Large Scale (Machine) Learning at Twitter

Aleksander Kołcz
Twitter, Inc.
What is Twitter

• Micro-blogging service
• Open exchange of information/opinions
  – Private communication possible via DMs
• Content restricted to 140 characters
• Seems tiny but ...
#egypt, #tunisia, #libya, #syria ...
#ocws, #occupywallstreet, ...
The Scale of Twitter

- Twitter has more than 140 million active users
- 340 million Tweets are sent per day
- Over 400 million monthly unique visitors to twitter.com
- 50 million people log into Twitter every day
Large scale infrastructure of information delivery

- Users interact via web-ui, sms, and various apps
- Over 55% of our active users are mobile users
- Real-time redistribution of content
- Plus search and other services living on top of it
  - E.g., 2.3B search queries/day (26K/sec)
Support for user interaction

• Search
  – Relevance ranking

• User recommendation
  – WTF or Who To Follow

• Content recommendation
  – Relevant news, media, trends

• ML is an important component but really just a part of a larger whole
Problems we are trying to solve

• Relevance
  – ranking in search
Problems we are trying to solve

• Who to follow
Problems we are trying to solve

Content recommendation (stories/media)
(other) problems we are trying to solve

• Trending topics
• Language detection
• Anti-spam
• Revenue optimization
• User interest modeling
• Growth optimization
Recommendation/Personalization

- Other users – friends
- Other users – opinion sources
- News/information sources
- Specific tweets
- Specific urls
- Lists/hashtags

Clustering/grouping
Categorization/Tagging
Is this BIG data?
Challenges

• Twitter is International
  – 70% of accounts are outside the US
  – Twitter supports more than 28 different languages

• Twitter is real time
  – The “now” factor is very important for people interacting with Twitter
  – The concept of relevance is highly time dependent

• 140 characters: “documents” are very short and vocabulary contains many non-standard acronyms
  – @XXXX_Cool
  – Please please please
What type of machine learning?

• At this point ML is an extensive and diverse field

• Algorithms differ widely in terms of implementational and operational complexity

• Decisions which techniques to use are driven by the nature of the problems and infrastructure/operational constraints
ML for social networks

• Power of simple models
  – It has been found again and again that with enough data relatively simple models (e.g., Naïve Bayes or Logistic Regression) work remarkably well
  – Ease of integration with data processing flows (both training and inference) trumps model sophistication
ML for social networks

- Power of graph aggregation
  - Even an imperfect signal (e.g., classifier) can be useful when its effect on a graph node is taken in the context of the node’s neighborhood.
ML for social networks

- Challenges of scalability and adaptive processing
  - Learning and inference need to handle data streams and combinations thereof
  - Models need to be able to quickly adapt to data stream change
Are there wheels not to reinvent?

- Temptation is always there ....
- Data processing infrastructure?
- Core ML algorithm libraries ?
- Data pre/post processing ?
- Visualization ?
- Glue code ?
Analytics Ecosystem

- Apache Mesos
- Hadoop
- Cloudera
- Mallet
- ZooKeeper
- Mahout
- Cassandra
- Apache HBase
- Scala
- Twitter
Maximizing the use of Hadoop

• We cannot afford too many diverse computing environments

• Most of analytics job are run using Hadoop cluster
  – Hence, that’s where the data live
  – It is natural to structure ML computation so that it takes advantage of the cluster and is performed close to the data
AVOID: “janky” analysis of messy data

- Too Much Data
  - Run Model on all data
  - Build model on a local box

- Down-Sample
  - Probably lose intermediate results and data
Leveraging off-line tools

• While the data is big, a lot of useful feature components can be learned with smaller datasets.
• The ML tool ecosystem provides a wide range of options for optimization and tuning various models.
• Once tuned, the models can be applied to big data in a distributed fashion — often not as final models but as feature extractors.
• The key is not to rely primarily on ad-hockery in production pipelines.
Large scale learning frameworks

• Participation in the open source community
  – There are a number of initiatives for using ML over Hadoop (e.g., Mahout)

• Upsides
  – Larger support and developer network

• Downsides
  – Not always convenient to integrate with internal analytics and data processing flows
Our extensions

