Google Prediction API
Machine Learning as a Service on the Cloud

Max Lin
Software Engineer
Google Research NYC
Google Prediction API

Machine Learning as a Web Service on the Cloud
- Google's machine learning algorithms and infrastructure
- HTTP RESTful web service
- Train asynchronously; predict in real time

- Easy to use, fast to integrate
- Smarter apps
- Automatic model development
How does it work?

The Prediction API finds relevant **features** in the data during **training**.

<table>
<thead>
<tr>
<th><strong>&quot;english&quot;</strong></th>
<th>The quick brown fox jumped over the lazy dog.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>&quot;english&quot;</strong></td>
<td>To err is human, but to really foul things up you need a computer.</td>
</tr>
<tr>
<td><strong>&quot;spanish&quot;</strong></td>
<td>No hay mal que por bien no venga.</td>
</tr>
<tr>
<td><strong>&quot;spanish&quot;</strong></td>
<td>La tercera es la vencida.</td>
</tr>
</tbody>
</table>

The Prediction API later searches for those **features** during **prediction**.

<table>
<thead>
<tr>
<th>?</th>
<th>To be or not to be, that is the question.</th>
</tr>
</thead>
<tbody>
<tr>
<td>?</td>
<td>La fe mueve montañas.</td>
</tr>
</tbody>
</table>
A virtually endless number of applications

<table>
<thead>
<tr>
<th>Customer Sentiment</th>
<th>Transaction Risk</th>
<th>Species Identification</th>
<th>Message Routing</th>
<th>Diagnostics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Churn Prediction</td>
<td>Legal Docket</td>
<td>Suspicious Activity</td>
<td>Work Roster Assignment</td>
<td>Inappropriate Content</td>
</tr>
<tr>
<td>Recommend Products</td>
<td>Political Bias</td>
<td>Uplift Marketing</td>
<td>Email Filtering</td>
<td>Career Counselling</td>
</tr>
</tbody>
</table>

... and many more ...
Three steps to use the Prediction API

1. Upload
   - Upload your training data to Google Storage

2. Train
   - Build a model from your data

3. Predict
   - Make new predictions

   Use the Google Storage API or gsutil to upload your {data} to Google Storage

   POST prediction/v1/training/{data}
   make a training request

   GET prediction/v1/training/{data}
   get training status

   POST prediction/v1/training/{data}/predict
   make a prediction request
Analyzing Big Data

Challenges

- Provision large number of machines
- Develop scalable machine learning algorithms
- Integrate and deploy models with existing apps
Make Big Data Analysis Easy

With machine learning as a web service
- Don't need to provision large number of machines
- Don't require substantial investment upfront
- Don't require deep machine learning expertise
- Easy to integrate with existing apps and deploy models
Automatically categorize and respond to emails by language

- Customer: ACME, a multinational organization
- Goal: Respond to customer emails in their languages
- Data: Many emails, tagged with their languages
- Outcome: Predict language and respond accordingly
Step 1: Upload

Upload your training data to Google Storage
  ● Training data: outputs and input features
  ● Data format: comma separated value format (CSV)

$ head -n 2 ${data}
"english","To err is human, but to really ..."
"spanish","No hay mal que por bien no venga."

Upload to Google Storage
$ gsutil cp ${data} gs://${bucket}/${object}
Step 2: Train

Create a new model by training on data
To train a model:

POST prediction/v1/training/${data}
Training runs asynchronously.
To see if it finishes:

GET prediction/v1/training/${data}

{"data": {
   "data": "${data}'",
   "modelinfo": "estimated accuracy: ${acc}'"}}
Step 3: Predict

Make predictions on new data in JSON format

POST prediction/v1/training/${data}/predict

{"data": {
  "input": {
    "text": ["Cada niño es un artista"]
  }
}}
Step 3: Predict

