CS6240 HW5 Report

Xiao Wang **Kaibin Yin**

wang.xiao1@husky.neu.edu **yin.k@husky.neu.edu**

Type of model used and parameters explored for it.

1. Type of model: Random Forest

We choose Random Forest as model because 1. It runs efficiently on large data bases, 2. It is able to handle large amount of features (in our case is 21*21*7) without feature selection. 3. It doesn't over fit easily[1].

2. Parameters

In spark mllib, there is Random Forest[2] module with parameters list below:

1) **algo:** Type of decision tree, either Classification or Regression. In this case is classification.
2) **numClasses**: Number of classes. In this case is 2: 0 for background, and 1 for foreground.
3) **maxBins**: Number of bins used when discretizing continuous features.
4) **impurity**: Impurity measure (discussed above) used to choose between candidate splits. This measure must match the algo parameter. In this case, we use gini index[3] as impurity measurement.
5) **maxDepth**: Maximum depth of each tree in the forest. Deeper trees are more expressive (potentially allowing higher accuracy), but they are also more costly to train and are more likely to overfit. As we have about more than 3000 features, it is reasonable to have trees with 10 or 11 depth at max.
6) **numTrees**: Number of trees in the forest.

[1] "Talk:Random forest - Wikipedia." https://en.wikipedia.org/wiki/Talk%3ARandom_forest. Accessed 19 Apr. 2018.
[2] "MLlib Ensemble guide - Apache Spark." https://spark.apache.org/docs/latest/mllib-ensembles.html. Accessed 19 Apr. 2018.
[3] "Gini coefficient - Wikipedia." https://en.wikipedia.org/wiki/Gini_coefficient. Accessed 20 Apr. 2018.

Pseudo code

1. Pre Process

```
sc.textFile(trainPath).map(line => {
    val data =  input.split(",").map(i => i.toDouble)
    val neighbors = data.slice(0, data.length-1)
    val expanded = transform(neighbors)
    LabeledPoint(data(data.length-1), Vectors.dense(expanded))
})
```

To increase data diversity, we can expand the input with data transforming. Function `mirroring` is to make a mirror copy of original input. Function `rotate90` is to make a copy rotating 90 degree of original input. To get a new sample with rotating 180 degree, we just need to apply `rotate90` twice, and so on for 270 degree.

```
def mirroring(input: Array[Double]): Array[Double] = {
    val res = new Array[input.length]
    for (i = 0 to 10*21*7) {
      val x = i/(21*7)
      val y = i%(21*7)/7
      val z = i%7
      res.update(i, input((20-x)*21*7+y*7+z))
      res.update((20-x)*21*7+y*7+z, input(i))
    }
    return res
  }


def rotate90(input: Array[Double]): Array[Double] = {
    val res = new Array[input.length]
    for (i = 0 to 10*21*7) {
      val x = i/(21*7)
      val y = i%(21*7)/7
      val z = i%7
      res.update(i, input(y*21*7+(20-x)*7+z))
    }
    return res
  }
```

2. Training Phase

```
// training parameters
val numClasses = 2
val categoricalFeaturesInfo = Map[Int, Int]()
val numTrees = 64
val featureSubsetStrategy = "auto"
val impurity = "gini"
val maxDepth = 10
val maxBins = 128

// train phase
val model = RandomForest.trainClassifier(trainData, numClasses,
categoricalFeaturesInfo,
      numTrees, featureSubsetStrategy, impurity, maxDepth, maxBins)
```

3. Validation Phase

```
val labelAndPreds = validData.map { point =>
    val prediction = model.predict(point.features)
        (point.label, prediction)
    }
```

4. Test Phase

```
val testPredict = testData.map(point => model.predict(point.features).toInt)
```

Analysis

1. How many tasks are created during each stage of the model training process?
2. Is data being shuffled?
   Because data are distributed to workers during preprocessing, but training phase need to gather data in some way. So data are being shuffled before training phase.
3. How did changes of parameters controlling partitioning affect the running time?
   numTrees: As it takes similar time to train each tree, the total training time is linear to numTrees
   maxDepth: As the number of nodes in tree is 2^maxDepth, the total training time is 2^maxDepth

Preprocessing

1. Split the input entries into 21*21*7 features
2. (With args[2] isTransform set to true) expand features with 7 transformations
   Suppose all training axons are aligned in one direction, without transformation, the model is very likely to be overfitted and perform badly if the prediction dataset are not aligned in the same direction

Result

Accuracy result

maxDepth numTrees	16	64
8	99.7404%	99.746%
10	99.7439%	99.759%

TP = True Positive

TN = True Negative

FP = False Positive

FN = False Negative

Precision = TP/(TP+FP)

Recall = TP/(TP+FN)

maxDepth	numTrees	TP	TN	FP	FN	Precision	Recall
8	16	6350	963885	1938	587	0.766168	0.915381
10	16	6740	963799	1818	673	0.787567	0.909213
8	64	6559	963808	1792	664	0.785414	0.908071
10	64	6581	963839	1707	633	0.794039	0.912253

Training Time

maxDepth numTrees	16 (9 workers)	64 (16 workers)
8	1876s	1866s
10	1924s	3254s