



APACHE SPARK MACHINE LEARNING

Bin Jiang
11/26/2016

What is Spark MLlib

- **Spark MLlib** is a module (a library / an extension) of Apache Spark to provide distributed machine learning algorithms on top of Spark's RDD abstraction. Its goal is to simplify the development and usage of large scale machine learning

What is Spark MLlib

- Common learning algorithms and utilities:
 - Classification
 - Regression
 - Frequent itemsets (via FP-growth Algorithm)
 - Feature extraction and selection
 - Clustering
 - Statistics
 - Linear Algebra
 - Recommendation (Collaborative filtering)
 - Dimensionality reduction
 - Model import and export
 - Lower-level optimization primitives and higher-level pipeline APIs

What is Spark MLlib

- There are two libraries for Machine Learning in Spark MLlib: `org.apache.spark.mllib` for RDD-based Machine Learning and a higher-level API under `org.apache.spark.ml` for DataFrame-based Machine Learning with Pipelines.

Spark MLlib vs Spark ML

- (Old-fashioned) RDD-based API vs (the latest and greatest) DataFrame-based API
- Using spark.ml is recommended because with DataFrames the API is more versatile, flexible and high-level

Why is Machine Learning

Machine Learning uses large datasets to identify (infer) patterns and make decisions (aka *predictions*). Automated decision making is what makes Machine Learning so appealing. You can teach a system from a dataset and let the system act by itself to predict future

Why is Spark MLlib

- The amount of data (measured in TB or PB) is what makes Spark MLlib especially important since a human could not possibly extract much value from the dataset in a short time
- Spark handles data distribution and makes the huge data available by means of RDDs, DataFrames, and recently Datasets

Machine Learning Use Cases

Use cases for Machine Learning (and hence Spark MLlib that comes with appropriate algorithms):

- Security monitoring and fraud detection
- Operational optimizations
- Product recommendations or (more broadly) Marketing optimization+
- Ad serving and optimization

Machine Learning Concepts

- Observation

An observation is used to learn about or evaluate (i.e. draw conclusions about) the observed item's target value.

Spark models observations as rows in a DataFrame.

Feature

Machine Learning Concepts

- Feature
 - A feature (aka dimension or variable) is an attribute of an observation. It is an independent variable
 - Spark models features as columns in a DataFrame (one per feature or a set of features)

Machine Learning Concepts

- Feature
 - A feature (aka dimension or variable) is an attribute of an observation. It is an independent variable
 - Spark models features as columns in a DataFrame (one per feature or a set of features)
 - **Note** - Ultimately, it is up to an algorithm to expect one or many features per column.

Machine Learning Concepts

- There are two classes of features:
 - Categorical with discrete values, i.e. the set of possible values is limited, and can range from one to many thousands. There is no ordering implied, and so the values are incomparable
 - Numerical with quantitative values, i.e. any numerical values that you can compare to each other. You can further classify them into discrete and continuous features

Machine Learning Concepts

- Label
 - A label is a variable that a machine learning system learns to predict that are assigned to observations
 - There are categorical and numerical labels
 - A label is a dependent variable that depends on other dependent or independent variables like features

Data Types

- Local vector
 - Dense
 - sparse
- Labeled point
- Local matrix
- Distributed matrix
 - RowMatrix
 - IndexedRowMatrix
 - CoordinateMatrix
 - BlockMatrix

Data Types – Local Vector

- A local vector has integer-typed and 0-based indices and double-typed values, stored on a single machine
- Two types of local vectors: dense and sparse
 - Dense vector is backed by a double array representing its entry values
 - Sparse vector is backed by two parallel arrays: indices and values

Data Types – Labeled point

- A labeled point is a local vector associated with a label, either dense or sparse
- Used in supervised learning algorithms
- Use a double to store a label
- Both regression and classification
- Label is either 0 (negative) or 1 (positive) for binary classification
- labels is class indices starting from zero: 0, 1, ... for multiclass classification

Data Types – Labeled point

- MLlib supports reading training datasets stored in LIBSVM format
- Used by LIBSVM and LIBLINEAR
- Text format in which each line represents a labeled sparse feature vector
 - label index1:value1 index2:value2 ...

