In the past few years, the generation of data and our capability to store and process it has grown exponentially. There is a need for scalable analytics frameworks and people with the right skills to get the information needed from this Big Data. Apache Mahout is one of the first and most prominent Big Data machine learning platforms. It implements machine learning algorithms on top of distributed processing platforms such as Hadoop and Spark.

Starting with the basics of Mahout and machine learning, you will explore prominent algorithms and their implementation in Mahout development. You will learn about Mahout building blocks, addressing feature extraction, reduction and the curse of dimensionality, delving into classification use cases with the random forest and Naive Bayes classifier and item and user-based recommendation. You will then work with clustering Mahout using the K-means algorithm and implement Mahout without MapReduce. Finish with a flourish by exploring end-to-end use cases on customer analytics and test analytics to get a real-life practical know-how of analytics projects.

Who this book is written for
If you are a Java developer and want to use Mahout and machine learning to solve Big Data Analytics use cases then this book is for you. Familiarity with shell scripts is assumed but no prior experience is required.

What you will learn from this book
- Configure Mahout on Linux systems and set up the development environment
- Become familiar with the Mahout command line utilities and Java APIs
- Understand the core concepts of machine learning and the classes that implement them
- Integrate Apache Mahout with newer platforms such as Apache Spark
- Solve classification, clustering, and recommendation problems with Mahout
- Explore frequent pattern mining and topic modeling, the two main application areas of machine learning
- Understand feature extraction, reduction, and the curse of dimensionality
- Explore end-to-end use cases on customer analytics and test analytics to get a real-life practical know-how of analytics projects.

In this package, you will find:

- The author biography
- A preview chapter from the book, Chapter 1 'Introduction to Mahout'
- A synopsis of the book’s content
- More information on Learning Apache Mahout
Chandramani Tiwary is a Data Scientist with special interest towards building data driven products. He is a founding member of the Data Science team at Gainsight and was instrumental in framing Data Science use cases for customer success management in the SaaS industry. He is also an advisor to multiple organizations. He can be followed on Twitter at https://twitter.com/Mani_on_Twitt and on LinkedIn at https://www.linkedin.com/today/author/60666042.
Learning Apache Mahout is aimed at providing a strong foundation in machine learning using Mahout. This book is ideal for learning the core concepts of machine learning and the basics of Mahout. This book will go from the basics of Mahout and machine learning, to feature engineering and the implementation of various machine learning algorithms in Mahout. Algorithm usage examples will be explained using both the Mahout command line and its Java API. We will conclude the book with two chapters of end-to-end case studies. Ideally, chapters 1, 2 and 3 should be read sequentially, chapters 4 to 8 in any order, and chapters 9 and 10 after chapter 1 to 8 have been completed.

What this book covers

Chapter 1, Introduction to Mahout, covers the setup of the learning environment, installation, and the configuration of the various tools required for this book. It will discuss the need for a machine learning library such as Mahout and introduce the basics of Mahout with command line and code examples.

Chapter 2, Core Concepts in Machine Learning, covers the fundamental concepts in machine learning. It will discuss the important steps involved in a machine learning project, such as data processing, model training, and efficacy, and provides an intuitive explanation of different algorithms.

Chapter 3, Feature Engineering, covers the most important phase of a machine learning project, feature extraction and representation. It will discuss common data preprocessing tasks, manual and automated feature transformation, feature selection, and dimensionality reduction.
Chapter 4, Classification with Mahout, covers classification algorithms implemented in Mahout. It will discuss the important phases of building a classifier, such as preprocessing data, creating a train and test set, and measuring model efficacy. The algorithms that will be covered are logistic regression, random forest, and naïve Bayes.

Chapter 5, Frequent Pattern Mining and Topic Modeling, covers algorithms for frequent pattern mining and topic modeling. This chapter will provide an intuitive explanation of the algorithms and include both command line and code examples, while also providing practical examples.

Chapter 6, Recommendation with Mahout, covers algorithms to build recommender systems in Mahout. It will discuss item-based and user-based recommenders. This chapter will provide an intuitive explanation of the algorithms and include both command line and code examples, while also providing practical examples.

Chapter 7, Clustering with Mahout, covers algorithms to perform clustering in Mahout. It will discuss algorithms such as k-means, fuzzy k-means, streaming k-means, and so on. This chapter will provide an intuitive explanation of the algorithm and include both command line and code examples, while also providing practical examples.

Chapter 8, New Paradigm in Mahout, covers the porting of Mahout on top of Apache Spark. It will discuss the installation and configuration of Mahout and Spark, explain the important concepts of Spark and Mahout binding, and cover some basic examples.

