USE OF SPARK MLLIB FOR PREDICTING OFFLINING OF DIGITAL MEDIA

Christopher Burdorf
NBCUniversal

OVERVIEW

- Problem Definition
- Cluster Configurations
- Parameters
- MLLib libraries utilized
- Results
- Conclusions

Problem Definition

- Digital media files distributed internationally 24/7 over internet to cable TV channels in APAC, Europe, and Asia.
- TV Shows, Commercials
- Fast Isilon storage fills up frequently, thus offlining is necessary to create space for new files
- Multiple parameters are used to pick candidates
- Previous system works well, but is slow and has large overhead

Cluster Configurations

- Development
  - 1 Linux System with 16 cores and 16 Gigs RAM
  - 3 Mac OS Systems each with 8 cores and 16 Gigs RAM
  - Mesos cluster manager
- Production
  - 3 Linux systems with 32 cores and 64 Gigs of RAM
  - Mesos master and slaves running in Docker containers
  -

Features/Parameters

- File size
- File age
- Days since last airing
- Days until next airing
- Immediately remove files that have not been scheduled for more than 3 weeks (was 2 weeks to start with).

K-Means Clustering

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} \left\| x_i^{(j)} - c_j \right\|^2$$

where xi(j) -cj is a chosen distance measure between a data point and the cluster centre , is an indicator of the distance of the n data points from their respective cluster centers.

K means clustering results

- Attempted with multiple centroid points
- No meaningful clusters for any sets of data attempted
- Results appeared to be tied to the nature of the data
- Multiple parameters created difficulties

Naive Bayes classifier

- From Bayes theorem: p(C_k|x)=(p(C_k)p(x|C_k))/p(x)
- posterior = (prior X likelihood)/evidence
- Thus, the attempt was to classify media files for removal or retention on the file system based on the different parameter values and their expected results
- Had difficulties training for many factors
  - Adding new factors to training set altered results for previously trained classifier and required retraining the entire classification system

Support Vector Machines

To obtain the optimal hyperplane, one solves the following convex quadratic optimization problem with respect to weight vector w and bias b:

$$\min_{w,b} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} \epsilon_i, \tag{1}$$

subject to the constraints:

$$y_i(\langle w, \phi(x_i) \rangle + b) \geq 1 - \epsilon_i, \quad \epsilon_i \geq 0 \quad \text{for } i = 1 \ldots, n \tag{2}$$

The regularization parameter C controls the trade-off between maximizing the margin $1/\|w\|$ and minimizing the sum of slack variables of the training examples

$$\epsilon_i = max(0, 1 - y_i(\langle w, \phi(x_i) \rangle + b)) \text{for } i = 1, \ldots, n. \tag{3}$$

The training example x_i is correctly classified if $0 \leq \epsilon_i < 1$ and is misclassified when $\epsilon_i \geq 1$.

SVM Results

- Found to be most robust method
- Compensated well for new added features
- Built training set from production data and auto-generated data that fit the criteria
- Spark Mllib optimization allows us to build a predictive system in just over an hour
  - Run twice daily due to constant changes in schedules and online media.

Production environment

- Spark Mllib SVM is trained and predictions are generated for each media file online twice daily. Predictions are stored in HBase
- Media manager daemon regularly scans the file system and if available space requires purging, it will select files based on predictions stored in Hbase
- Backup system checks to see if schedule has changed for a given media file selected to be purged, because Spark SVM predictions are only generated twice daily

Production issues

- System was run in test mode for 6 weeks without a problem
- Once switched to production, issues arose
  - 2 weeks was too short a time to immediately offline files
    - Switched to 3 weeks
  - Docker containers filled up with Spark temp files and crashed over the weekend (ouch!)
    - Solved with cron job periodically removing them.
  - Web service access to Hbase locked up.
    - Solved by having thread timeout on web services call.

Conclusions

- System has been running in production mode for some time now.
  - Fine tuning appears to be complete
- Spark SVM has performed well
  - Fast and robust
- Good application for machine learning though critical aspect of the system increased task complexity.