PART 1. BATCH COMPUTING MODELS FOR BIG DATA ANALYTICS
5. ADVANCED DATA ANALYTICS WITH APACHE SPARK

Today’s topics
• Advanced Data Analytics with Apache Spark

Spark stack of libraries
- SQL and DataFrames
- Machine learning
- GraphX
- Spark Streaming

FAQs
• Term project proposal
• New deadline: 10/11
• PA1 demo
• CSB120

Large scale data analysis using Spark with case study
Predicting Forest Cover with Decision Trees
Regressions and Classifications

- **Regression analysis**
  - A statistical process for estimating the relationships among variables
  - Dependent variable vs. independent variable ("predictors")
  - How the typical value of the dependent variable changes when any one of the independent variables is varied, while the other independent variables are held fixed
  - Predicting numerical values
  - Income, temperature, size

- **Classifications**
  - Identifying which of a set of categories a new observation belongs
  - Predicting a label or category
  - Spam, picture of a cat

Vectors and Features

- **Feature vector**
  - These features together, in order
  - \((13.1, 19.0, 0.73, \text{cloudy}, 1)\)
  - Features do **NOT** need to be the same type

- **Categorical features**
  - Features that have no ordering
  - Cloudy, clear

- **Numeric features**
  - Features that can be quantified by a number and have a meaningful ordering
  - \(23^\circ \text{F}, 56^\circ \text{F}\ (23^\circ \text{F} < 56^\circ \text{F})\)

Training Examples

- A learning algorithm needs to train on data in order to make predictions
- **Inputs**
  - Correct outputs (from historical data)
    - "One day, the weather was between 12 and 16 degrees Celsius, with 10% humidity, clear, with no forecast of a cold snap, and the following day the high temperature was 17.2 degrees"
  - Target (output)
    - 17.2 degree
- Feature vector often includes the target value
  - \((13.1, 19.0, 0.73, \text{cloudy}, 1, 17.2)\)

Decision Trees and Forests

- **Decision trees** can naturally handle both categorical and numeric features
  - Easy to parallelize
  - Robust to outliers
    - A few extreme and possibly erroneous data points may not affect predictions at all

- **Random Decision Forests**
  - Extended decision tree algorithm

Decision tree

- Spark MLlib’s DecisionTree and RandomForest implementation
- Each decision leads to one of two results
  - Prediction or another decision

[Image of decision tree diagram]
Example: Finding a good pet

<table>
<thead>
<tr>
<th>Name</th>
<th>Weight (kg)</th>
<th># of legs</th>
<th>color</th>
<th>Good pet?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fido</td>
<td>20.5</td>
<td>4</td>
<td>Brown</td>
<td>Y</td>
</tr>
<tr>
<td>Mr. Slither</td>
<td>3.1</td>
<td>0</td>
<td>Green</td>
<td>N</td>
</tr>
<tr>
<td>Nemo</td>
<td>0.2</td>
<td>0</td>
<td>Tan</td>
<td>Y</td>
</tr>
<tr>
<td>Daisy</td>
<td>1300.8</td>
<td>4</td>
<td>Grey</td>
<td>N</td>
</tr>
<tr>
<td>Kitty</td>
<td>12.1</td>
<td>4</td>
<td>Grey</td>
<td>Y</td>
</tr>
<tr>
<td>Jim</td>
<td>150.9</td>
<td>2</td>
<td>Tan</td>
<td>N</td>
</tr>
<tr>
<td>McPig</td>
<td>0.1</td>
<td>100</td>
<td>Brown</td>
<td>N</td>
</tr>
<tr>
<td>Spot</td>
<td>10.0</td>
<td>4</td>
<td>Brown</td>
<td>Y</td>
</tr>
</tbody>
</table>

Decision tree for “Finding a good pet” example

- Weight >= 100kg?
  - No
  - Is color green yes
    - Yes
    - Not suitable
    - No
    - Is color green no
      - Yes
      - Not suitable

Covertype dataset

- Dataset with records of the types of forest covering parcels of land in Colorado, USA
  - Each example contains several features describing:
    - Each parcel of land
    - Elevation, slope, distance to water, shade, and soil type
  - The forest cover type is to be predicted
    - From the rest of features
    - Total 54 features
  - Used in Kaggle competition
  - It includes categorical and numeric features
    - 581,012 examples

