   2. Data visualizations
3. Prepare Data
   1. Data Cleaning
   2. Feature Selection
   3. Data Transforms
4. Evaluate Algorithms
   1. Split-out validation dataset
   2. Test options and evaluation metric
   3. Spot Check Algorithms
   4. Compare Algorithms
5. Improve Accuracy
   1. Algorithm Tuning
   2. Ensembles
6. Finalize Model
   1. Predictions on validation dataset
   2. Create standalone model on entire training dataset
   3. Save model for later use

Data Science Project Life Cycle

- Lab (“Lab2_mlProjectLifeCycleBostonHousePricePrediction.ipynb “)
- Lab (“Lab2_Extension_customerDeliveryBostonHousePricePrediction.ipynb “)
- Hands-On (“ Lab3_mlProjectLifeCycleKingCountyHousePricePrediction.ipynb “)

Possible Capstone Projects

- NLP
  - Text classification
  - Language modelling
  - Speech recognition
  - Caption generation
  - Machine translation
  - Document summarization
- Image Analytics
  - Image detection
  - Brightness correction in Zoom video call(help speaker to deliver talks with ease)
  - Weed identification in crops using deep learning techniques
  - Store monitoring (brands detection etc) and supply chain optimization using deep learning
  - Psychology analysis using hand written notes

- Consumer complaints data analysis and case outcome prediction
  - Collect case status data from district, state and NCDRC
  - Extract important basic info about each case (using NLP techniques) and add to the features
  - Build predictive models for the outcome of the complaints
- Consumer complaints judgments summarization
  - Real estate
  - Healthcare
- Similar consumer complaints judgements identification
- NLP based features engineering for consumer complaints judgements(may need to use sequential models also)

References

- Ref1: https://www.analyticsvidhya.com/blog/2016/02/complete-guide-parameter-tuning-gradient-boosting-gbm-python/

Summary

- Grid Search
- Pipeline
- Data Science Project Life Cycle
- Case study

Questions?