Apply the trained model to make predictions on new data

POST prediction/v1/training/${data}/predict

{"data": {
    "input": {
        "text": ["Cada niño es un artista"]
    }
}}

{"data": {
    "output": {
        "output_label": "Spanish"
    }
}}
Step 3: Predict

Make predictions on new data in Python

import urllib2

headers = {"Content-Type": "application/json"}test_data = {"data": {"input": {"text": "Cada nino es un artista"}}}
urllib2.Request("https://www.googleapis.com" + "/prediction/v1/training/%s/prediction" % data, data = simplejson.dumps(test_data),headers = headers)
Google Prediction API Capabilities

Data
- Input Features: numeric or unstructured text
- Output: up to 100s of discrete categories

Training
- Many machine learning techniques
- Automatically selected
- Performed asynchronously

Access from many environments
- Web app, mobile app, desktop app
- Programming language independent
Supervised Learning on Large Data

Shard data to overcome memory limitation
MapReduce as building blocks

Accuracy, training time, and resource trade-off

Some strategies for large-scale supervised learning:
1. Sub-sample
2. Embarrassingly parallelize some algorithms
3. Distributed gradient descent
4. Majority Vote
5. Parameter mixture
6. Iterative parameter mixture
1. Sub-Sampling

Big Data

Shard 1

Discarded Data

Machine 1

A Classifier

Sub-sample data

Train on single machine
2. Distributed Naive Bayes Training
3. Distributed Gradient Descent

Big Data

Shard 1  Shard 2  ...  Shard m

Mapper 1  Mapper 2  ...  Mapper m

Shard data

Compute Gradients

Sum up gradients and update parameters

Reducer

Conditional Maximum Entropy

updated weights
4. Majority Vote

Big Data

Shard data
- Shard 1
- Shard 2
- ... (m)

Maximize weights locally
- Mapper 1
- Mapper 2
- ... (m)

Save local models
- Model 1
- Model 2
- ... (m)

Majority Vote
5. Parameter Mixture

Big Data

Shard 1  Shard 2  ...  Shard m

Mapper 1  Mapper 2  ...  Mapper m

Reducer

Conditional Maximum Entropy

Maximize weights locally

Mix weights
Compare Different Training Strategies

Parameter mixture vs. distributed gradient descents
- Parameter mixture has significantly reduced network use
- Converges to the same point and at a comparable rate
On 7 data sets ranging 1M to 1B instances
- Parameter mixture achieves comparable or higher accuracies
- Three orders of magnitude less network usage
- Modestly reduce wall clock time by 15+%  

See Mann, et al., "Efficient Large-Scale Distributed Training of Conditional Maximum Entropy Models", NIPS 2009
5. Parameter Mixture Recap

Big Data

Shard data
- Shard 1
- Shard 2
- Shard m

Maximize weights locally
- Mapper 1
- Mapper 2
- Mapper m

Mix weights

Reducer

Conditional Maximum Entropy
6. Iterative Parameter Mixture

Big Data

Shard data
- Shard 1
- Shard 2
- ... Shard m

Optimize weights in one epoch
- Mapper 1
- Mapper 2
- ... Mapper m

Mix weights

Reducer

Perceptron

updated weights
Compare Parameter Mixture Strategies

![Graph showing the comparison of parameter mixture strategies. The x-axis represents Wall Clock, and the y-axis represents Test Data F-measure. Different strategies are represented by different markers and line styles.]
Compare Parameter Mixture Strategies

- Sub-sampling provides inferior performance
- Parameter mixture improves, but not as good as all data
- Iterative parameter mixture achieves as good as all data
- Distributed algorithms return better classifiers quicker

See McDonald, et al., "Distributed Training Strategies for the Structured Perceptron", NAACL 2010
Google Prediction API

- Machine learning as a service on the cloud
- Make ML easy to use; make every app smarter
- A web service accessible from many platforms
- Strategies on distributed machine learning

To request access and get more information about the Google Prediction API, please go to

http://code.google.com/apis/predict