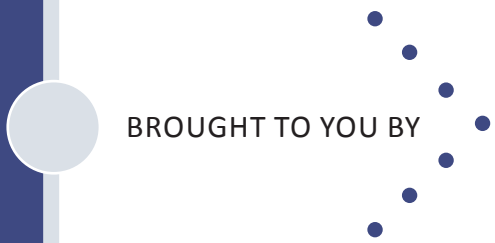
insideBIGDATA

Guide to Machine Learning

by Daniel D. Gutierrez, May 2014



REVOLUTION ANALYTICS

Introduction to Machine Learning

As the primary facilitator of data science and big data, machine learning has garnered much interest by a broad range of industries as a way to increase value of enterprise data assets. Through techniques of supervised and unsupervised statistical learning, organizations can make important predictions and discover previously unknown knowledge to provide actionable business intelligence. In this guide, we'll examine the principles underlying machine learning based on the R statistical environment. We'll explore machine learning with R from the open source R perspective as well as the more robust commercial perspective using Revolution Analytics' Revolution R Enterprise (RRE) for big data deployments.

SUPERVISED LEARNING is where a model is fit that relates the response to the predictors, with the aim of accurately predicting the response for future observations.

Instead of using labeled data sets, UNSUPERVISED LEARNING is a set of statistical tools intended for applications where there is only a set of feature variables measured across a number of observations.

Supervised machine learning is typically associated with prediction where for each observation of the predictor measurements (also known as feature variables) there is an associated response measurement (also known as the class label). Supervised learning is where a model is fit that relates the response to the predictors, with the aim of accurately predicting the response for future observations. Many classical learning algorithms such a linear regression and logistic regression, operate in the supervised domain.

