Smartphone-based Human Activity Recognition Using Machine Learning and Distributed Computing

Team 19: Yuhan Wang, Minchen Wang, Tianqi Wang, Ziqi Pan

January, 2019

1 Introduction

Smart equipment has been a common necessity for people in modern daily life. The portability characteristic makes those smart devices prolific data sources, making it possible to use data collected from human activities to predict certain types of behavior, which is helpful for further research in many other fields such as health monitoring and medical emergency. As different sensors on one device could get thousands of records per day, the volume of recorded data explodes, which means analysis and modeling would be challenging for a single machine. Thus, we built a workflow based on a distributed system including Amazon S3 cloud storage, mongoDB, Amazon EMR cluster and SparkML in this project.

2 Data Description

We use the data collected by the Department of Electrical and Computer Engineering in UCSD. They designed a mobile application which could be used on iPhone and Android smartphones as well as Pebble smartwatch to collect data on both sensor measurements and ground truth labels. The whole data set contains a total of 377,346 labeled examples (minutes) of sensor data from 60 users with 280 features from different sensors (e.g. accelerator, gyroscope, magnetometer, etc), which could be used in the analysis and model.

3 Methodology

In this project, we do binary classification on each of the first five common activities to judge if the subject do one activity or not using Random Forest Classification Method. Considering the size of our data and the Random Forest model we choose, the workflow is designed around distributed data storage and distributed computing, using Amazon S3, EC2 and EMR, MongoDB and Spark.

Figure 1: Data Flow

1. Step 1: Upload the data on S3

   Amazon S3 (Simple Storage Service) is a scalable storage infrastructure that provides object storage through a web service interface. We can easily store and retrieve data on and from S3, and the next step is just to load data from S3 to distributed MongoDB.

2. Step 2: Load S3 data to MongoDB

   We created 10 instances on EC2, three for shard 1 replica set, three for shard 2 replica set, three for mongoDB config server replica set (each with one primary and two secondaries) and the last one for mongos which connects external client with internal distributed database. Each instance has size of t2.micro. After setting everything up, we imported data from S3 to mongoDB and distributed them into two shards, with 39.36% of data on shard 1 and 60.63% of data on shard 2. The final size of these two replica sets is 3.87 GiB in total.

```
mongos> db.sensor.getShardDistribution()

Shard rs2 at rs2/172.31.82.241:27018,172.31.88.93:27018,172.31.92.144:27018
 data : 2.35GiB docs : 228741 chunks : 30
 estimated data per chunk : 80.23MiB
 estimated docs per chunk : 7624

Shard rs1 at rs1/172.31.34.254:27018,172.31.37.218:27018,172.31.39.101:27018
 data : 1.52GiB docs : 148605 chunks : 30
 estimated data per chunk : 52.09MiB
 estimated docs per chunk : 4953

Totals
 data : 3.87GiB docs : 377346 chunks : 60
 Shard rs2 contains 60.63% data, 60.61% docs in cluster, avg obj size on shard : 10KiB
 Shard rs1 contains 39.36% data, 39.38% docs in cluster, avg obj size on shard : 10KiB
```

Figure 2: MongoDB Status For Data Storage

3. Step 3: Run SparkML on EMR cluster

   Based on Amazon EMR (Amazon Elastic MapReduce) cluster, we could run distributed frameworks such as Apache Spark easily and fast to process vast amount of data across different EC2 instances that are internally linked by the EMR framework. For this project, we run three m4.xlarge instances, with one instance set as the master and the other two as cores. Meanwhile, we use YARN as our cluster manager to arbitrate all the available cluster resources and help manage the distributed applications running on the YARN system. Through YARN, we set up the environment in the master instance of EMR and launch the jupyter notebook to use SparkSQL and SparkML for the analysis and modeling part.

4 Modeling

Our model is based on Random Forest algorithm, a tree-based ensembling model. The bagging ensemble characteristic equips this algorithm with great parallel capability, making it more reasonable to deploy the model on a distributed system. To deploy our model in spark environment, we use PySpark ML package.

From all 280 features from the raw data, we choose 172 of them which are numeric and related to the most common sensors. In order to get rid of overfitting, we set 50 as number of trees and 10 as the maximum depth for the purpose of regularization. Besides, to speed

up the whole training process and to better make use of the cluster storage, our model increase the Maximum memory in MB allocated to histogram aggregation.

For the purpose of reporting, our model includes 10 labels with highest frequencies: Indoors, At Homes, Sitting, Phone on Table, Lying Down, Sleeping, At School, Computer Work, Standing and Talking.

Since the data are imbalanced for most target labels, we use F1 score as our major metric and accuracy as supplementary metric in our report. The final modeling evaluation shows as Figure 3.

Human Activity	Accuracy	F1_score
OR_indoors	0.9178	0.9018
LOC_home	0.7436	0.7458
SITTING	0.6599	0.6599
PHONE_ON_TABLE	0.7625	0.7602
LYING_DOWN	0.7702	0.7667
SLEEPING	0.8231	0.8222
AT_SCHOOL	0.9277	0.9031
COMPUTER_WORK	0.7928	0.7599
OR_standing	0.8868	0.8347
TALKING	0.8156	0.7384

Figure 3: Model Evaluation

5 Conclusion

1. Insights from Distributed Database and Computing

   Distributed databases such as MongoDB enable us to store and retrieve data in a faster and safer way because of sharding and replication. On the other hand, the AWS EMR service provides us with a way to distribute computing on different machines within a cluster, and thus largely increases our efficiency in doing large scales of model training. It only takes us around 1.5 minutes to train one random forest model (50 trees and maximum depth of 10) with about 90k lines of data. Since the next stage is to try more advanced ensembling models such as XGBoost and LightGBM, such distributed computing will help a lot to save us time in training and improving our models.

2. Model Performance

   Overall, our random forest model has a satisfying performance, both accuracy f1 score achieve to a high level. Over 10 labels, the average f1 score is 0.789 and the average accuracy is 0.81, metrics fluctuation exits among labels due to divergence of imbalance degree among labels.

6 Further Extension

1. To achieve better model performance, more advanced ensembling model could be used, e.g. Boosting models like XGBoost, LightGBM and stacking models.

2. Hundreds of features from different sensors are included in our model, Neural Netework could be used to deep exploit the informtion behind those huge amounts of features.
3. Due to the sparsity of different labels, F1-Score would not be an appropriate metric for measuring model performance, hence, other metrics could be used to measure the overall performance for multiple behaviors prediction.

References

[1] The ExtraSensory Dataset: A dataset for behavioral context recognition in-the-wild from mobile sensors *http://extrasensory.ucsd.edu/download*. Department of Electrical and Computer Engineering, University of California, San Diego.

[2] Vaizman, Y., Ellis, K., Lanckriet, G., and Weibel, N. *ExtraSensory App: Data Collection In-the-Wild with Rich User Interface to Self-Report Behavior*. Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI 2018), ACM, April 2018. doi:10.1145/3173574.3174128

[3] Vaizman, Y., Ellis, K., and Lanckriet, G. *Recognizing Detailed Human Context In-the-Wild from Smartphones and Smartwatches*. EEE Pervasive Computing, vol. 16, no. 4, October-December 2017 , pp. 62-74. doi:10.1109/MPRV.2017.3971131