Chapter 9, Case Study – Churn Analytics and Customer Segmentation, covers the steps involved in a machine learning project from start to finish. It will discuss all the important steps that need to be performed for a successful machine learning project. It will take a couple of use cases from customer analytics, churn analytics, and customer segmentation, to walk through the process.

Chapter 10, Case Study – Text Analytics, covers the steps involved in a text analytics project. It will discuss the vector space model of representing text, text clustering, and classification.
Introduction to Mahout

Mahout is an open source machine learning library from Apache. Mahout primarily implements clustering, recommender engines (collaborative filtering), classification, and dimensionality reduction algorithms but is not limited to these.

The aim of Mahout is to provide a scalable implementation of commonly used machine learning algorithms. Mahout is the machine learning tool of choice if the data to be used is large. What we generally mean by large is that the data cannot be processed on a single machine. With Big Data becoming an important focus area, Mahout fulfils the need for a machine learning tool that can scale beyond a single machine. The focus on scalability differentiates Mahout from other tools such as R, Weka, and so on.

The learning implementations in Mahout are written in Java, and major portions, but not all, are built upon Apache's Hadoop distributed computation project using the MapReduce paradigm. Efforts are on to build Mahout on Apache Spark using Scala DSL. Programs written in Scala DSL will be automatically optimized and executed in parallel on Apache Spark. Commits of new algorithms in MapReduce have been stopped and the existing MapReduce implementation will be supported.

The purpose of this chapter is to understand the fundamental concepts behind Mahout. In particular, we will cover the following topics:

- Why Mahout
- When Mahout
- How Mahout
Why Mahout

We already have many good open source machine learning software tools. The statistical language R has a very large community, good IDE, and a large collection of machine learning packages already implemented. Python has a strong community and is multipurpose, and in Java we have Weka.

So what is the need for a new machine learning framework?

The answer lies in the scale of data. Organizations are generating terabytes of data daily and there is a need for a machine learning framework that can process that amount of data.

That begs a question, can't we just sample the data and use existing tools for our analytics use cases?

Simple techniques and more data is better

Collecting and processing data is much easier today than, say, a decade ago. IT infrastructure has seen an enormous improvement; ETL tools, click stream providers such as Google analytics, stream processing frameworks such as Kafka, Storm, and so on have made collecting data much easier. Platforms like Hadoop, Cassandra, and MPP databases such as Teradata have made storing and processing huge amount of data much easier than earlier. From a large-scale production algorithm standpoint, we have seen that simpler algorithms on very large amounts of data produce reasonably good results.

Sampling is difficult

Sampling may lead to over-fitting and increases the complexity of preparing data to build models to solve the problem at hand. Though sampling tends to simplify things by allowing scientists to work on a small sample instead of the whole population and helps in using existing tools like R to scale up to the task, getting a representative sample is tricky.

I'd say when you have the choice of getting more data, take it. Never discard data. Throw more (commodity) hardware at the data by using platforms and tools such as Hadoop and Mahout.
Community and license
Another advantage of Mahout is its license. Mahout is Apache licensed, which means that you can incorporate pieces of it into your own software regardless of whether you want to release your source code. However, other ML software, such as Weka, are under the GPL license, which means that incorporating them into your software forces you to release source code for any software you package with Weka components.

When Mahout
We have discussed the advantages of using Mahout, let's now discuss the scenarios where using Mahout is a good choice.

Data too large for single machine
If the data is too large to process on a single machine then it would be a good starting point to think about a distributed system. Rather than scaling and buying bigger hardware, it could be a better option to scale out, buy more machines, and distribute the processing.

Data already on Hadoop
A lot of enterprises have adopted Hadoop as their Big Data platform and have used it to store and aggregate data. Mahout has been designed to run algorithms on top of Hadoop and has a relatively straightforward configuration.

If your data or the bulk of it is already on Hadoop, then Mahout is a natural choice to run machine learning algorithms.

Algorithms implemented in Mahout
Do check whether the use case that needs to be implemented has a corresponding algorithm implemented in Mahout, or you have the required expertise to extend Mahout to implement your own algorithms.
How Mahout
In this section, you will learn how to install and configure Mahout.

Setting up the development environment
For any development work involving Mahout, and to follow the examples in this book, you will require the following setup:

- Java 1.6 or higher
- Maven 2.0 or higher
- Hadoop 1.2.1
- Eclipse with Maven plugin
- Mahout 0.9

I prefer to try out the latest version, barring when there are known compatibility issues. To configure Hadoop, follow the instructions on this page http://hadoop.apache.org/docs/r1.2.1/single_node_setup.html. We will focus on configuring Maven, Eclipse with the Maven plugin, and Mahout.