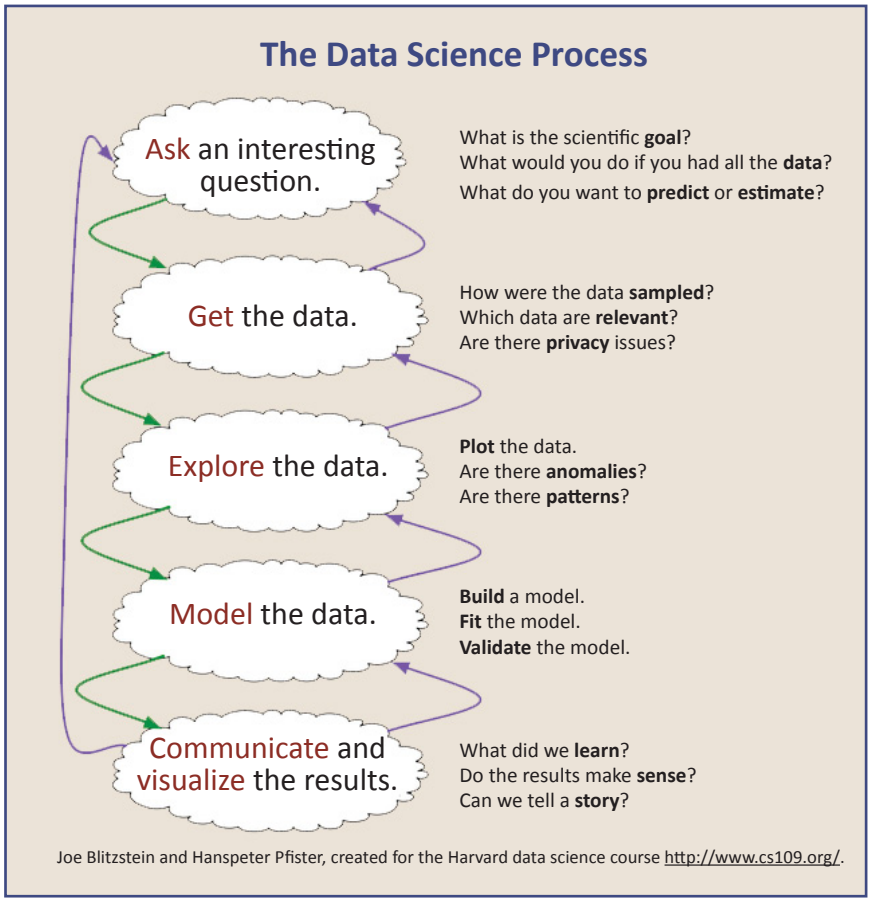
Contents

Unsupervised machine learning is a more open-ended style of statistical learning. Instead of using labeled data sets, unsupervised learning is a set of statistical tools intended for applications where there is only a set of feature variables measured across a number of observations. In this case, prediction is not the goal because the data set is unlabeled, i.e. there is no associated response variable that can supervise the analysis. Rather, the goal is to discover interesting things about the measurements on the feature variables. For example, you might find an informative way to visualize the data, or discover subgroups among the variables or the observations.

One commonly used unsupervised learning technique is k-means clustering that allows for the discovery of "clusters" of data points. Another technique called principal component analysis (PCA) is used for dimensionality reduction, i.e. reducing the number of feature variables while maintaining the variation in the data, in order to simplify the data used in other learning algorithms, speed up processing, and reduce the required memory footprint.

There are a number of other steps in the data science pipeline *(see figure below)* that contribute to the success of a machine learning project: understanding the problem domain, data access, data munging, exploratory data analysis, feature engineering, model selection, model validation, deploy, visualization and communicate results. We'll briefly take a look at these steps in this guide.

Open source R is the choice of an increasing number of data science practitioners worldwide, however, there are significant limitations to the open source version of R in terms of capacity and performance for production systems. Big data is largely about volume of data, so R needs a more robust infrastructure to manage this environment. This is where a commercial product like RRE comes into play. This guide will illustrate the use of both open source R and commercial R for machine learning applications.



R – The Data Scientist's Choice

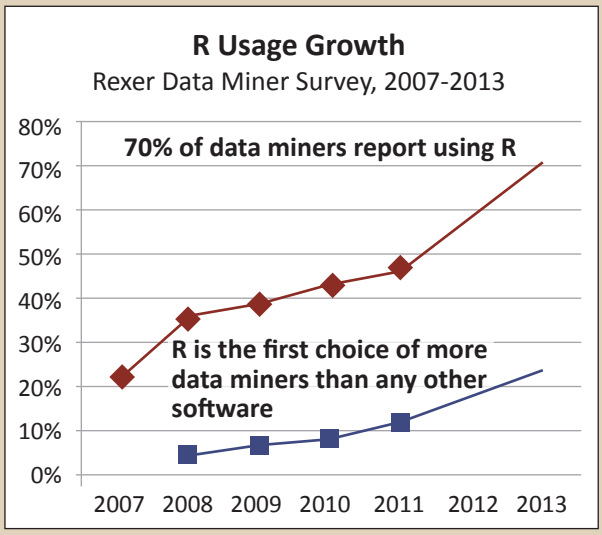
Although there are many choices for performing tasks related to data analysis, data modeling, and machine learning, R has become the overwhelming favorite among data scientists today. In large part this is due to the extensive use of R in academia over commercial products like SAS and SPSS. There are currently spirited debates between the R user community and both the SAS and Python communities as to what is the best tool for data scientists. R has compelling justifications including the availability of free open source R, a widely used extensible language, over 5,000 packages available on CRAN to extend the functionality of R, top-rated visualization capabilities using ggplot2, and a thriving user community with the likes of blog consolidator: r-bloggers.com. The growth of R can be seen in the graphic below:

Here is a short list of facts about R that demonstrate its popularity and growth:

1. **R is the highest paid IT skill** (Dice.com survey, January 2014)
2. **R is the most-used data science language after SQL** (O'Reilly survey, January 2014)
3. **R is used by 70% of data miners** (Rexer survey, October 2013)
4. **R is #15 of all programming languages** (RedMonk language rankings, January 2014)
5. **R is growing faster than any other data science language** (KDNuggets survey, August 2013)
6. **R is the #1 Google Search for Advanced Analytics software** (Google Trends, March 2014)
7. **R has more than 2 million users worldwide** (Oracle estimate, February 2012)

The only issue with R is its inherent limitation as a scalable production environment. R is notoriously memory-based, meaning it can only run on the confines of a single compute environment. It cannot be easily deployed in a distributed computing architecture so big data deployments are not possible. In this guide, we'll use RRE as an example of a commercial version of R that was specifically designed for big data applications. RRE is composed of a set of big data functions based on Parallel External Memory Algorithms (PEMAs) that run on the following platforms: multicore workstations and servers (including those in the cloud), Windows HPC server clusters, IBM platform LSF clusters, Hadoop clusters and Teradata. The RRE RevoScaleR package contains these high-performance analysis functions.

