10.06 Pipelines & GridSearch in Spark

Spark Pipelines

- Pipelines are a simple and effective way to manage complex machine learning workflows
- Inspired by the popular implementation in scikit-learn, the concept of Pipelines is to facilitate the creation, tuning, and inspection of practical ML workflows.
- A Spark Pipeline is specified as a sequence of stages, and each stage is either a Transformer or an Estimator. These stages are run in order, and the input DataFrame is transformed as it passes through each stage.

Vector Assembler

- Vector Assembler is a transformer that assembles all the features into one vector from multiple columns that contain type double.

- StringIndexer to be used if any of our columns contains string values to convert it into numeric values.

Spark Pipelines – Components

- Transformers
  - A Transformer is an abstraction that includes feature transformers and learned models. Technically, a Transformer implements a method transform(), which converts one DataFrame into another, generally by appending one or more columns. For example:
    - A feature transformer might take a DataFrame, read a column (e.g., text), map it into a new column (e.g., feature vectors), and output a new DataFrame with the mapped column appended.
    - A learning model might take a DataFrame, read the column containing feature vectors, predict the label for each feature vector, and output a new DataFrame with predicted labels appended as a column.

Spark Pipelines – Components

- Transformers
  - A Transformer is an abstraction that includes feature transformers and learned models. Technically, a Transformer implements a method transform(), which converts one DataFrame into another, generally by appending one or more columns. For example:
    - A feature transformer might take a DataFrame, read a column (e.g., text), map it into a new column (e.g., feature vectors), and output a new DataFrame with the mapped column appended.
    - A learning model might take a DataFrame, read the column containing feature vectors, predict the label for each feature vector, and output a new DataFrame with predicted labels appended as a column.

Spark Pipelines – Components

- Estimators
  - An Estimator abstracts the concept of a learning algorithm or any algorithm that fits or trains on data. Technically, an Estimator implements a method fit(), which accepts a DataFrame and produces a Model, which is a Transformer.
  - For example, a learning algorithm such as LogisticRegression is an Estimator, and calling fit() trains a LogisticRegressionModel, which is a Model and hence a Transformer.

Spark Pipelines

- In machine learning, it is common to run a sequence of algorithms to process and learn from data.

- E.g., a simple text document processing workflow might include several stages:
  - Split each document’s text into words.
  - Convert each document’s words into a numerical feature vector.
  - Learn a prediction model using the feature vectors and labels.
  - MLlib represents such a workflow as a Pipeline, which consists of a sequence of PipelineStages (Transformers and Estimators) to be run in a specific order. We will use this simple workflow as a running example in this section.

Spark Hyperparameter Tuning

- An important task in ML is model selection, or using data to find the best model or parameters for a given task. This is also called tuning.

- Tuning may be done for individual Estimators such as LogisticRegression, or for entire Pipelines which include multiple algorithms, featurization, and other steps.

- Users can tune an entire Pipeline at once, rather than tuning each element in the Pipeline separately.

Spark Hyperparameter Tuning

- MLlib supports model selection using tools such as CrossValidator and TrainValidationSplit. These tools require the following items:

  - Estimator: algorithm or Pipeline to tune
  - Set of ParamMaps: parameters to choose from, sometimes called a “parameter grid” to search over
  - Evaluator: metric to measure how well a fitted Model does on held-out test data

Spark Hyperparameter Tuning

- At a high level, these model selection tools work as follows:
  - They split the input data into separate training and test datasets.
  - For each (training, test) pair, they iterate through the set of ParamMaps:
  - For each ParamMap, they fit the Estimator using those parameters, get the fitted Model, and evaluate the Model's performance using the Evaluator.
  - They select the Model produced by the best-performing set of parameters.

Spark Hyperparameter Tuning

- The Evaluator can be:
  - RegressionEvaluator for regression problems
  - BinaryClassificationEvaluator for binary data
  - MulticlassClassificationEvaluator for multiclass problems
  - MultilabelClassificationEvaluator for multi-label classifications
  - RankingEvaluator for ranking problems

Spark Hyperparameter Tuning

- Cross-Validation
  - CrossValidator begins by splitting the dataset into a set of folds which are used as separate training and test datasets. E.g., with k=3 folds, CrossValidator will generate 3 (training, test) dataset pairs, each of which uses 2/3 of the data for training and 1/3 for testing. To evaluate a particular ParamMap, CrossValidator computes the average evaluation metric for the 3 Models produced by fitting the Estimator on the 3 different (training, test) dataset pairs.
  - After identifying the best ParamMap, CrossValidator finally re-fits the Estimator using the best ParamMap and the entire dataset.

Spark Hyperparameter Tuning

- Train-Validation Split
  - In addition to CrossValidator Spark also offers TrainValidationSplit for hyper-parameter tuning. TrainValidationSplit only evaluates each combination of parameters once, as opposed to k times in the case of CrossValidator. It is, therefore, less expensive, but will not produce as reliable results when the training dataset is not sufficiently large.

  - Unlike CrossValidator, TrainValidationSplit creates a single (training, test) dataset pair. It splits the dataset into these two parts using the trainRatio parameter. For example with trainRatio=0.75, TrainValidationSplit will generate a training and test dataset pair where 75% of the data is used for training and 25% for validation.

  - Like CrossValidator, TrainValidationSplit finally fits the Estimator using the best ParamMap and the entire dataset.