Configuring Maven
Maven can be downloaded from one of the mirrors of the Apache website http://maven.apache.org/download.cgi. We use Apache Maven 3.2.5 and the same can be downloaded using this command:

```
wget http://apache.mirrors.tds.net/maven/maven-3/3.2.5/binaries/apache-maven-3.2.5-bin.tar.gz
cd /usr/local
sudo tar xzf $HOME/Downloads/ /usr/local/apache-maven-3.2.5-bin.tar.gz
sudo mv apache-maven-3.2.5 maven
sudo chown -R $USER maven
```

Configuring Mahout
Mahout can be configured to be run with or without Hadoop. Currently, efforts are on to port Mahout on Apache Spark but it is in a nascent stage. We will discuss Mahout on Spark in Chapter 8, New Paradigm in Mahout. In this chapter, you are going to learn how to configure Mahout on top of Hadoop.

We will have two configurations for Mahout. The first we will use for practicing command line examples of Mahout and the other, compiled from source, will be used to develop Mahout code using Java API and Eclipse.
Though we can use one Mahout configuration, I will take this opportunity to discuss both approaches.

Download the latest Mahout version using one of the mirrors listed at the Apache Mahout website https://mahout.apache.org/general/downloads.html. The current release version is mahout-distribution-0.9.tar.gz. After the download completes, the archive should be in the Downloads folder under the user's home directory. Type the following on the command line. The first command moves the shell prompt to the /usr/local directory:

```bash
cd /usr/local
```

Extract the downloaded file to the directory mahout-distribution-0.9.tar.gz under the /usr/local directory. The command `tar` is used to extract the archive:

```bash
sudo tar xzf $HOME/Downloads/mahout-distribution-0.9.tar.gz
```

The third command `mv` renames the directory from mahout-distribution-0.9 to mahout:

```bash
sudo mv mahout-distribution-0.9 mahout
```

The last command `chown` changes the ownership of the file from the root user to the current user. The Linux command `chown` is used for changing the ownership of files and directories. The argument `-R` instructs the `chown` command to recursively change the ownership of subdirectories and `$USER` holds the value of the logged in user's username:

```bash
sudo chown -R $USER mahout
```

We need to update the `.bashrc` file to export the required variables and update the `$PATH` variable:

```bash
cd $HOME
vi .bashrc
```

At the end of the file, copy the following statements:

```bash
#Statements related to Mahout
export MAVEN_HOME=/usr/local/maven
export MAHOUT_HOME=/usr/local/mahout
PATH=$PATH:/bin:$MAVEN_HOME/bin:$MAHOUT_HOME/bin
###end of mahout statement
```

Exit from all existing terminals, start a new terminal, and enter the following command:

```bash
echo $PATH
```

Check whether the output has the path recently added to Maven and Mahout.
Introduction to Mahout

Type the following commands on the command line; both commands should be recognized:

```
mvn --version
mahout
```

Configuring Eclipse with the Maven plugin and Mahout

Download Eclipse from the Eclipse mirror mentioned on the home page. We have used Eclipse Kepler SR2 for this book. The downloaded archive should be in the Downloads folder of the user's home directory. Open a terminal and enter the following command:

```
cd /usr/local
sudo tar xzf $HOME/Downloads/eclipse-standard-kepler-SR2-linux-gtk-x86_64.tar.gz
sudo chown -R $USER eclipse
```

Go into the Eclipse directory and open up the Eclipse GUI. We will now install the Maven plugin. Click on Help then Eclipse Marketplace and then in the search panel type m2e and search. Once the search results are displayed hit Install and follow the instructions. To complete the installation hit the Next button and press the Accept button whenever prompted. Once the installation is done, Eclipse will prompt for a restart. Hit OK and let Eclipse restart.
Now to add Mahout dependency to any Maven project we need, add the following dependency in the `pom.xml` file:

```xml
<dependency>
  <groupId>org.apache.mahout</groupId>
  <artifactId>mahout-core</artifactId>
  <version>0.9</version>
</dependency>
<dependency>
  <groupId>org.apache.mahout</groupId>
  <artifactId>mahout-examples</artifactId>
  <version>0.9</version>
</dependency>
<dependency>
  <groupId>org.apache.mahout</groupId>
  <artifactId>mahout-math</artifactId>
  <version>0.9</version>
</dependency>
```

Eclipse will download and add all the dependencies.