R is Exploding in Popularity & Functionality



Source: www.rexeranalytics.com

"I've been astonished by the rate at which R has been adopted. Four years ago, everyone in my economics department [at the University of Chicago] was using Stata; now, as far as I can tell, R is the standard tool, and students learn it first."

– Deputy Editor for New Products at Forbes

"A key benefit of R is that it provides near-instant availability of new and experimental methods created by its user base — without waiting for the development/release cycle of commercial software. SAS recognizes the value of R to our customer base..."

– Product Marketing Manager SAS Institute, Inc.

Data Access

The first step in a machine learning project is to access disparate data sets and bring them into the R environment, typically a data frame object. The data frame often serves as the primary argument passed to a machine learning algorithm as its data source. R has a wide range of data access methods such as **read.table(), read.csv(), read.xlsx(), readLines(),** as well as packages that extend R's ability to access external data sources, such as **RODBC** to read SQL databases, and **RJSONIO** to read JSON files. For unstructured social media data sources, R has packages such as **twitteR** to read Twitter streams. R offers a high level of flexibility in terms of the types of data sources that can be accessed.

Importing data from XDF is the quickest and most efficient method of accessing big data files and getting the data into the R environment.

For big data projects, however, open source R bogs down in terms of speed and capacity. There's only so much data that open source R can accommodate since it keeps all objects in memory. Exactly how many observations can be stored depends on the number of feature variables per observation and the amount of data stored in each variable, e.g. character-based categorical data take up more memory than quantitative variables. Plus, the open source R input routines have not been optimized for best performance so reading in a large data set requires patience.

RRE in contrast, has the **RxTextData()** big data function to read delimited and fixed format data sets. Other big data access functions are: SPSS with **RxSpssData(),** SAS with RxSasData(), ODBC databases with **RxOdbcData(),** and Teradata with **RxTeradata().** RRE also offers a highly optimized binary file format called XDF that includes data compression capabilities. Importing data from XDF is the quickest and most efficient method of accessing big data files and getting the data into the R environment.

Data Munging

The next phase of a machine learning project involves a process called "data munging." It is often the case where the data imported into the R environment is inconvenient or incompatible with machine learning algorithms, so with data munging (also known as data transformation) the data can be massaged into a more hospitable form. Data munging cannot be taken lightly as many times it can consume up to 80% of the entire machine learning project. The amount of time needed for a particular project depends on the health of the data: how clean, how complete, how many missing elements, etc.

Open source R has many mechanisms and packages to facilitate data transformations and cleaning, e.g. **dplyr, reshape2, lubridate,** etc. The specific tasks and their sequence should be recorded carefully so you can replicate the process. This process becomes part of your *data pipeline.* Here is a short list of typical data munging tasks, but there potentially are many more depending on the data:

- Data sampling
- Create new variables
- Discretize quantitative variables
- Date handling (e.g. changing dates stored as integers to R date objects)
- Merge, order, reshape data sets
- Other data manipulations such as changing categorical variables to multiple binary variables
- Handling missing data
- Feature scaling
- Dimensionality reduction

Open source R has its limitations with respect to performing data munging on large data sets. Sometimes a good option is to perform transformations outside the R environment, possibly at the data warehouse level using ETL steps based on SQL stored procedures. Once the data is in shape, then it can be imported into R.

Alternatively, RRE has big data functions for data transformations like **rxSplit()** which can be used to minimize the number of passes through a large data set. It can efficiently split a large data set into pieces in order to distribute it across the nodes of a cluster. You might also want to split your data into training and test data so that you can fit a model using the training data and validate it using the test data. RRE also can perform sorting with **rxSort(),** merging with **rxMerge(),** and missing value handling with big data features such as the **removeMissing** argument for the **rxDTree()** algorithm.