• PigML
  – A library of UDFs wrapping the ML functionality

• Scalding/PyCascading
  – Cascading with Scala/Jython

• ML Java lib
  – Used in both cases to capture the low level ML functionality
Build/reuse/integrate

- ML library
- External/Open source + internal
- PigML / Scalding
- Big Data processing pipeline
- Community (Hadoop/Pig)
MapReduce

Shuffle and Sort: aggregate values by keys

Source: Lin and Dyer (2010)
import java.io.IOException;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

        JobControl jc = new JobControl("Find top 100 sites for users 18 to 25");
        jc.addJob(loadPages);

import org.apache.hadoop.mapred.lib.IdentityMapper;
import org.apache.hadoop.mapred.TextInputFormat;
import org.apache.hadoop.mapred.SequenceFileOutputFormat;
import org.apache.hadoop.mapred.Reporter;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.io.Writable;
import org.apache.hadoop.io.LongWritable;
import java.io.IOException;

public static class LoadAndFilterUsers extends MapReduceBase {

    public void map(LongWritable k, Text val, 
                    Reporter reporter) throws IOException {
        String line = val.toString();
        int firstComma = line.indexOf(',') + 1;
        int secondComma = line.indexOf(',', firstComma);
        String key = line.substring(firstComma, secondComma);
        String value = line.substring(firstComma + 1);
        if (value.charAt(0) == '1') {
            outKey = new Text(key);
            outVal = new Text("1" + value);
            oc.collect(outKey, outVal);
        }
    }

    public void reduce(Text key, 
                       Writable val, 
                       OutputCollector<Text, Writable> oc, 
                       Reporter reporter) throws IOException {
        String line = val.toString();
        int firstComma = line.indexOf(',') + 1;
        int secondComma = line.indexOf(',', firstComma);
        String key = line.substring(0, firstComma);
        String value = line.substring(firstComma + 1);
        int age = Integer.parseInt(value);
        if (age < 18 || age > 25) return;
        String visitCounts = groupVisits by url;
        String urlInfo = load '/data/urlInfo' as (url, category, pRank);
        String urlInfoVisits = join visitCounts by url, urlInfo by url;
        String gCategories = group visitCounts by category;
        String topUrls = foreach gCategories generate 
                         top(visitCounts,10);
        String cleanTopUrls = topUrls;
        String urlVisits = count(urlInfoVisits);
        String length = urlVisits.length();
        String split = urlVisits.substring(0, length - 1);
        String key = urlInfoVisits;
        String value = split;
        String secondComma = line.indexOf(',', firstComma);
        String key = line.substring(0, firstComma);
        String value = line.substring(firstComma + 1);
        String key = line.substring(firstComma, secondComma);
        String value = line.substring(secondComma + 1);
        int age = Integer.parseInt(value);
        if (age < 18 || age > 25) return;
        String visitCounts = groupVisits by url;
        String urlInfo = load '/data/urlInfo' as (url, category, pRank);
        String urlInfoVisits = join visitCounts by url, urlInfo by url;
        String gCategories = group visitCounts by category;
        String topUrls = foreach gCategories generate 
                         top(visitCounts,10);
        String cleanTopUrls = topUrls;
        String urlVisits = count(urlInfoVisits);
        String length = urlVisits.length();
        String split = urlVisits.substring(0, length - 1);
        String key = urlInfoVisits;
        String value = split;
        String secondComma = line.indexOf(',', firstComma);
        String key = line.substring(0, firstComma);
        String value = line.substring(firstComma + 1);
        String key = line.substring(firstComma, secondComma);
        String value = line.substring(secondComma + 1);
        if (value.charAt(0) == '1') {
            outKey = new Text(key);
            outVal = new Text("1" + value);
            oc.collect(outKey, outVal);
        }
    }

    public void reduce(Text key, 
                       Iterator<LongWritable> iterator, 
                       OutputCollector<Text, LongWritable> oc, 
                       Reporter reporter) throws IOException {
        String line = val.toString();
        int firstComma = line.indexOf(',') + 1;
        int secondComma = line.indexOf(',', firstComma);
        String key = line.substring(0, firstComma);
        String value = line.substring(firstComma + 1);
        if (value.charAt(0) == '1') {
            outKey = new Text(key);
            outVal = new Text("1" + value);
            oc.collect(outKey, outVal);
        }
    }