Now we should import the code repository of this book to Eclipse. Open Eclipse and follow the following sequence of steps. The `pom.xml` file has all the dependencies included in it and Eclipse will download and resolve the dependencies.

Go to File | Import | Maven | Existing Maven Projects | Next | Browse to the location of the source folder that comes with this book | Finish.

---

**Downloading the example code**

You can download the example code files from your account at http://www.packtpub.com for all the Packt Publishing books you have purchased. If you purchased this book elsewhere, you can visit http://www.packtpub.com/support and register to have the files e-mailed directly to you.
Introduction to Mahout

**Mahout command line**

Mahout provides an option for the command line execution of machine learning algorithms. Using the command line, an initial prototype of the model can be built quickly.

A few examples of command line are discussed. A great place to start is to go through Mahout's example scripts, the example scripts, are located under the Mahout home folder in the `examples` folder:

```
$ cd $MAHOUT_HOME
$ cd examples/bin
$ ls --ltr
```

The Mahout example scripts are as follows:

Open the file `README.txt` in `vi` editor and read the description of the scripts. We will be discussing them in the subsequent sections of this chapter:

```
$ vi README.txt
```
The description of the example script is as follows:

It is a good idea to try out a few command line Mahout algorithms before writing Mahout Java code. This way we can shortlist a few algorithms that might work on the given data and problem, and save a lot of time.

### A clustering example

In this section, we will discuss the command line implementation of clustering in Mahout and use the example script as reference.

On the terminal please type:

```bash
 vi cluster-reuters.sh
```

This script clusters the Reuters dataset using a variety of algorithms. It downloads the dataset automatically, parses and copies it to HDFS (Hadoop Distributed File System), and based upon user input, runs the corresponding clustering algorithm.
On the `vi` terminal type the command:

```
:set number
```

This will display the line numbers of the lines in the file. The algorithms implemented are `kmeans`, `fuzzykmeans`, `lda`, and `streamingkmeans`; line 42 of the code has a list of all algorithms implemented in the script:

```
algorithm=( kmeans fuzzykmeans lda streamingkmeans) #A list of all algorithms implemented in the script
```

Input is taken from the user in line 51 by the `read` statement:

```
read -p "Enter your choice : " choice
```

Line 57 sets the `temp` working directory variable:

```
WORK_DIR=/tmp/mahout-work-${USER}
```

On line 79, the `curl` statement downloads the Reuters data to the working directory, first checking whether the file is already present in the working directory between lines 70 to 73:

```
curl http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.tar.gz -o ${WORK_DIR}/reuters21578.tar.gz
```

From line 89, the Reuters `tar` is extracted to the `reuters-sgm` folder under the working directory:

```
tar xzf ${WORK_DIR}/reuters21578.tar.gz -C ${WORK_DIR}/reuters-sgm
```

**Reuter's raw data file**

Let's have a look at one of the raw files. Open the `reut2-000.sgm` file in a text editor such as `vi` or `gedit`. 
The Reuter's raw file looks like this:

```
$qf 200-2015 00:00:00/15:00:00/08:00/08:00
$TITLE $AMIA COCA REVIEW $TITLE
$DATETIMESalvador, Palibin $SPAMLEVEL=ABBY-80Shutters continued throughout the week in the Bolivian cocoa zone, alleviating the drought since early January and improving prospects for the coming campaign.

Although normal harvests have not been restored, Centenario Smith said the weather would be late this year. 

The dry period means the temperature will be late this year. 

Although normal harvests have not been restored, Centenario Smith said the weather:
```

The Reuters data is distributed in 22 files, each of which contains 1,000 documents, except for the last (reut2-021.sgm), which contains 578 documents. The files are in the SGML (standard generalized markup language) format, which is similar to XML. The SGML file needs to be parsed.

On line 93, the Reuters data is parsed using Lucene. Lucene has built-in classes and functions to process different file formats. The logic of parsing the Reuters dataset is implemented in the ExtractReuters class. The SGML file is parsed and the text elements are extracted from it.

```
Apache Lucene is a free/open source information retrieval software library.
```

We will use the ExtractReuters class to extract the sgm file to text format.

```
$MAHOUT org.apache.lucene.benchmark.utils.ExtractReuters ${WORK_DIR}/reuters-sgm ${WORK_DIR}/reuters-out
```

---
Introduction to Mahout

Now let's look at the Reuters processed file. The following figure is a snapshot taken from the text file extracted from the sgm files we saw previously by the ExtractReuters class:

On lines 95 to 101, data is loaded from a local directory to HDFS, deleting the reuters-sgm and reuters-out folders if they already exist:

```
echo "Copying Reuters data to Hadoop"
$HADOOP dfs -rmdir $(WORK_DIR)/reuters-sgm
$HADOOP dfs -rmdir $(WORK_DIR)/reuters-out
$HADOOP dfs -put $(WORK_DIR)/reuters-sgm $(WORK_DIR)/reuters-sgm
$HADOOP dfs -put $(WORK_DIR)/reuters-out $(WORK_DIR)/reuters-out
```

On line 105, the files are converted into sequence files. Mahout works with sequence files.

```
$MAHOUT seqdirectory -i $(WORK_DIR)/reuters-out -o $(WORK_DIR)/reuters-out-seqdir -c UTF-8 -chunk 64 -xm sequential
```

On lines 109 to 111, the sequence file is converted to a vector representation. Text needs to be converted into a vector representation so that a machine learning algorithm can process it. We will talk about text vectorization in details in Chapter 10, Case Study – Text Analytics.

```bash
$MAHOUT seq2sparse -i ${WORK_DIR}/reuters-out-seqdir/
-o ${WORK_DIR}/reuters-out-seqdir-sparse-kmeans --maxDFPercent 85 --namedVector
```

From here on, we will only explain the k-means algorithm execution; we encourage you to read and understand the other three implementations too. A detailed discussion of clustering will be covered in Chapter 7, Clustering with Mahout.

Clustering is the process of partitioning a bunch of data points into related groups called clusters. K-means clustering partitions a dataset into a specified number of clusters by minimizing the distance between each data point and the center of the cluster using a distance metric. A distance metric is a way to define how far or near a data point is from another. K-means requires users to provide the number of clusters and optionally user-defined cluster centroids.

To better understand how data points are clustered together, please have a look at the sample figure displaying three clusters. Notice that the points that are nearby are grouped together into three distinct clusters. A few points don't belong to any clusters, those points represent outliers and should be removed prior to clustering.
Here is an example of the command line for k-means clustering:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--input (-i)</td>
<td>This is the path to the job input directory.</td>
</tr>
<tr>
<td>--clusters (-c)</td>
<td>These are the input centroids and they must be a SequenceFile of type Writable or Cluster/Canopy. If k is also specified, then a random set of vectors will be selected and written out to this path first.</td>
</tr>
<tr>
<td>--output (-o)</td>
<td>This is the directory pathname for the output.</td>
</tr>
<tr>
<td>--distanceMeasure</td>
<td>This is the class name of DistanceMeasure; the default is SquaredEuclidean.</td>
</tr>
<tr>
<td>--convergenceDelta</td>
<td>This is the convergence delta value; the default is 0.5.</td>
</tr>
<tr>
<td>--maxIter (-x)</td>
<td>This is the maximum number of iterations.</td>
</tr>
<tr>
<td>--maxRed (-r)</td>
<td>This is the number of reduce tasks; this defaults to 2.</td>
</tr>
<tr>
<td>--k (-k)</td>
<td>This is the k in k-means. If specified, then a random selection of k vectors will be chosen as the Centroid and written to the cluster's input path.</td>
</tr>
<tr>
<td>--overwrite (-ow)</td>
<td>If this is present, overwrite the output directory before running the job.</td>
</tr>
<tr>
<td>--help (-h)</td>
<td>This prints out Help.</td>
</tr>
<tr>
<td>--clustering (-cl)</td>
<td>If this is present, run clustering after the iterations have taken place.</td>
</tr>
</tbody>
</table>

Lines 113 to 118 take the sparse matrix and runs the k-means clustering algorithm using the cosine distance metric. We pass --k the number of clusters as 20 and --x the maximum number of iterations as 10:

```bash
$MAHOUT kmeans \
-i ${WORK_DIR}/reuters-out-seqdir-sparse-kmeans/tfidf-vectors/ \
-c ${WORK_DIR}/reuters-kmeans-clusters \
-o ${WORK_DIR}/reuters-kmeans \
-dm org.apache.mahout.common.distance.CosineDistanceMeasure \
-x 10 -k 20 -ow --clustering \
```

Lines 120 to 125 take the cluster dump utility, read the clusters in sequence file format, and convert them to text files:

```bash
$MAHOUT clusterdump \
-i ${WORK_DIR}/reuters-kmeans/clusters-*.final \
-o ${WORK_DIR}/reuters-kmeans/clusterdump \
-d ${WORK_DIR}/reuters-out-seqdir-sparse-kmeans/dictionary.file-0 \
```
Chapter 1

```bash
-dt sequencefile -b 100 -n 20 --evaluate -dm org.apache.mahout.common.distance.CosineDistanceMeasure -sp 0 \
--pointsDir ${WORK_DIR}/reuters-kmeans/clusteredPoints \
&

cat ${WORK_DIR}/reuters-kmeans/clusterdump
```