Data Types – Local Matrix

- Integer-typed row and column indices and double-typed values
- Dense matrices and sparse matrices
 - Dense matrices is stored in a single double array in column-major order
 - Sparse matrices is stored in the Compressed Sparse Column (CSC) format in column-major order

Data Types – Distributed matrix

- Stored distributively in one or more RDDs
- Converting a distributed matrix to a different format may require a global shuffle
- Three types of distributed matrices
 - RowMatrix
 - IndexedRowMatrix
 - CoordinateMatrix

Data Types – Row Matrix

- Row-oriented distributed matrix without meaningful row indices
- Each row is a local vector
- The number of columns is not huge for a RowMatrix
- Single local vector can be reasonably communicated to the driver

Data Types – Indexed Row Matrix

- Similar to a RowMatrix but with meaningful row indices
- Each row is represented by its index (long-typed) and a local vector

Data Types – Coordinate Matrix

- Distributed matrix backed by an RDD of its entries
- Each entry is a tuple of (i: Long, j: Long, value: Double), where i is the row index, j is the column index, and value is the entry value
- Should be used only when both dimensions of the matrix are huge and the matrix is very sparse

Feature Extraction and Transformation

- Word2Vec
 - Computes distributed vector representation of words
 - Similar words are close in the vector space
 - Skip-gram model

Feature Extraction and Transformation

- StandardScaler

- Common pre-processing step
- Scaling to unit variance and/or removing the mean using column summary statistics
- Improve the convergence rate during the optimization process
- Prevents against features with very large variances
- Two parameters in the constructor (withMean and withStd)

Feature Extraction and Transformation

- Normalizer
 - Common operation for text classification or clustering
 - Scales individual samples to have unit L_p norm

Classification and Regression

- SVM
 - Standard method for large-scale classification tasks
 - Linear method with the loss function
 - Linear SVMs are trained with an L2 regularization
 - Support alternative L1 regularization
 - Outputs an SVM model

Classification and Regression

- Logistic regression
 - Used to predict a binary response
 - Linear method with the loss function
 - For binary classification problems, the algorithm outputs a binary logistic regression model
 - Binary logistic regression can be generalized into multinomial logistic regression to train and predict multiclass classification problems
 - For multiclass classification problems, the algorithm will output a multinomial logistic regression model

Classification and Regression

- Linear least squares regression
 - The most common formulation for regression problems
 - Linear method with the loss function
 - Uses no regularization
 - The average loss or training error is known as the mean squared error

Classification and Regression

- Linear lasso regression
 - The most common formulation for regression problems
 - Linear method with the loss function
 - Uses L1 regularization
 - The average loss or training error is known as the mean squared error

Classification and Regression

- Linear ridge regression
 - The most common formulation for regression problems
 - Linear method with the loss function
 - Uses L2 regularization
 - The average loss or training error is known as the mean squared error

Classification and Regression

- Decision trees
 - Popular methods for the machine learning tasks of classification and regression
 - Easy to interpret, handle categorical features, extend to the multiclass classification setting
 - Do not require feature scaling
 - Capture non-linearities and feature interactions
 - Random forests and boosting are among the top performers for classification and regression tasks

Classification and Regression

- Decision trees
 - Problem specification parameters
 - **algo**: Classification or Regression
 - **numClasses**: Number of classes (for Classification only)
 - **categoricalFeaturesInfo**: Specifies which features are categorical and how many categorical values each of those features can take

Classification and Regression

- Decision trees

- Tunable parameters

- **maxBins**: Number of bins used when discretizing continuous features
 - **maxMemoryInMB**: Amount of memory to be used for collecting sufficient statistics
 - **subsamplingRate**: Fraction of the training data used for learning the decision tree
 - **impurity**: Impurity measure (discussed above) used to choose between candidate splits

Classification and Regression

- Random forests
 - Ensembles of decision trees
 - Train multiple trees in parallel
 - Less prone to overfitting
 - Be easier to tune since performance improves monotonically with the number of trees
 - One of the most successful machine learning models for classification and regression
 - Combine many decision trees in order to reduce the risk of overfitting

Classification and Regression

- Gradient-boosted trees
 - GBTs train one tree at a time, so they can take longer to train than random forests
 - Training more trees with GBTs increases the likelihood of overfitting
 - Ensembles of decision trees
 - GBTs handle categorical features, extend to the multiclass classification setting
 - Capture non-linearities and feature interactions without feature scaling

Classification and Regression

- Naive Bayes

- Simple multiclass classification algorithm with the assumption of independence between every pair of features
- Computes the conditional probability distribution of each feature given label
- Bayes' theorem to compute the conditional probability distribution of label given an observation and use it for prediction