Preparing data

- The covtype.data file should be extracted and copied into HDFS
  - File is available at /user/ds/

- LabeledPoint
  - The Spark MLlib abstraction for a feature vector
  - Consists of a Spark MLlib Vector of features, and a target value (label)
  - LabeledPoint is only for numeric features
  - It can be used with categorical features, with appropriate encoding

Using LabeledPoint for the categorical features

- One-hot coding
  - One categorical feature that takes on N distinct values becomes N numeric features, each taking on the value 0 or 1
  - Exactly one of the N values has value 1 and the others are 0
    - Cloudy, rainy or clear
      - Cloudy: 1,0,0
      - Rainy: 0,1,0
      - Clear: 0,0,1

- 1-of-N coding
  - Cloudy: 1
  - Rainy: 2
  - Clear: 3
Categorical values in Covtype data set

- The covtype.info file says that four of the columns are actually a one-hot encoding of a single categorical feature
- The target itself is a categorical value encoded as the values 1 to 7
- The remaining features are numeric features in various units, like meters, degrees, or a qualitative "index" value

A First Decision Tree

- Spark MLlib requires input in the form of LabeledPoint objects

```scala
import org.apache.spark.mllib.linalg._
import org.apache.spark.mllib.regression._
val rawdata = sc.textFile("hdfs:///user/ds/covtype.data")
val data = rawdata.map { line =>
  val values = line.split(',').map(_.toDouble)
  LabeledPoint(values.last - 1, Vectors.dense(values.init))
}
```

Splitting data

- Training, cross-validation, and test
- Training and CV sets are used to choose a good setting of hyperparameters for this data set
- Test set is used to produce an unbiased evaluation of the expected accuracy of a model built with those hyperparameters

```scala
val Array(trainData, testData) = data.randomSplit(Array(0.8, 0.1))
trainData.cache()
testData.cache()
```

Building a DecisionTreeModel on the training set

- Building a DecisionTreeModel on the training set with some default arguments
- Compute some metrics about the resulting model using the CV set

```scala
def getMetrics(model: DecisionTreeModel, data: RDD[LabeledPoint]): MulticlassMetrics = {
  val predictionsAndLabels = data.map(example =>
    (model.predict(example.features), example.label))
  new MulticlassMetrics(predictionsAndLabels)
}
val model = DecisionTree.trainClassifier(trainData, 7, Map[Int, Int](), "gini", 4, 100)
val metrics = getMetrics(model, testData)
```

Confusion matrix

- 7 x 7 matrix
- The row number corresponds to an actual correct value
- The column number corresponds to a predicted value

```scala
val matrix = metrics.confusionMatrix
... 
```

10/3/2017
CS535 Big Data
Fall 2017 Colorado State University
http://www.cs.colostate.edu/~cs535

Week 7- A
Precision in multiclass metrics

- Binary classification
  - Positive vs. negative class
  - Precision is the fraction of examples that the classifier marked positive that are actually positive
    - \( PPV = \frac{TP}{TP+FP} \)
  - Recall is the fraction of all examples that are actually positive that the classifier marked positive
    - \( TPR = \frac{TP}{TP+FN} \)

- Multiclass problem
  - Positive class vs. negative (all else)

Is 70% accuracy good?

- Classifier that classifies at random in proportion to its prevalence in the training set
  - What is the baseline?
  - A broken clock will be correct twice a day
  - Randomly guessing a classification would also be occasionally correct

Decision Tree Hyperparameters [1/2]

- Hyperparameters
  - Values we have to choose by building models
  - Maximum depth, maximum bins, and impurity measure
- Maximum depth
  - Limits the number of levels in the decision tree
  - Useful to avoid overfitting the training data
- Maximum bins
  - feature \( \leq \) value
  - feature in (value1, value2, ...)
  - A larger number of bins requires more processing time
  - More optimised decision rule

Decision Tree Hyperparameters [2/2]