The `clusterdump` utility outputs the center of each cluster and the top terms in the cluster. A sample of the output is shown here:

```
Weight : [props : optional]: Point:

VL-17930(n=1223) c=[0:0.039, 0.001:0.007, 0.025:0.008, 0.04:0.008, 0.05:0.014, 0.0625:0.016, 0.07:0.02
Top Terms:
  pct => 2.7694996927116001
  1 => 2.63786436396078
  4 => 2.1153221974485934
  B => 2.0397753262923422
  bond => 1.9461722151724317
  mlm => 1.8595630760903492
  issue => 1.85918251766470211
  lead => 1.82338999489252727
  coupon => 1.7956528815692891
  manager => 1.7729479791819304
  2 => 1.7198113299612324
  bank => 1.6741793239496814
  priced => 1.5499389947407635
  due => 1.46888888115640229
  said => 1.400677342223497
  issuing => 1.278256596563795
  5 => 1.261963713450792
  ltd => 1.222211218621842
  issues => 1.201082583264574
  eurobond => 1.1860701322745129
```

### A classification example

In this section, we will discuss the command line implementation of classification in Mahout and use the example script as a reference.

Classification is the task of identifying which set of predefined classes a data point belongs to. Classification involves training a model with a labeled (previously classified) dataset and then predicting new unlabeled data using that model.

The common workflow for a classification problem is:

1. Data preparation
2. Train model
3. Test model
4. Performance measurement

Repeat steps until the desired performance is achieved, or the best possible solution is achieved or the project's time is up.
On the terminal, please type:

```
vi classify-20newsgroups.sh
```

On the `vi` terminal, type the following command to show the line numbers for lines in the script:

```
:set number
```

The algorithms implemented in the script are `cnaivebayes`, `naivebayes`, `sgd`, and a last option `clean`, which cleans up the work directory.

Line 44 creates a working directory for the dataset and all input/output:

```
export WORK_DIR=/tmp/mahout-work-${USER}
```

Lines 64 to 74 download and extract the `20news-bydate.tar.gz` file after making sure it is not already downloaded:

```
if [ ! -e ${WORK_DIR}/20news-bayesinput ]; then
  if [ ! -e ${WORK_DIR}/20news-bydate ]; then
    if [ ! -f ${WORK_DIR}/20news-bydate.tar.gz ]; then
      echo "Downloading 20news-bydate"
    fi
    mkdir -p ${WORK_DIR}/20news-bydate
    echo "Extracting..."
    cd ${WORK_DIR}/20news-bydate && tar xzf ../20news-bydate.tar.gz && cd ..&cd ..
  fi
fi
```

The 20 newsgroups dataset consists of messages, one per file. Each file begins with header lines that specify things such as who sent the message, how long it is, what kind of software was used, and the subject. A blank line follows, and then the message body follows as unformatted text.

Lines 90 to 101 prepare the directory and copy the data to the Hadoop directory:

```
echo "Preparing 20newsgroups data"
rm -rf ${WORK_DIR}/20news-all
mkdir ${WORK_DIR}/20news-all
cp -R ${WORK_DIR}/20news-bydate/*/* ${WORK_DIR}/20news-all
```

```
if [ "$HADOOP_HOME" != "" ] && [ "$MAHOUT_LOCAL" == "" ]; then
  echo "Copying 20newsgroups data to HDFS"
```

set +e
$HADOOP dfs -rmr ${WORK_DIR}/20news-all
set -e
$HADOOP dfs -put ${WORK_DIR}/20news-all ${WORK_DIR}/20news-all
fi

A snapshot of the raw 20newsgroups data file is provided below.