Exploratory Data Analysis

Once you have clean, transformed data inside the R environment, the next step for machine learning projects is to become intimately familiar with the data using exploratory data analysis (EDA). The way to gain this level of familiarity is to utilize the many features of the R statistical environment that support this effort — numeric summaries, plots, aggregations, distributions, densities, reviewing all the levels of factor variables and applying general statistical methods. A clear understanding of the data provides the foundation for model selection, i.e. choosing the correct machine learning algorithm to solve your problem.

Open source R has many mechanisms for EDA including **hist()** for histograms, **boxplot()** for boxplots, **barplot()** for barplots, **plot()** for scatterplots, **heatmap()** for heatmaps, etc. Using these tools allows for a deep understanding of the data being employed for machine learning. This understanding serves the purpose of feature engineering.

The open source R has difficulty performing EDA on big data data sets, however RRE has a number of big data functions that process data in chunks using the specified compute context. For example **rxGetVarInfo()** can find potential outliers, **rxSummary()** show summaries for continuous and categorical variables, **rxHistogram()** for density estimation, **rxLinePlot()** for drawing line plots.

Feature engineering is the process of determining which predictor variables will contribute the most to the predictive power of a machine learning algorithm.

Feature engineering is when you use your knowledge about the data to select and create features that make machine learning algorithms work better.

Feature Engineering

Feature engineering is the process of determining which predictor variables will contribute the most to the predictive power of a machine learning algorithm. There are two commonly used methods for making this selection – the *Forward Selection Procedure* starts with no variables in the model. You then iteratively add variables and test the predictive accuracy of the model until adding more variables no longer makes a positive effect. Next, the *Backward Elimination Procedure* begins with all the variables in the model. You proceed by removing variables and testing the predictive accuracy of the model.

The process of feature engineering is as much of an art as a science. Often feature engineering is a give-and-take process with exploratory data analysis to provide much needed intuition about the data. It's good to have a domain expert around for this process, but it's also good to use your imagination. Feature engineering is when you use your knowledge about the data to select and create features that make machine learning algorithms work better.

One problem with machine learning is too much data. With today's big data technology, we're in a position where we can generate a large number of features. In such cases, fine-tuned feature engineering is even more important.

Supervised Learning

Supervised machine learning is the type of statistical learning most often associated with data science since it offers a number of methods for prediction namely regression and classification.

Open Source R has algorithms that implement a statistical process for estimating the relationships among variables and is widely used for prediction and forecasting.

Regression is the most common form of supervised learning. In regression, there is a response quantitative variable, such as a systolic blood pressure reading of a hospital patient, based on a series of feature variables such as the patient's age, gender, BMI, and blood sodium levels. The relationship between systolic blood pressure and the feature variables in the training set would provide a predictive model. The model is built using complete observations which provide the value of the response variable as well as the feature variables.

Open Source R has algorithms to implement regression such as the linear model **lm(),** regression trees with **tree(),** and ensemble methods with **randomForest().** In a nutshell, these algorithms implement a statistical process for estimating the relationships among variables and are widely used for prediction and forecasting.

RRE has big data versions of regression algorithms including **rxLinMod()** for fitting linear regression models, **rxPredict()** to compute fitted values and model residuals, as well as regression tree support with **rxDTree()** which fits tree-based models using a binning-based recursive partitioning algorithm with a numeric response variable and **rxDForest()** which is an ensemble of decision trees where each tree is fitted to a bootstrap sample of the original data. These algorithms are designed to work with arbitrarily large data sets.

Classification is another popular type of supervised learning. In classification, there is a response categorical variable, such as income bracket, which could be partitioned into three classes or categories: high income, middle income, and low income. The classifier examines a data set where each observation contains information on the response variable as well as the predictor (feature) variables. For example, suppose an analyst would like to be able to classify the income brackets of persons not in the data set, based on characteristics associated with that person, such as age, gender, and occupation. This is a classification task that would proceed as follows: examine the data set containing both the feature variables and the already classified response variable, income bracket. In this way, the algorithm learns about which combinations of variables are associated with which income brackets. This data set is called the training set. Then the algorithm would look at new observations for which no information about income bracket is available. Based on the classifications in the training set, the algorithm would assign classifications to the new observations. For example, a 58 year old female controller might be classified in the high-income bracket.

