CS6240 – FINAL PROJECT REPORT

SANKAR GIREESAN NAIR
ROHIT PATNAIK

FOREGROUND-BACKGROUND CLASSIFICATION

MODEL USED:

Random Forest Model:

Random forests are ensembles of decision trees. Random forests are one of the most successful machine learning models for classification and regression. **We were getting a total accuracy of ~99%.** They combine many decision trees to reduce the risk of overfitting.

Parameters used:

- **numTrees**: Number of trees in the forest.
  - Random Forest trains the models in parallel.
  - Increasing the number of trees decreased the variance in predictions, improving the model's accuracy.
  - Training time increases roughly linearly in the number of trees.
- **maxDepth**: Maximum depth of each tree in the forest.
  - Increasing the depth makes the model more expressive and powerful. However, deep trees take longer to train and are also more prone to overfitting.
  - In general, it is acceptable to train deeper trees when using random forests than when using a single decision tree. One tree is more likely to overfit than a random forest (because of the variance reduction from averaging multiple trees in the forest).

Increasing the number of trees increased the accuracy for us. Thought the run time increased.
The maximum number of trees allowed is 30.

Increasing the maxDepth should have increased the accuracy. However, given the data we are handling, in which the label "0" dominates number of label "1" records, the model tries to overfits the data. This counter affects the accuracy.

Source: https://spark.apache.org/docs/2.1.0/mllib-ensembles.html

PSEUDO CODE:

Preprocessing Training Data:

1. TrainingData = Read 4 Image files
2. Split each record with ","
3. Convert each row to a LabeledPoint(Label, Array[features]
4. Balancing the Training Data with equal number of labeled 0s and labeled 1

5. Create 5 Training samples with replacement using Bootstrap Sample Technique

Training Models:

1. Use the 5 above samples to create 5 independent Random Forest models
2. Save each model to S3

Validation:

1. Retrieve all 5 models from S3
2. Create an EnsembleModel to predict on the Validation Data
3. The validation data is filtered to produce two sets. One with all "1" as label and one with all "0" as label.
4. Individual accuracy is calculated for both the sets.
5. The Output of Prediction is determined on Majority votes
6. The final accuracy is calculated for Validation Data and printed

Prediction:

1. Retrieve all models from the provided input path
2. Read testData
3. Split each record with ","
4. Extract the features out of each row
5. Predict the label using the Ensemble Model
6. Save the output label file to the given output path

Q: How many tasks are created during each stage of the model training process?

190 Tasks per Stage

Q: Is data being shuffled?

Yes. We are using bootstrap sampling which will shuffle the data randomly to create new samples.

Q: How many iterations are executed during model training (for methods that have multiple iterations)?

Our primary model is Random Forest and the different trees are ran parallel.

Q: How did changes of parameters controlling partitioning affect the running time? E.g., for bagging, was it better to partition the model file, the test data, or both?

5 RF Models with 4 image files as input with no sampling (each model gets 20% of total data) took 1 hour 43 mins using 21 m4.large machines to train the model and validating it.

5 RF Models with 3 image files as input with bootstrap sampling took 4 hours 26 mins using 21 m4.large machines to train the model and validating it.

PREPROCESSING STEP:

1. TrainingData = Read 4 Image files
2. Split each record with ","
3. Convert each row to a LabeledPoint(Label, Array[features]
4. Balancing the Training Data with equal number of labeled 0s and labeled 1

Why?

Spark mllib models require an RDD[LabeledPoint] to train. We are balancing the labeled points Training data to equal number of zeros and ones. This is done because the accuracy of any Machine Learning Classifier depends a lot on training data balancing.

Approach 1:

Create 5 samples using random split where each model received 20% of the entire training data

Approach 2:

Create 5 Training samples with replacement using Bootstrap Sample Technique.

Explanation:

In the first approach, the total Training Data size was 24 GB and in the second approach, the total Training Data size was increased to 60 GB to train the 5 RF models with more data to improve the accuracy.

Validation Data:

Read 1 Image file (Different from training data)

The data is categorized into two sets: one with all "1" and one with all "0"

The prediction is done separately on both data sets

Why?

Most of the model might predict it correctly for record with label "0", but not for records with label "1". Since the amount of "0" present is 99.7%, the total accuracy will not be affected even if the prediction for records with label "1" is completely wrong.

ACCURACY:

Models	Amount of Data	Number of Trees	Max Depth	Max Bins	Accuracy

Random Forest	34% of Training Data	30	20	3000	78.84%
Random Forest	20% of Training Data	10	5	100	76.6%
Random Forest	100% of Training Data	12	10	100	82.2%

Models	**Amount of Data**	**Number of Boosting Iterations**	**Maximum Boosting Depth**	**Accuracy***
Gradient Boosted Trees	10% of Training Data	3	4	84%
Gradient Boosted Trees	33% of Training Data	5	6	Unable to Train (Didn't finish in 12 hours)

Models	**Amount of Data**	**Accuracy***
Logistic Regression	33% of Training Data	68.88%

Ensemble Models:

Models	**Amount of Data for each model**	**Number of Trees**	**Max Depth**	**Max Bins**	**Accuracy***
3 Random Forest	33% of Training Data (without replacement)	10	5	100	75.04%
5 Random Forest	20% of Training Data (without replacement)	30	20	3000	78.88%
5 Random Forest	50% of Training Data (with replacement)	10	10	100	74.8%
5 Random Forest	50% of Training Data	10	10	100	96.4% (1)

	(with replacement) and Balanced Training Data				98.48% (0) Total Accuracy – 98.47%

N.B.: *All the models gave good accuracy of 99.9 % for the validation data. However, the accuracy mentioned is for all the foreground values. It will be like **recall(tp/(tp+fn)).**

RUNNING TIME and SPEED UP:

Models	Number of Machines	Running Time	Speed Up
Random Forest (3 RF)	11	104	
Random Forest (3 RF)	21	59	**1.76**
Random Forest (5 RF)	11	381	
Random Forest (5 RF)	21	240	**1.59**

Prediction:

Models	Number of Machines	Running Time	Speed Up
Prediction	6	15	
Prediction	11	9	**1.66**

Conclusion:

We tried three different ensemble models –

- Ensemble model with Gradient Boosted Trees, Logistic Regression, and Random Forest. It was done without sampling with replacement with each model getting 1/3rd of the Training data on AWS. Running this ensemble with small amount of data gave good accuracy, however,

we couldn't train the GBT model on AWS with 1/3rd data because it took around 12-13 hours. When we ran the prediction code on AWS for the models we received for Logistic Regression and Random Forest individually, the overall accuracy for 1 was 66% and 78% respectively.

- Ensemble model with 3 Random Forest models. It was done without sampling with replacement with each model getting 1/3rd of Training data on AWS. The parameters set for the Random Forest model was numTrees=10, maxDepth=5, maxBins = 100. Predicting for a test file on the ensemble gave us a very low accuracy of ~60%.
- Ensemble model with 5 Random Forest models. It was done with sampling with replacement which increased overall Training Data by nearly ~1.5 times the original data. Each model received around 20% of the sampling data. The parameters set for the Random Forest model was numTrees=10, maxDepth=5, maxBins = 100. Predicting for a test file on the ensemble gave us an accuracy of 74.8%.
- **FINAL MODEL USED: We realized that regardless of the model we use, we won't get a good accuracy if the Training Data (as provided) is unbalanced. So, we decided to balance the data with equal number of label record as 1 and label record as 0. This increased the accuracy significantly to 98.5%.**