Lines 104 to 106 convert the full 20 newsgroups dataset into sequence files:

$ mahout seqdirectory -i ${WORK_DIR}/20news-all -o ${WORK_DIR}/20news-seq -ow

Lines 109 to 111 convert the sequence files to vectors calculating the term frequency and inverse document frequency. Term frequency and inverse document frequency are ways of representing text using numeric representation:

./bin/mahout seq2sparse \
-0 ${WORK_DIR}/20news-seq \n-o ${WORK_DIR}/20news-vectors -lnorm -nv -wt tfidf
Lines 114 to 118 split the preprocessed dataset into training and testing sets. The test set will be used to test the performance of the model trained using the training sets:

```
./bin/mahout split \
-i $(WORK_DIR)/20news-vectors/tfidf-vectors \
--trainingOutput $(WORK_DIR)/20news-train-vectors \
--testOutput $(WORK_DIR)/20news-test-vectors \
--randomSelectionPct 40 --overwrite --sequenceFiles -xm sequential
```

Lines 120 to 125 train the classifier using the training sets:

```
./bin/mahout trainnb \
-i $(WORK_DIR)/20news-train-vectors -el \n-o $(WORK_DIR)/model \n-li $(WORK_DIR)/labelindex \n-ow $c
```

Lines 129 to 133 test the classifier using the test sets:

```
./bin/mahout testnb \
-i $(WORK_DIR)/20news-train-vectors\n-m $(WORK_DIR)/model \n-l $(WORK_DIR)/labelindex \n-ow -o $(WORK_DIR)/20news-testing $c
```

**Mahout API – a Java program example**

Though using Mahout from the command line is convenient, fast, and serves the purpose in many scenarios, learning the Mahout API is important too. The reason being, using the API gives you more flexibility in terms of creating your machine learning application, and not all algorithms can be easily called from the command line. Working with the Mahout API helps to understand the internals of a machine learning algorithm.

Mahout core JAR files have the implementation of the main machine learning classes and the Mahout examples JAR file has some example code and wrappers built around the Mahout core classes. It is worth spending time going through the documentation and getting an overall understanding. The documentation for the version you are using can be found in the Mahout installation directory.
The Mahout documentation directory looks like this:

We will now look at a Mahout code example. We will write a classification example in which we will train an algorithm to predict whether a client has subscribed to a term deposit. Classification refers to the process of labeling a particular instance or row to a particular predefined category, called a class label. The purpose of the following example is to give you a hang of the development using Mahout, Eclipse, and Maven.

The dataset

We will use the bank-additional-full.csv file present in the learningApacheMahout/data/chapter4 directory as the input for our example. First, let's have a look at the structure of the data and try to understand it. The following table shows various input variables along with their data types:

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Description</th>
<th>Variable Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age</td>
<td>Age of the client</td>
<td>Numeric</td>
</tr>
<tr>
<td>Job</td>
<td>Type of job, for example, entrepreneur, housemaid, or management</td>
<td>Categorical</td>
</tr>
<tr>
<td>Marital</td>
<td>Marital status</td>
<td>Categorical</td>
</tr>
<tr>
<td>Education</td>
<td>Education level</td>
<td>Categorical</td>
</tr>
<tr>
<td>Default</td>
<td>Has the client defaulted on credit?</td>
<td>Categorical</td>
</tr>
</tbody>
</table>
Based on many attributes of the customer, we try to predict the target variable $y$ (has the client subscribed to a term deposit?), which has a set of two predefined values, Yes and No. We need to remove the header line to use the data.

We will use logistic regression to build the model; logistic regression is a statistical technique that computes the probability of an unclassified item belonging to a predefined class.

You might like to run the example with the code in the source code that ships with this book; I will explain the important steps in the following section. In Eclipse, open the code file `OnlineLogisticRegressionTrain.java` from the package `chapter4.logistic.src`, which is present in the directory `learningApacheMahout/src/main/java/chapter4/src/logistic` in the code folder that comes with this book.

The first step is to identify the source and target folders:

```java
String inputfile = "data/chapter4/train_data/input_bank_data.csv";
String outputFile = "data/chapter4/model";
```
Once we know where to get the data from, we need to tell the algorithm about how to interpret the data. We pass the column name and the corresponding column type; here, \( n \) denotes the numeric column and \( w \), the categorical columns of the data:

```java
```

```java
List<String> typeList = Arrays.asList("n", "w", "w", "w", "w", "w", "w", "w", "w", "w", "w", "n", "n", "n", "n", "n", "n", "n");
```

Set the classifier parameters. `LogisticModelParameters` is a wrapper class, in Mahout's example distribution, used to set the parameters for training logistic regression and to return the instance of a `CsvRecordFactory`:

```java
LogisticModelParameters lmp = new LogisticModelParameters();
    lmp.setTargetVariable("y");
    lmp.setMaxTargetCategories(2);
    lmp.setNumFeatures(20);
    lmp.setUseBias(false);
    lmp.setTypeMap(predictorList, typeList);
    lmp.setLearningRate(0.5);
    int passes = 50;
```

We set the the target variable \( y \) to be used for training, the maximum number of target categories to be 2 (Yes, No), the number of features or columns in the data excluding the target variable (which is 20), and some other settings (which we will learn about later in this book). The variable passed has been given a value of 50, which means the maximum number of iteration over the data will be 50.