Open Source R has a number of classification algorithms such as logistic regression with **glm(),** decision trees with **tree(),** and ensemble methods with **randomForest().** In a nutshell, classification is the process of identifying to which of a set of categories a new observation belongs, on the basis of a training set of data containing observations whose category membership is known.

RRE has big data versions of classification algorithms including logistic regression using **rxGlm()** or the optimized **rxLogit()** for modeling data with a binary response variable, as well as classification tree support with **rxDTree().** Also included for classification is RRE's Decision Forest algorithm **rxDForest().** These algorithms are designed to work with arbitrarily large data sets.

Unsupervised Learning

Unsupervised Learning is often considered more challenging than supervised learning because there is no corresponding response variable for each observation. In this sense, we're working blind and the question is what kind of statistical analysis is possible? Using unsupervised techniques like clustering, we can seek to understand the relationships between the variables or between the observations by determining whether observations fall into relatively distinct groups. For example, in a customer segmentation analysis we might observe multiple variables: gender, age, zip code, income, etc. Our belief may be that the customers fall in different groups like frequent shoppers and infrequent shoppers. A supervised learning analysis would be possible if the customer shopping history were available, but this is not the case in unsupervised learning — we don't have response variables telling us whether a customer is a frequent shopper or not. Instead, we can attempt to cluster the customers on the basis of the feature variables in order to identify distinct customer groups.

Another style of unsupervised learning is called Principal Component Analysis (PCA) and is best thought of as a dimensionality reduction technique where the number of feature variables is reduced while retaining nearly the same amount of variation.

Open Source R has the k-means clustering algorithm **kmeans()** which uses a partitioning approach for determining clusters. K-means clustering is a technique that groups observations of quantitative data using one of several *iterative relocation algorithms* — that is, starting from some initial selection of clusters, which may be random, points are moved from one cluster to another so as to minimize sums of squares.

Using unsupervised techniques like clustering, we can seek to understand the relationships between the variables or between the observations by determining whether observations fall into relatively distinct groups.

Another style of unsupervised learning is called Principal Component Analysis (PCA) and is best thought of as a dimensionality reduction technique where the number of feature variables is reduced while retaining nearly the same amount of variation. R has several algorithms to compute principal components: **prcomp(), princomp(), pca(),** and **svd().**

RRE has big data implementations of these unsupervised learning algorithms such as **rxKmeans().** To do PCA on an arbitrary number of rows, RRE offers **rxCovCor()** and its relatives **rxCov()** and **rxCor()** that can be used to calculate an eigenvalue decomposition of a covariance or correlation matrix respectively and then use R's **princomp()** algorithm. Another technique called *density estimation* is often included with unsupervised learning, and RRE has its **rxHistogram()** to provide a big data solution.

Production Deployment

The final step in completing a machine learning project with R is to determine how best to deploy the solution to a production environment. Deploying open source R can problematic for some/all of the following reasons:

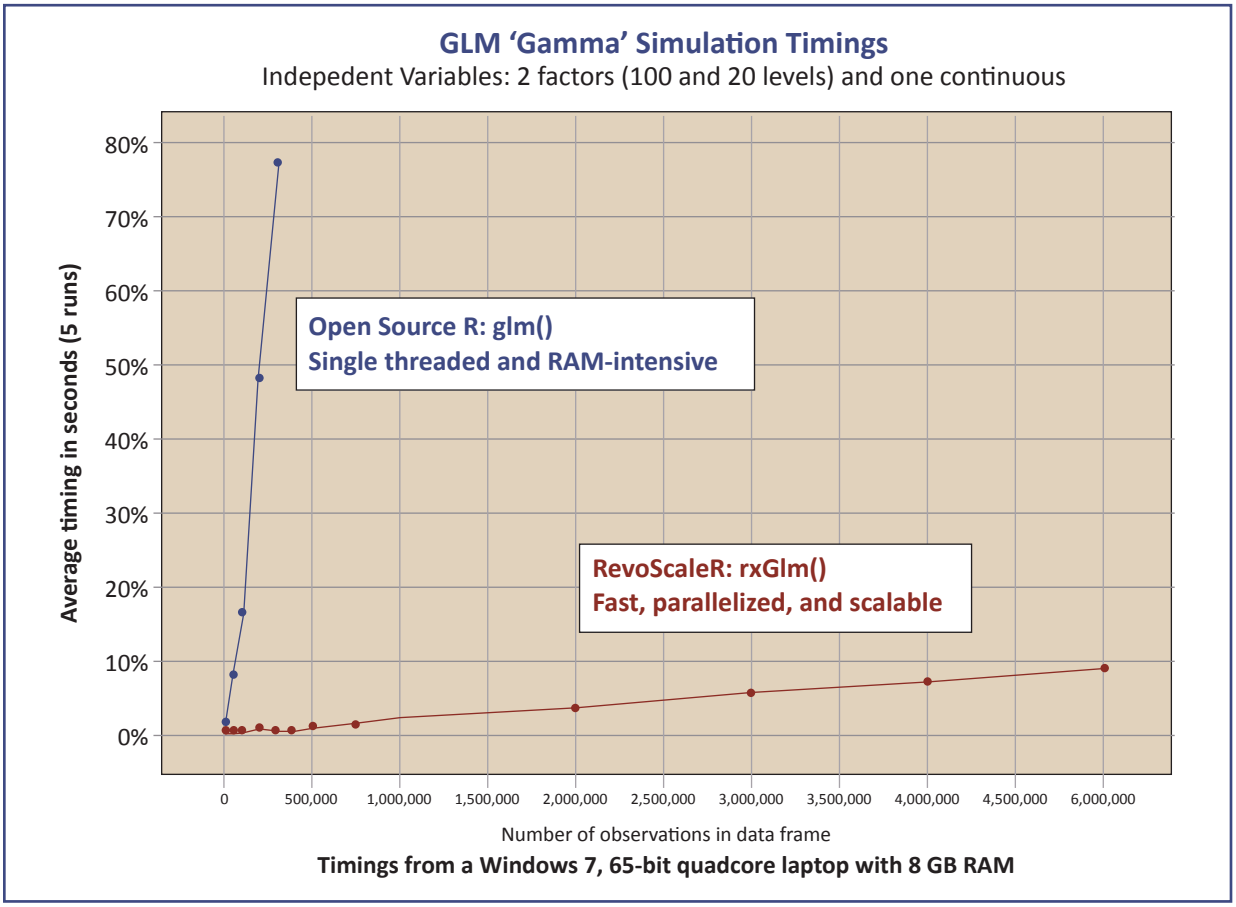
- In-memory bound – need hybrid memory and disk scalability
- Single threaded – need parallel threading
- Packages not optimized for big data
- Risk of deploying open source with GPL license

Many times, in order to avoid these issues, data scientists will opt to convert their working R solution to a different programming environment like Python. This path, however, is far from optimal since it requires redevelopment and significant retesting.

Commercial products like RRE offer a much more robust production environment that has big data in mind. RRE's catchphrase "write once, deploy anywhere" indicates that you can develop your R solution once and deploy with a number of different choices. For example, you can specify a compute context with **rxSetComputeContext()** — HPC cluster, Hadoop, Teradata data warehouse, etc. and run your R code unaltered. You also can use **rxExec()** to run any R function on a parallel infrastructure like Hadoop. RRE includes many big data parallelized algorithms for descriptive statistics, statistical tests, and sampling.

In addition, R performance and capacity is production-ready with RRE. The following graph differentiates between **glm()** and **rxGlm().** RRE will use all cores (e.g. quad core processor) and cluster nodes available on your parallel platform to perform required computations.

RRE also includes DeployR: a web services software development kit (SDK) for exposing R functions via web services for custom application and integration with third party products such as Alteryx, Tableau, Excel, etc.



R and Hadoop

In order to take advantage of the benefits of distributed computing, Revolution Analytics has created the RevoScaleR package, providing functions for performing scalable and extremely high performance data management, analysis, and visualization. Hadoop provides a distributed file system and a MapReduce framework for distributed computation. The data manipulation and analysis functions in RevoScaleR are appropriate for small and large datasets, but are particularly useful in three common situations:

1. to analyze data sets that are too big to fit in memory and,
2. to perform computations distributed over several cores, processors, or nodes in a cluster, or
3. to create scalable data analysis routines that can be developed locally with smaller data sets, then deployed to larger data and/or a cluster of computers.

