Gradient boosting trees (GBTs)

Module Objectives

•After completing this set of lessons, you should be able to:

- Understand the Pipelines API for Gradient-Boosted Trees
- Understand and use RFs and GBTs parameters
- Outline the differences between RFs and GBTs regarding its parameters

Gradient-boosting trees

- Like Random forests, they are ensembles of Decision Trees
- Iteratively train decision trees in order to minimize a loss function
- Supports binary classification
- Supports regression
- Supports continuous and categorical features

Basic algorithm

- Iteratively trains a sequence of decision trees
- On each iteration, uses the current ensemble to make label predictions and compares it to true labels
- Re-labels dataset to put more emphasis on instances with poor predictions, according to a loss function
- With each iteration, reduces the loss function, thus correct for previous mistakes
- Supported loss functions:
  - –classification: Log Loss (twice binomial negative log likelihood)
  - –regression: Squared Error (L2 loss, default) and Absolute Error (L1 loss, more robust to outliers)

Gradient-Boosted Trees Parameters

–loss: loss function (Log Loss, for classification, Squared and Absolute errors, for regression)

–numIterations: number of trees in the ensemble

- •each iteration produces one tree
- •if it increases:
  - –model gets more expressive, improving training data accuracy
  - –test-time accuracy may suffer (if too large)

–learningRate: should NOT need to be tuned

- •if behavior seems unstable, decreasing it may improve stability

Validation while training

- Gradient-Boosted Trees can overfit when trained with more trees
- The method `runWithValidation` allows validation while training
  - takes a pair of RDDs: training and validation datasets
- Training is stopped when validation error improvement is less than the tolerance specified as `validationTol` in `BoostingStrategy`
  - validation error decreases initially and later increases
  - there might be cases in which the validation error does not change monotonically
    - set a large enough negative tolerance
    - examine validation curve using `evaluateEachIteration`, which gives the error or loss per iteration
    - tune the number of iterations

GBT Classification (1)

```
from pyspark.mllib.tree import GradientBoostedTrees, GradientBoostedTreesModel
from pyspark.mllib.util import MLUtils

data = MLUtils.loadLibSVMFile(sc, "sample_libsvm_data.txt")
trainingData, testData = data.randomSplit([0.7, 0.3])

labels = testData.map(lambda x: x.label)
features = testData.map(lambda x: x.features)
```

GBT Classification (2)

```
model = GradientBoostedTrees.trainClassifier(trainingData,
                                             categoricalFeaturesInfo={},
                                             numIterations=3)

print(model.toDebugString())
```

```
TreeEnsembleModel classifier with 3 trees

  Tree 0:
    If (feature 406 <= 20.0)
     Predict: -1.0
    Else (feature 406 > 20.0)
     Predict: 1.0
```

GBT Classification (3)

```
predictions = model.predict(features)

labelsAndPredictions = labels.zip(predictions)

testErr = labelsAndPredictions.filter(lambda (v, p): v != p).count() / float(testData.count())

print('Test Error = ' + str(testErr))
```

```
Test Error = 0.0833333333333
```

GBT regression

```
from pyspark.mllib.tree import GradientBoostedTrees, GradientBoostedTreesModel

model = GradientBoostedTrees.trainRegressor(trainingData,
                                            categoricalFeaturesInfo={},
                                            numIterations=3)

print(model.toDebugString())
```

```
TreeEnsembleModel regressor with 3 trees

  Tree 0:
    If (feature 406 <= 20.0)
     Predict: 0.0
    Else (feature 406 > 20.0)
     Predict: 1.0
  Tree 1:
    Predict: 0.0
  Tree 2:
    Predict: 0.0
```

GBT regression

```
predictions = model.predict(features)

labelsAndPredictions = labels.zip(predictions)

testMSE = labelsAndPredictions.map(lambda (v, p): (v - p) * (v - p)).sum() / float(testData.count())
print('Test Mean Squared Error = ' + str(testMSE))
```

```
Test Mean Squared Error = 0.0833333333333
```

Random Forests vs GBTs

- Number of trees
  - –RFs: more trees reduce variance and the likelihood of overfitting; improves performance monotonically
  - –GBTs: more trees reduce bias, but increase the likelihood of overfitting and performance can start to decrease if the number of trees grows too large
- Parallelization
  - –RFs: can train multiples trees in parallel
  - –GBTs: train one tree at a time
- Depth of trees
  - –RFs: deeper trees
  - –GBTs: shallower trees

Lesson Summary

- Having completed theis lesson, you should be able to:
  - Understand the Pipelines API for Gradient-Boosted Trees
  - Perform classification and regression with RFs and GBTs
  - Understand and use RFs and GBTs parameters
  - Outline the differences between RFs and GBTs regarding its parameters