# XGBoost

# Part 4: Features and hyperparameters of XGBoost

By: Noureddin Sadawi, PhD

University of London

# Some features of XGBoost

- **Regularisation:** XGBoost uses both L1 and L2 regularisation to penalise complex models (this helps in preventing overfitting).
- **Handling sparse data:** this is data with a large percentage of non actual values (e.g. zeros, missing values etc). XGBoost has an internal algorithm that splits the data while being aware of sparsity.
- **Block structure for parallel learning:** XGBoost is developed to use multiple cores on the CPU and this helps in faster computing.

# Some features of XGBoost

- **Out-of-core computing:** In case of huge data that does not fit into memory, XGBoost exploits the available disk space.
- **Weighted quantile sketch:** This is a useful feature when the data is weighted (i.e. instances have weights such as when data is class-imbalanced).
- **Cache awareness:** XGBoost has been developed for optimised hardware usage (i.e. cache memory) to speed up procedures such as gradient descent.

# Model hyperparameters

- Hyperparameters are not model parameters.
- Model parameters are learnt from data during loss function optimisation (i.e. during model training).
- Model parameters specify how map input data into the desired output (e.g. regression coefficients).
- Hyperparameters define how our model is structured (i.e. they are outside the model, not part of it).
- Examples:
  - number of iterations to perform in boosting
  - depth of a decision tree
  - weak learner in boosting.

# XGBoost hyperparameters

Three types of parameters: general parameters, booster parameters and task parameters:

- '**General parameters** relate to which booster we are using to do boosting, commonly tree or linear model
- **Booster parameters** depend on which booster you have chosen
- **Learning task parameters** decide on the learning scenario. For example, regression tasks may use different parameters with ranking tasks.’

Source: https://xgboost.readthedocs.io/en/latest/parameter.html

# Example general parameters

‘booster [default= gbtree ]

- Which booster to use. Can be gbtree, gblinear or dart; gbtree and dart use tree based models while gblinear uses linear functions…

nthread [default to maximum number of threads available if not set]

- Number of parallel threads used to run XGBoost. When choosing it, please keep thread contention and hyperthreading in mind.’

Source: https://xgboost.readthedocs.io/en/latest/parameter.html

# Example booster parameters

‘eta [default=0.3, alias: learning_rate]

- Step size shrinkage used in update to prevents overfitting. After each boosting step, we can directly get the weights of new features, and eta shrinks the feature weights to make the boosting process more conservative. range: [0,1]…

max_depth [default=6]

- Maximum depth of a tree. Increasing this value will make the model more complex and more likely to overfit…

scale_pos_weight [default=1]

- Control the balance of positive and negative weights, useful for unbalanced classes.’

Source: https://xgboost.readthedocs.io/en/latest/parameter.html

# Example learning task parameters

‘objective [default=reg:squarederror].’ Examples:

- reg:squarederror: regression with squared loss…
- binary:logistic: logistic regression for binary classification, output probability…
- multi:softmax: set XGBoost to do multiclass classification using the softmax objective…

eval_metric [default according to objective]

- Evaluation metrics for validation data…

seed [default=0]

- Random number seed.’

Source: https://xgboost.readthedocs.io/en/latest/parameter.html

# Hyperparameter tuning

- XGBoost has a large number of hyperparameters and each one of them can have multiple values.
- What is the best combination of hyperparameter values?
- This question is answered through hyperparameter tuning, which is the process of searching for the ideal set of hyperparameter values that result in the best mode.
- This is an intensive process that can take time.
- There is no one optimal set of hyperparameter values that works for all problems.

# Hyperparameter Tuning Methods

**Exhaustive Grid Search:**

- Create a grid of hyperparameters and their possible values
- Exhaustively generate candidates from the grid of parameter values and evaluate the model using them

**Randomised Parameter Optimisation:**

- This method uses randomised search to select candidate sets of parameter values
- Does not go through all possible combinations

More here: https://scikit-learn.org/stable/modules/grid_search.html