The RevoScaleR high performance analysis functions are portable. The same high performance functions work on a variety of computing platforms, including Windows and RHEL workstations and servers and distributed computing platforms such as Hadoop, IBM Platform LSF, and Microsoft HPC Server. So, for example, you can do exploratory analysis on your laptop, then deploy the same analytics code on a Hadoop cluster. The underlying RevoScaleR code handles the distribution of the computations across cores and nodes, so you don't have to worry about it.

The architecture of the RevoScaleR package provides an under-the-covers example of scalable analytics in Hadoop:

- A master process is initiated to run the main thread of the algorithm.
- The master process initiates a Hadoop MapReduce job to make a pass through the data.
- Parallel and distributed Hadoop "Mapper" tasks produce "intermediate results objects" for each chunk of data. These are merged using a Hadoop "Reducer" task.

PEMAs run in-database, so that not only is analytical speed improved, but data is also analyzed in-place in the database, eliminating data movement delays and latency.

- The master process examines the results. For iterative algorithms, it decides if another pass through the data is required. If so, it initiates another MapReduce job and repeats.
- When complete, the final results are computed and returned.

When running "inside" Hadoop, the RevoScaleR analysis functions process data contained in the Hadoop Distributed File System (HDFS). This "inside" architecture provides the greatest scalability. HDFS data can also be accessed directly from RevoScaleR, without performing the computations within the Hadoop framework; this is known as a "beside" architecture. The "beside" architecture is often used when processing small to medium data volumes.

R and Teradata

Another production deployment option available for RRE uses massively parallel analytical processing inside the Teradata platform. The highest possible performance is achieved because RRE for Teradata includes a library of PEMAs (included with all versions of RRE and usable on a wide variety of big data platforms in addition to Teradata). PEMAs are pre-built, external-memory, parallelized versions of the most common statistical and predictive analytics algorithms and run directly in parallel on the Teradata nodes. These PEMAs run in-database, so that not only is analytical speed improved, but data is also analyzed in-place in the database, eliminating data movement delays and latency.

As in the Hadoop case, RRE supports both an "inside" and a "beside" Teradata architecture.

R in the Cloud

As an alternative to an on premise hardware solution to run a production deployment of an R-based machine learning solution, companies can now deploy to the cloud by using a hosted version of RRE on Amazon Web Services (AWS). RRE is available through the AWS Marketplace and offers scalable predictive analytics designed to analyze data sets of various sizes. The data can be stored in Amazon S3 and Amazon Relational Data Service (RDS). An attractive aspect of a cloud implementation is that you only pay as you go, just for capacity you need with no long-term commitments. This alternative is attractive to start-up companies wishing to avoid large up-front costs for data centers and commodity servers. It's also attractive for companies with spiky or cyclical demand for analytics.

Summary

In summary, the R statistical environment is a well-accepted platform for developing machine learning solutions to business problems. R is the clear choice globally for data scientists, data analysts, and researchers alike. R is unique in terms of statistical environments in that open source R caters to practitioners exploring modestly sized data sets, performing model design, and evaluating model performance. Commercial versions of R such as RRE, take the next step forward by providing a robust, big data capable production deployment platform.

Revolution Analytics *is the leading commercial provider of software and services based on the open source R project for statistical computing. Revolution Analytics meets the demands of data-driven businesses worldwide, applying data science to innovate and derive value from Big Data. Its Revolution R Enterprise software builds on the power of R with performance, scalability and enterprise readiness. Revolution Analytics also offers consulting, training and support services.*

Leading organizations including Merck, Bank of America and Acxiom rely on Revolution R Enterprise for their data analysis, development and mission-critical production needs. Revolution Analytics is committed to fostering the growth of the R community, and offers free and discounted licenses of Revolution R Enterprise to academics, academic institutions and public service organizations. Revolution Analytics is headquartered in Mountain View, California.

For more information on Revolution Analytics visit: www.revolutionanalytics.com or call 1-650-646-9545