Collaborative Filtering

- Commonly used for recommender systems
- Fill in the missing entries of a user-item association matrix
- Model-based collaborative filtering is supported in Spark
- ALS algorithm is used to learn the latent factors which describe users and products
- Matrix factorization based collaborative filtering treats the entries in the user-item matrix as *explicit* preferences
- Many real-world use cases to only have access to *implicit feedback*

Clustering

- K-means
 - One of the most commonly used clustering algorithms that clusters the data points into a predefined number of clusters

Clustering

- Power iteration clustering (PIC)
 - Scalable and efficient algorithm for clustering vertices of a graph given pairwise similarities as edge properties

Clustering

- Latent Dirichlet allocation (LDA)
 - Topic model which infers topics from a collection of text documents
 - LDA uses a function based on a statistical model of how text documents are generated
 - LDA supports different inference algorithms via `setOptimizer` function

Dimensionality Reduction

- Singular value decomposition (SVD)
 - Factorizes a matrix into three matrices: UU , Σ , and VV
 - UU is an orthonormal matrix, whose columns are called left singular vectors
 - Σ is a diagonal matrix with non-negative diagonals in descending order, whose diagonals are called singular values
 - VV is an orthonormal matrix, whose columns are called right singular vectors
 - large matrices, usually we don't need the complete factorization but only the top singular values and its associated singular vectors

Frequent Pattern Mining

- FP-growth
 - “FP” stands for frequent pattern
 - First step of FP-growth is to calculate item frequencies and identify frequent items
 - Second step of FP-growth uses a suffix tree (FP-tree) structure to encode transactions without generating candidate sets explicitly
 - Then frequent itemsets can be extracted from the FP-tree

Frequent Pattern Mining

- Association Rules
 - Implements a parallel rule generation algorithm for constructing rules that have a single item as the consequent

Evaluation Metrics

- Binary classification model evaluation
 - Binary classifiers are used to separate the elements of a given dataset into one of two possible groups (e.g. fraud or not fraud) and is a special case of multiclass classification. Most binary classification metrics can be generalized to multiclass classification metrics
 - In the binary case, the model may output a probability for each class
 - Available metrics
 - ❖ Precision (Positive Predictive Value)
 - ❖ Recall (True Positive Rate)
 - ❖ F-measure
 - ❖ Receiver Operating Characteristic (ROC)
 - ❖ Area Under ROC Curve
 - ❖ Area Under Precision-Recall Curve

PMML model export

- KMeansModel
- LinearRegressionModel
- RidgeRegressionModel
- LassoModel
- SVMModel
- Binary LogisticRegressionModel

Machine Learning Application

- Machine learning, predictive analytics, and related data science topics are used for solving real-world problems across varied business domains
- Driving mission-critical business decision-making in many organizations
- Such use cases as recommendation engines, targeted advertising, speech recognition, fraud detection, image recognition and categorization

Spark ML Pipelines

- The Pipeline API lets developers quickly assemble distributed machine learning pipelines
- The Pipeline API has been standardized for applying different machine learning algorithms
- The Pipeline API can combine multiple machine learning algorithms into a single pipeline
- The Pipeline API eases the implementation of data analytics and machine learning applications

Spark ML components

- Datasets
 - Used for storing and processing input data, transformed input data, feature vectors, labels, predictions
- Pipelines
 - ML workflows are implemented as pipelines consisting of a sequence of stages
- Pipeline stages
 - Each pipeline stage comprises of a Transformer or an Estimator

Spark ML components

- Transformer

- An algorithm that transforms an input DataFrame into another DataFrame with one or more features added to it, e.g. RegexTokenizer, Binarizer, OneHotEncoder, various indexers

- Estimator

- A machine learning algorithm that learns from the input data provided, e.g. LogisticRegression and RandomForest
- The input to an estimator is a DataFrame and the output is a Transformer, The output Transformers from these Estimators are the corresponding models, e.g. LogisticRegressionModel, RandomForestModel

The Behavior of the Pipeline

- It will execute each stage and pass the result of the current stage to the next
- If the stage is a Transformer, then the pipeline calls `transform()` on it
- If the stage is an Estimator, then the pipeline first calls `fit()` followed by `transform()`
- If it is the last stage in the pipeline, then the Estimator will not call `transform()` (after `fit()`)

Spark ML Pipeline Steps

- Data ingestion
- Data cleansing and preprocessing
- Feature engineering
 - Extract and generate specific features from the input data using various techniques, are then combined into a feature vector and passed to the next step in the process
 - E.g. VectorAssembler is used to create the feature vector from the specified DataFrame columns