- Good rule should distinguish examples more meaningfully
  - Example
    - A rule that divides the Covtype set into only 1-3 category and 4-7 category would be a great rule
  - A good rule divides the training data's target values into relatively homogeneous or "pure" subsets
  - Minimizing the impurity of the two subsets

Gini impurity

- Gini impurity
  - Measuring impurity degree
  - Within a subset, it is the probability that a randomly chosen classification of a randomly chosen example is incorrect
  - Includes the sum of products of proportions of classes
    - \( I_g(p) = 1 - \sum_{i=1}^{c} p_i^2 \)
  - If the subset contains only one class
    - The value is 0

Entropy

- Borrowed from information theory
  - How much uncertainty does the collection of target values in the subset contain?
    - \( I_e(p) = \sum_{i=1}^{c} p_i \log(1/p_i) = -\sum_{i=1}^{c} p_i \log(p_i) \)
Tuning Decision Trees

- Spark tries a number of combinations of impurity measure, maximum depth or number of bins and reports the results

```scala
val evaluations = for (impurity <- Array("gini", "entropy"), depth <- Array(1, 20), bins <- Array(10, 300))
  yield {
    val model = DecisionTree.trainClassifier(trainData, 7, Map[Int, Int](), impurity, depth, bins)
    val predictionsAndLabels = cvData.map(example => (model.predict(example.features), example.label))
    val accuracy = new MulticlassMetrics(predictionsAndLabels).precision((impurity, depth, bins), 1)
    (impurity, depth, bins, accuracy)
}
evaluations.sortBy(_._2).reverse.foreach println
```

10/3/2017  
CS535 Big Data  
Fall 2017 Colorado State University  
http://www.cs.colostate.edu/~cs535  
10/3/2017  
Sangmi Lee Pallickara  
Week 7  

Categorical Features Revisited

- Map[Int, Int](){
  - Keys: Indices of features in the input Vector
  - Values: Distinct value counts
- Empty Map()
  - No features should be treated as categorical
  - All are numeric
- Numeric representation of categorical features
  - It can cause errors
  - The algorithm would be trying to learn from an ordering that has no meaning

10/3/2017  
CS535 Big Data  
Fall 2017 Colorado State University  
http://www.cs.colostate.edu/~cs535  
10/3/2017  
Sangmi Lee Pallickara  
Week 7  

Converting one-hot encoding to 1-n encoding

```scala
val numWilderness = values.slice(10, 14).indexOf(1.0).toDouble
val numSoil = values.slice(14, 54).indexOf(1.0).toDouble
val featureVector = Vectors.dense(values.slice(0, 10) :+ numWilderness :+ numSoil)
```

- 4 "wilderness" features
- 40 "soil" features
- Add derived features back to first 10

10/3/2017  
CS535 Big Data  
Fall 2017 Colorado State University  
http://www.cs.colostate.edu/~cs535  
10/3/2017  
Sangmi Lee Pallickara  
Week 7  

Treating the categorical features with one-hot encoding

- Encodes the categorical features as several binary 0/1 values
- Any decision rule on the "numeric" features will choose thresholds between 0 and 1
  - All are equivalent since all values are 0 or 1
- Considers the values of the underlying categorical feature individually
- Increases memory usage

10/3/2017  
CS535 Big Data  
Fall 2017 Colorado State University  
http://www.cs.colostate.edu/~cs535  
10/3/2017  
Sangmi Lee Pallickara  
Week 7  

Converting one-hot encoding to 1-n encoding

```scala
val eval = testData.map { line =>
  val values = line.split(',').map(_.toDouble)
  val numWilderness = values.slice(10, 14).indexOf(1.0).toDouble
  val numSoil = values.slice(14, 54).indexOf(1.0).toDouble
  val featureVector = Vectors.dense(values.slice(0, 10) :+ numWilderness :+ numSoil)
  (label, featureVector)
}  
```

- Specify value count for categorical features 10, 11
- Causes these features to be treated as categorical

10/3/2017  
CS535 Big Data  
Fall 2017 Colorado State University  
http://www.cs.colostate.edu/~cs535  
10/3/2017  
Sangmi Lee Pallickara  
Week 7
- Tree-building process completes several times faster
- By treating categorical features as categorical features, it improves accuracy by almost 3%