The `CsvRecordFactory` class returns an object to parse the CSV file based on the parameters passed. The `LogisticModelParameters` class takes care of passing the parameters to the constructor of `CsvRecordFactory`. We use the class `RandomAccessSparseVector` to encode the data into vectors and train the model using `lr.train(targetValue, input)`:

```java
CsvRecordFactory csv = lmp.getCsvRecordFactory();
lr = lmp.createRegression();
for (int pass = 0; pass < passes; pass++) {
    BufferedReader in = new BufferedReader(new FileReader(inputFile));

    csv.firstLine(in.readLine());
```
Introduction to Mahout

```java
String line = in.readLine();
int lineCount = 2;
while (line != null) {
    Vector input = new RandomAccessSparseVector(lmp.getNumFeatures());
    int targetValue = csv.processLine(line, input);

    // update model
    lr.train(targetValue, input);
    k++;

    line = in.readLine();
    lineCount++;
}
```

The output after running the code would be an equation denoting the logistic regression. Excerpts of the equation are copied here:

\[ y = -97.230 \cdot \text{age} + -12.713 \cdot \text{campaign} + \ldots \]

You will learn about logistic regression, how to interpret the equation, and how to evaluate the results in detail in Chapter 4, Classification with Mahout.

### Parallel versus in-memory execution mode

Mahout has both parallel and in-memory execution for many machine learning algorithms. In-memory execution can be used when the data size is smaller or to try out different algorithms quickly without installing Hadoop. In-memory execution is restricted to one machine whereas the parallel are designed to run on different machines. The parallel execution is implemented over Hadoop using the MapReduce paradigm, and for parallel execution; we call the code via the driver class to run the Hadoop MapReduce job. Let's see which algorithms have single machine and parallel execution. We have grouped the algorithms according to the paradigm such as collaborative filtering, classification, and so on. The first column of the table is the name of the column, the second column indicates whether the algorithm has a single machine implementation, and the third column indicates whether the algorithm has a parallel execution implementation.
The collaborative filtering table is as follows:

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Single machine</th>
<th>Parallel</th>
</tr>
</thead>
<tbody>
<tr>
<td>User-based collaborative filtering</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>Item-based collaborative filtering</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Matrix factorization with alternating least squares</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Matrix factorization with alternating least squares on implicit feedback</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Weighted matrix factorization</td>
<td>Y</td>
<td>N</td>
</tr>
</tbody>
</table>

The classification table is as follows:

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Single machine</th>
<th>Parallel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logistic regression</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>Naïve Bayes/Complementary naïve Bayes</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Random forest</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Hidden Markov models</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>Multilayer perceptron</td>
<td>Y</td>
<td>N</td>
</tr>
</tbody>
</table>

The clustering table is as follows:

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Single machine</th>
<th>Parallel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Canopy clustering</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>k-means clustering</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Fuzzy k-means</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Streaming k-means</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Spectral clustering</td>
<td>N</td>
<td>Y</td>
</tr>
</tbody>
</table>

The dimensionality reduction table is as follows:

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Single machine</th>
<th>Parallel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Singular value decomposition</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>Lanczos algorithm</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Stochastic SVD</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Principal component analysis</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>
The topic models table is as follows:

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Single machine</th>
<th>Parallel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latent Dirichlet allocation</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>

The miscellaneous table is as follows:

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Single machine</th>
<th>Parallel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frequent pattern mining</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>RowSimilarityJob</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>ConcatMatrices</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Collocations</td>
<td>N</td>
<td>Y</td>
</tr>
</tbody>
</table>

Summary

In this chapter, we discussed the guiding principle of Mahout and tried out some examples to get a hands-on feel of Mahout. We discussed why, when, and how to use Mahout and walked through the installation steps of the required tools and software. We then learned how to use Mahout from the command line and from the code, and finally concluded with a comparison between the parallel and the single-machine execution of Mahout.

This is the beginning of what will hopefully be an exciting journey. In the forthcoming chapters, we will discuss a lot of practical applications for Mahout. In the next chapter, we will discuss the core concepts of machine learning. A clear understanding of the concepts of different machine learning algorithms is of paramount importance for a successful career in data analytics.
Where to buy this book

You can buy Learning Apache Mahout from the Packt Publishing website.
Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.

Click here for ordering and shipping details.