Spark ML Pipeline Steps

- Model training
 - Specifying an algorithm
 - Split the input Dataset into training and test Datasets
 - The model is trained by calling the `fit()` method on the training Dataset
- Model validation
 - Evaluating and tuning the ML model
 - The model is applied to the test Dataset, using the `transform()` method
 - Performance measures for the model are computed, for example, accuracy, error

Spark ML Pipeline Steps

- Model selection
 - Choose the parameters for the Transformers and Estimators that produce the best ML model
- Cross-validation
 - Create a parameter grid and execute a grid-search for the most suitable set of parameters for a given model using a process called cross-validation

Spark ML Pipeline Steps

- Model Deployment
 - Deploy the best model for production
 - Convert the model parameters (such as coefficients, intercepts, or decision trees with branching logic) to other formats (such as JSON)

Spark ML Pipeline Steps

- Model Deployment
 - Deploy the best model for production
 - Convert the model parameters (such as coefficients, intercepts, or decision trees with branching logic) to other formats (such as JSON)

Feature Engineering

- The process of using domain knowledge of the data to create features
- Any attribute can be a feature
- The most challenging aspect of machine learning applications
- The quality and quantity/number of features influences the overall quality of the model
- An iterative process comprising of multiple cycles of data selection and model evaluation

What Feature Engineering do

- Feature transformations, such as indexing and binning, are used to reduce the dimensionality of predictor variables
- Removed irrelevant and low-frequency values from the model continuous variables are grouped into a reasonable number of bins
- Highly correlated, or redundant features can be dropped

What Feature Engineering do

- Multiple features can be combined to yield new features to reduce the overall dimensionality
- Normalize the values of some variables to avoid skewed results from using absolute values
- Apply transformations on the training Dataset to obtain a feature vector that will be fed into a machine learning algorithm

Features Selection

- Selecting features from raw data could lead to many different feature sets
- keep the ones that are most relevant to the problem to be solved
- The number of features can be limited
- Need time, patience, creativity, and familiarity with the raw input data

The Importance of Features

- The features with the highest scores are selected for inclusion in the training Dataset
- Generate new features from the raw data features if needed

The Approach to Estimate

- Group sets of related features and compare the performance of the model without those features to the complete model
- Execute a k-fold cross-validation for both, the complete and dropped models, and compare them on various statistical measures
- Visualization and applying specific methods known to work well on certain types of data

Methods for Textual Data

- NGram
- TF-IDF
- Feature Hashing

Dimensionality Reduction

- The attributes that are evaluated to be largely irrelevant to the problem need to be removed
- Limiting the number of resulting columns in the feature matrix using various transformations
- Selecting a subset of features that is most useful in solving the problem
- Compute correlation coefficients, covariances, and other statistics for choosing a good set of features

Dimensionality Reduction

- Principal Component Analysis (PCA)
- Unsupervised clustering methods for feature selection
- Search through various feature sets creating and evaluating models automatically
- Deep Learning

Tips to derive the Good Features

- Grouping for categorical values
- Limiting the number of categorical values
- Evaluate and add new features by computing polynomial features
- Evaluating each feature to test its correlation with the classes independently by using a ranking metric, E.g. top 10%
- Evaluating how good each feature is using criteria, such as Gini index and entropy

Tips to derive the Good Features

- Exploring the covariance between features, for example, if two features are changing the same way, it will probably not serve the overall purpose to select both of them as features
- Introducing new features if model is underfitting a Dataset

Tips to derive the Good Features

- Decompose date-time fields into separate fields for month, day, year
- Apply linear Transformers to numerical fields, such as weights and distances, for use in regression and other algorithms
- Explore storing a quantity measure as a rate or an aggregate quantity over a time interval to expose structures, such as seasonality

Spark ML Classification Model

- Perform EDA on input data
 - Data visualization using tools such as Zeppelin
 - Assessing feature types (numeric/categorical)
 - Computing basic statistics
 - Computing covariances and correlation coefficients
 - Creating pivot tables

Spark ML Classification Model

- Executing data pre-processing and data munging operations
 - Transformations required to convert the features from the source format to final variables
 - Categorical features may need to be transformed to a binary variable for each categorical value using one-hot encoding technique

Spark ML Classification Model

- Building the Spark ML pipeline
 - Feature Engineering
 - indexing categorical features and labels
 - assembling features into one column section
 - Spark ML classifier
 - Creating a Spark ML pipeline
 - Creating the training and test Datasets
 - Train the model
 - Test the model
 - Cross Validation
 - Choose the best model

Spark ML Classification Model