    public void map(Text key, 
                    Text val, 
                    OutputCollector<Text, Text> oc, 
                    Reporter reporter) throws IOException {
        String line = val.toString();
        int firstComma = line.indexOf(',') + 1;
        int secondComma = line.indexOf(',', firstComma);
        String key = line.substring(0, firstComma);
        String value = line.substring(firstComma + 1);
        String key = line.substring(0, firstComma);
PigML

• Embedding ML learning/eval functionality directly in Pig
• Flows naturally with other data processing operations
• Minimal learning curve for users
Training a model in Pig

```
training = load 'trn_data' using piggybank.ml.Storage() as (target: double, features: map[]);

store training into 'model-LR' using piggybank.ml.train.online.LRClassifierBuilder('with Pegasos withLambda:0.1');
```
Training a model in Pig

training = load 'trn_data' using piggybank.ml.Storage() as (target: double, features: map[]);

store training into 'model-LR' using piggybank.ml.train.online.LRCClassifierBuilder('with Pegasos withLambda:0.1');
Training a model in Pig

It’s just a store function!

```
training = load 'trn_data' using piggybank.ml.Storage() as (target: double, features: map[]);

store training into 'model-LR' using piggybank.ml.train.online.LRClassifierBuilder('with Pegasos withLambda:0.1');
```
Applying a model in Pig

DEFINE Classify
    piggybank.ml.classify.ClassifyFeaturesWithLRClassifier('model-LR');

data = load 'test_data' using piggybank.ml.Storage() as (target: double, features: map[]);

data = foreach data generate target, Classify(features) as prediction;

results = foreach data generate (target == prediction.label ? 1 : 0) as matching;

dump results;
Applying a model in Pig

DEFINE Classify
    piggybank.ml.classify.ClassifyFeaturesWithLRClassifier('model-LR');

data = load 'test_data' using piggybank.ml.Storage() as (target: double, features: map[]);

data = foreach data generate target, Classify(features) as prediction;

results = foreach data generate (target == prediction.label ? 1 : 0) as matching;

dump results;
Model training UDF internals

• A single node in the Hadoop cluster does not have extensive memory resources
• The learner cannot cache too much data
• Natural fit for:
  – Stochastic gradient descent (SGD), possibly with mini-batching
  – Effective for streaming the whole dataset through a single learner
Supervised classification in a nutshell

Given

\[ D = \{(x_i, y_i)\}_{i=1}^n \]

Induce

\[ f : X \rightarrow Y \]

s.t. loss is minimized

\[ \text{empirical loss} = \frac{1}{n} \sum_{i=0}^{n} \ell(f(x_i), y_i) \]

Consider functions of a parametric form:

\[ \arg \min_{\theta} \frac{1}{n} \sum_{i=0}^{n} \ell(f(x_i; \theta), y_i) \]

Key insight: machine learning as an optimization problem!

(closed form solutions generally not possible)
Gradient Descent

\[ w^{(t+1)} = w^{(t)} + \gamma^{(t)} \frac{1}{n} \sum_{i=0}^{n} \nabla l \left( f(x_i; \theta^{(t)}), y_i \right) \]

“batch” learning: update model after considering all training instances

Stochastic Gradient Descent (SGD)

\[ w^{(t+1)} = w^{(t)} + \gamma^{(t)} \nabla l \left( f(x; \theta^{(t)}), y \right) \]

“online” learning: update model after considering each (randomly-selected) training instance

In practice… just as good!

Solves the iteration problem!

What about the single reducer problem?
Ensembles

• Classifier committees are one of the best performing types of learners
• Some of these algorithms are sequential (not very MR friendly)
  – Boosting
• But others rely mostly on randomization
  – Each learner is trained over a different split (features and/or instances) of the data
Classifier Training

Pig storage function

map

reduce

label, feature vector

model

previous Pig dataflow

Making Predictions

model

UDF

prediction

model

UDF

prediction
Ensembles: continued

• Ensembles of linear classifiers
  – E.g., each trained using SGD over different subset of features

• Ensembles of decision trees (random forest)
  – Each tree is seeing only a subset of the dataset and node split variables are randomized at each split
Further advantages of parallelism

• Although stream based learners look at all of the data, a number of them can be executed in parallel
  – Effective for tuning hyper-parameters

• Generative models such as Naïve Bayes are naturally well suited to distributed learning
  – Just counting
Example: tweet sentiment detection

- Training/Test data: tweets with emoticons
  - 😊 😞
- Emoticons provide surrogate labels for training
  - Emoticons are removed from the data
- Logistic Regression trained over character 4grams
- Single classifier vs. use of ensembles
Ensembles with 10m examples better than 100m single classifier!

"for free"

Diminishing returns…

single classifier

10m ensembles

100m ensembles
Iterative algorithms

• What if one **REALLY** wants to iterative over big data till convergence?
  – Unrolling of small loops in Pig
  – Implementing the algorithm in Cascading with Scala or Python
  – Using Python Control Flow in Pig
  – Offload to custom MR job (e.g., Mahout)
Example: modeling topic distribution

- Latent Dirichlet Allocation (LDA)
  - Bayesian soft-clustering technique
  - Each document can naturally belong to multiple clusters with degree-of-membership
  - Popular in topics modeling for text data

- Why should we care?
  - Topics capture user interests
  - Knowledge of interests makes recommendation easier
Mahout/PigML integration

Data preprocessing
PigML

Mahout
Iterate LDA till convergence

LDA inference
PigML
LDA applications

- User interest modeling facilitates user matching as well as predicting engagement.
- It does not directly solve the more general problem of finding users who might be interested in a tweet, story, web-page, etc.
- LDA offers a probabilistic model of user interests (e.g., based on what they tweet about).
Anchoring LDA

- Runs of LDA can lead to significantly different results from one run to the next
- This may hamper the usefulness/interpretability of the results
- Anchoring clusters based on the known tag labels regularizes the clustering procedure
  - Also known as Labeled LDA (LLDA)
ML/Data Mining we contribute to

- Cassowary
  - Large graph mining in Java (single box)
- Pig
- Scalding
  - Cascading with Scala
- PyCascading
  - Cascading with Python
- Mahout
ML outside of Twitter

• What does the academic community consider important?
  – Sentiment detection
  – Event tracking (e.g., epidemics)
  – Politics
  – Locality detection
  – Spam detection
Publications mentioning …

- Google Scholar (late 2011)

<table>
<thead>
<tr>
<th>Query</th>
<th>Result count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Twitter spam</td>
<td>9,800</td>
</tr>
<tr>
<td>Web spam</td>
<td>52,300</td>
</tr>
<tr>
<td>Email spam</td>
<td>41,600</td>
</tr>
<tr>
<td>IM spam</td>
<td>30,000</td>
</tr>
<tr>
<td>SMS spam</td>
<td>10,500</td>
</tr>
</tbody>
</table>
Quick search

- Number of titles on Amazon (early 2011)

<table>
<thead>
<tr>
<th>Query</th>
<th>Result count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Twitter marketing</td>
<td>544</td>
</tr>
<tr>
<td>Twitter profits</td>
<td>100</td>
</tr>
<tr>
<td>Twitter money</td>
<td>236</td>
</tr>
<tr>
<td>Twitter rich</td>
<td>36</td>
</tr>
<tr>
<td>Twitter seo</td>
<td>21</td>
</tr>
</tbody>
</table>
Spam/spammer modeling

• Why spam Twitter (or other social media)?
• The types of spam seen
  – @replies/@mentions
  – Trend/search spam
  – Follow spam
• Long term vs. short term
  – Spammers favor “pump and dump”
  – Swift reaction to attacks is key
Example: normal interactions
Example: spammy interactions
ML @ Twitter

- Kumman Chellapila
- Jimmy Lin
- Yue Lu
- Jake Mannix
- Gilad Mishne
- Ram Ravichandran
- Miguel Rios
- Andy Schlaikjer
- Yifan Shi
- +others
THANK YOU